The Weblate Manual

Versiune 4.8.1

Michal Čihař

sept. 10, 2021

User docs

1 User docs **1**
1.1 Weblate basics . . . 1
1.2 Registration and user profile . . . 1
1.3 Translating using Weblate . . . 10
1.4 Downloading and uploading translations . . . 19
1.5 Glosar . . . 23
1.6 Verificări și reparații . . . 25
1.7 Searching . . . 47
1.8 Translation workflows . . . 52
1.9 Frequently Asked Questions . . . 55
1.10 Supported file formats . . . 62
1.11 Integrarea controlului versiunilor . . . 83
1.12 Weblate's REST API . . . 90
1.13 Weblate Client . . . 134
1.14 Weblate's Python API . . . 138

2 Administrator docs **141**
2.1 Configuration instructions . . . 141
2.2 Weblate deployments . . . 200
2.3 Upgrading Weblate . . . 201
2.4 Copierea de rezervă și mutarea Weblate . . . 208
2.5 Autentificare . . . 215
2.6 Controlul accesului . . . 225
2.7 Proiecte de traducere . . . 234
2.8 Language definitions . . . 250
2.9 Traducere continuă . . . 253
2.10 Licensing translations . . . 263
2.11 Procesul de traducere . . . 264
2.12 Verificări și reparații . . . 270
2.13 Traducere automată . . . 279
2.14 Add-ons . . . 286
2.15 Memorie de traducere . . . 302
2.16 Configurație . . . 303
2.17 Sample configuration . . . 333
2.18 Management commands . . . 348
2.19 Anunțuri . . . 359
2.20 Liste de componente . . . 362
2.21 Optional Weblate modules . . . 363
2.22 Customizing Weblate . . . 368
2.23 Interfața de gestionare . . . 370
2.24 Getting support for Weblate . . . 378

2.25 Legal documents . . . 381

3 Contributor docs **383**
3.1 Contributing to Weblate . . . 383
3.2 Starting contributing code to Weblate . . . 385
3.3 Weblate source code . . . 389
3.4 Debugging Weblate . . . 390
3.5 Weblate internals . . . 391
3.6 Developing addons . . . 393
3.7 Weblate frontend . . . 394
3.8 Reporting issues in Weblate . . . 396
3.9 Weblate testsuite and continuous integration . . . 396
3.10 Data schemas . . . 398
3.11 Releasing Weblate . . . 401
3.12 Security and privacy . . . 402
3.13 Despre Weblate . . . 403
3.14 Licență . . . 404

4 Change History **405**
4.1 Weblate 4.8.1 . . . 405
4.2 Weblate 4.8 . . . 405
4.3 Weblate 4.7.2 . . . 406
4.4 Weblate 4.7.1 . . . 406
4.5 Weblate 4.7 . . . 406
4.6 Weblate 4.6.2 . . . 407
4.7 Weblate 4.6.1 . . . 407
4.8 Weblate 4.6 . . . 408
4.9 Weblate 4.5.3 . . . 408
4.10 Weblate 4.5.2 . . . 408
4.11 Weblate 4.5.1 . . . 409
4.12 Weblate 4.5 . . . 409
4.13 Weblate 4.4.2 . . . 410
4.14 Weblate 4.4.1 . . . 410
4.15 Weblate 4.4 . . . 411
4.16 Weblate 4.3.2 . . . 411
4.17 Weblate 4.3.1 . . . 412
4.18 Weblate 4.3 . . . 412
4.19 Weblate 4.2.2 . . . 413
4.20 Weblate 4.2.1 . . . 413
4.21 Weblate 4.2 . . . 413
4.22 Weblate 4.1.1 . . . 414
4.23 Weblate 4.1 . . . 415
4.24 Weblate 4.0.4 . . . 416
4.25 Weblate 4.0.3 . . . 416
4.26 Weblate 4.0.2 . . . 416
4.27 Weblate 4.0.1 . . . 417
4.28 Weblate 4.0 . . . 417
4.29 Weblate 3.x series . . . 418
4.30 Weblate 2.x series . . . 429
4.31 Weblate 1.x series . . . 440
4.32 Weblate 0.x series . . . 444

Indexul de Module Python **448**

HTTP Routing Table **449**

Index **452**

CAPITOLUL 1

User docs

1.1 Weblate basics

1.1.1 Project and component structure

In Weblate translations are organized into projects and components. Each project can contain number of components and those contain translations into individual languages. The component corresponds to one translatable file (for example *GNU gettext* or *Android string resources*). The projects are there to help you organize component into logical sets (for example to group all translations used within one application).

Internally, each project has translations to common strings propagated across other components within it by default. This lightens the burden of repetitive and multi version translation. The translation propagation can be disabled per *Component configuration* using *Permiteți propagarea traducerii* in case the translations should diverge.

Vezi și:

../devel/integration

1.2 Registration and user profile

1.2.1 Înregistrare

Everybody can browse projects, view translations or suggest translations by default. Only registered users are allowed to actually save changes, and are credited for every translation made.

You can register by following a few simple steps:

1. Fill out the registration form with your credentials.
2. Activate registration by following the link in the e-mail you receive.
3. Optionally adjust your profile to choose which languages you know.

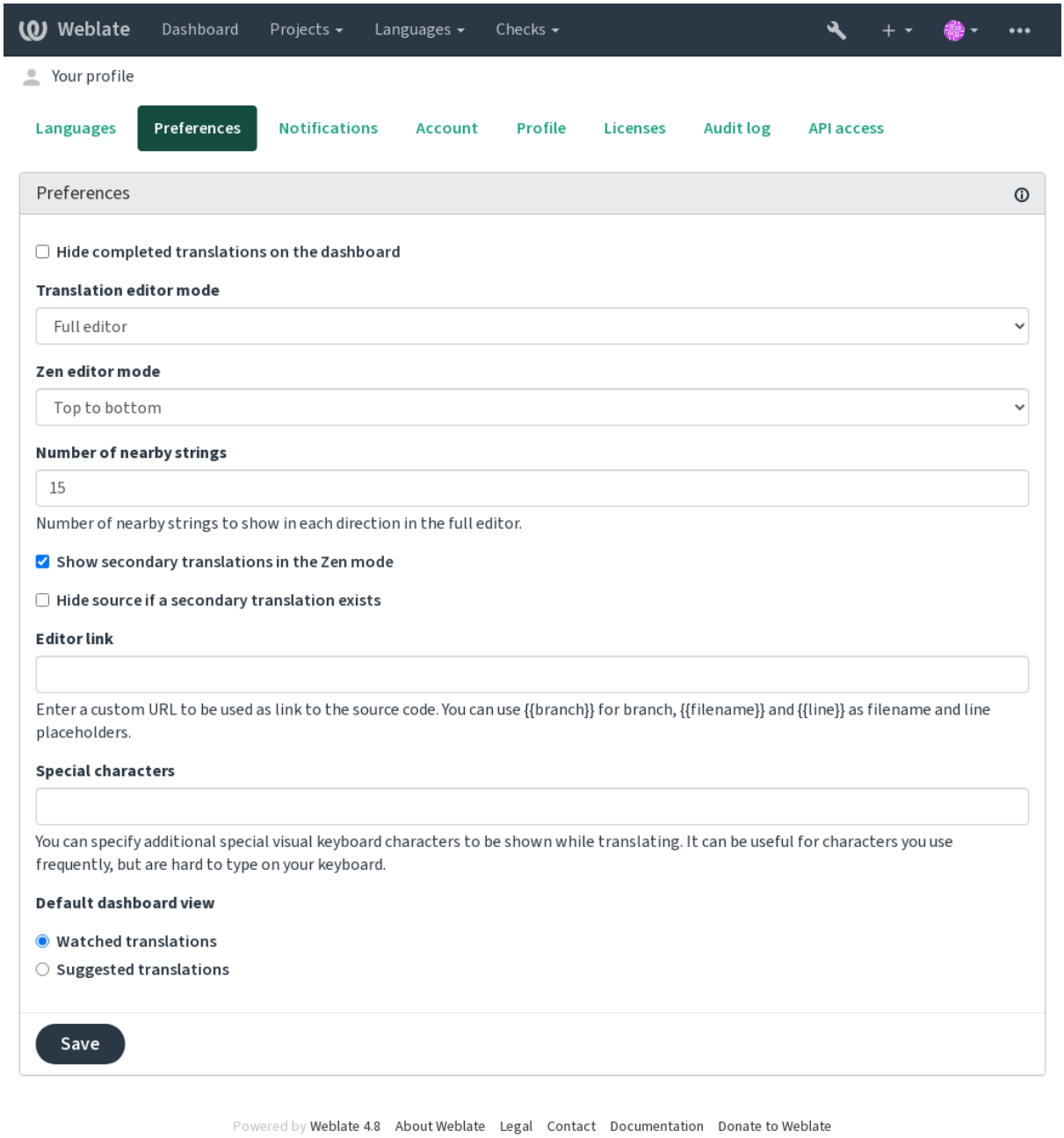
1.2.2 Tabloul de bord

When you sign in, you will see an overview of projects and components, as well as their respective translation progression.

Nou în versiunea 2.5.

Components of projects you are watching are shown by default, and cross-referenced with your preferred languages.

Sugestie: You can switch to different views using the navigation tabs.



The menu has these options:

- *Projects* > *Browse all projects* in the main menu showing translation status for each project on the Weblate instance.
- Selecting a language in the main menu *Languages* will show translation status of all projects, filtered by one of

your primary languages.

- *Watched translations* in the Dashboard will show translation status of only those projects you are watching, filtered by your primary languages.

In addition, the drop-down can also show any number of *component lists*, sets of project components preconfigured by the Weblate administrator, see *Liste de componente*.

You can configure your personal default dashboard view in the *Preferences* section of your user profile settings.

Notă: When Weblate is configured for a single project using `SINGLE_PROJECT` in the `settings.py` file (see *Configurație*), the dashboard will not be shown, as the user will be redirected to a single project or component instead.

1.2.3 Profilul utilizatorului

The user profile is accessible by clicking your user icon in the top-right of the top menu, then the *Settings* menu.

The user profile contains your preferences. Name and e-mail address is used in VCS commits, so keep this info accurate.

Notă: All language selections only offer currently translated languages.

Sugestie: Request or add other languages you want to translate by clicking the button to make them available too.

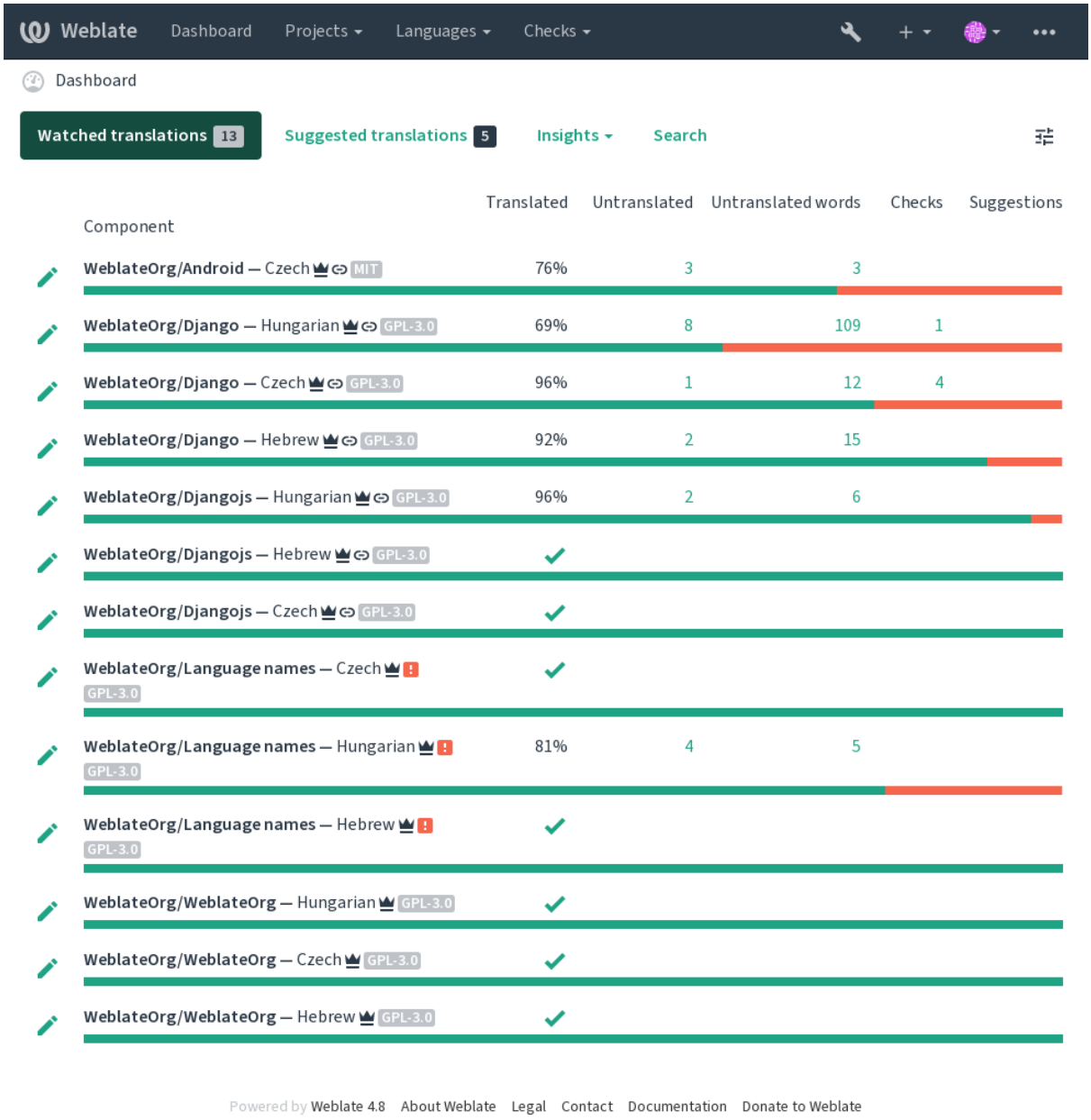
Limbi

1.2.4 Interface language

Choose the language you want to display the UI in.

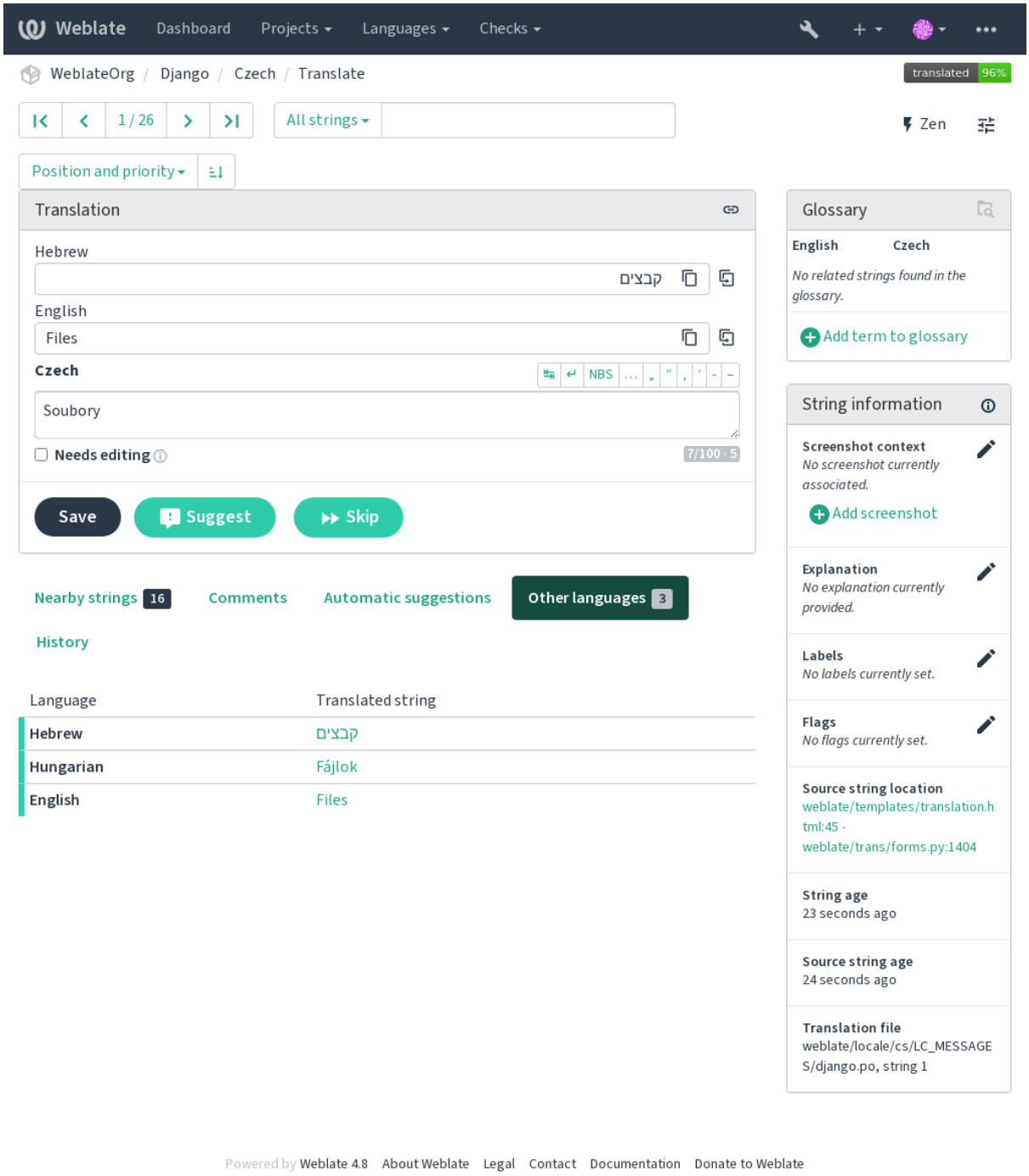
Limbi traduse

Choose which languages you prefer to translate, and they will be offered on the main page of watched projects, so that you have easier access to these all translations in each of those languages.



Limbi secundare

You can define which secondary languages are shown to you as a guide while translating. An example can be seen in the following image, where the Hebrew language is shown as secondarily:



1.2.5 Preferințe

Vizualizarea implicită a tabloului de bord

On the *Preferences* tab, you can pick which of the available dashboard views to present by default. If you pick the *Component list*, you have to select which component list will be displayed from the *Default component list* drop-down.

Vezi și:

Liste de componente

Link editor

A source code link is shown in the web-browser configured in the *Component configuration* by default.

Sugestie: By setting the *Editor link*, you use your local editor to open the VCS source code file of translated strings. You can use *Template markup*.

Usually something like `editor://open/?file={{filename}}&line={{line}}` is a good option.

Vezi și:

You can find more info on registering custom URL protocols for the editor in the Nette documentation.

1.2.6 Notificări

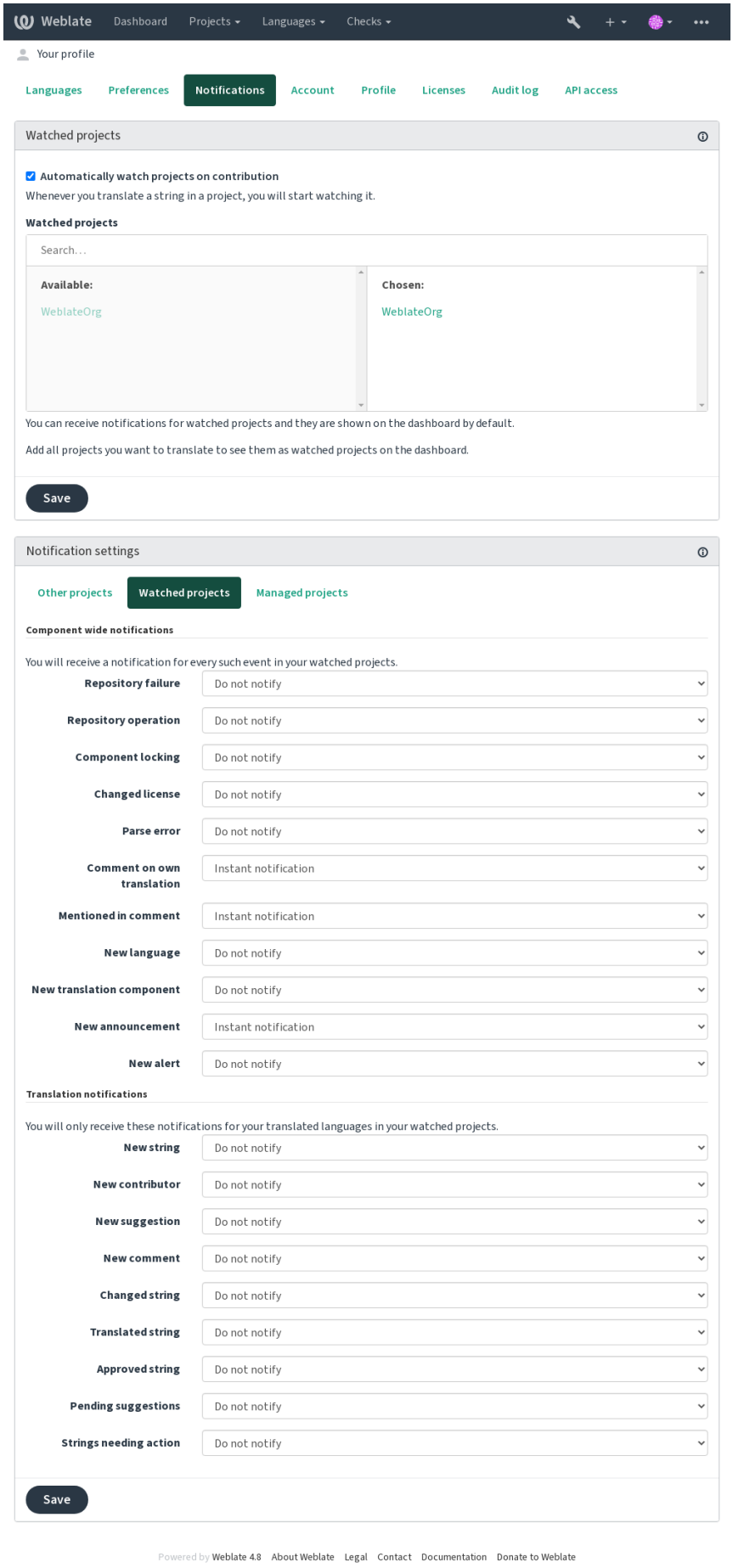
Subscribe to various notifications from the *Notifications* tab. Notifications for selected events on watched or administered projects will be sent to you per e-mail.

Some of the notifications are sent only for events in your languages (for example about new strings to translate), while some trigger at component level (for example merge errors). These two groups of notifications are visually separated in the settings.

You can toggle notifications for watched projects and administered projects and it can be further tweaked (or muted) per project and component. Visit the component overview page and select appropriate choice from the *Watching* menu.

In case *Automatically watch projects on contribution* is enabled you will automatically start watching projects upon translating a string. The default value depends on `DEFAULT_AUTO_WATCH`.

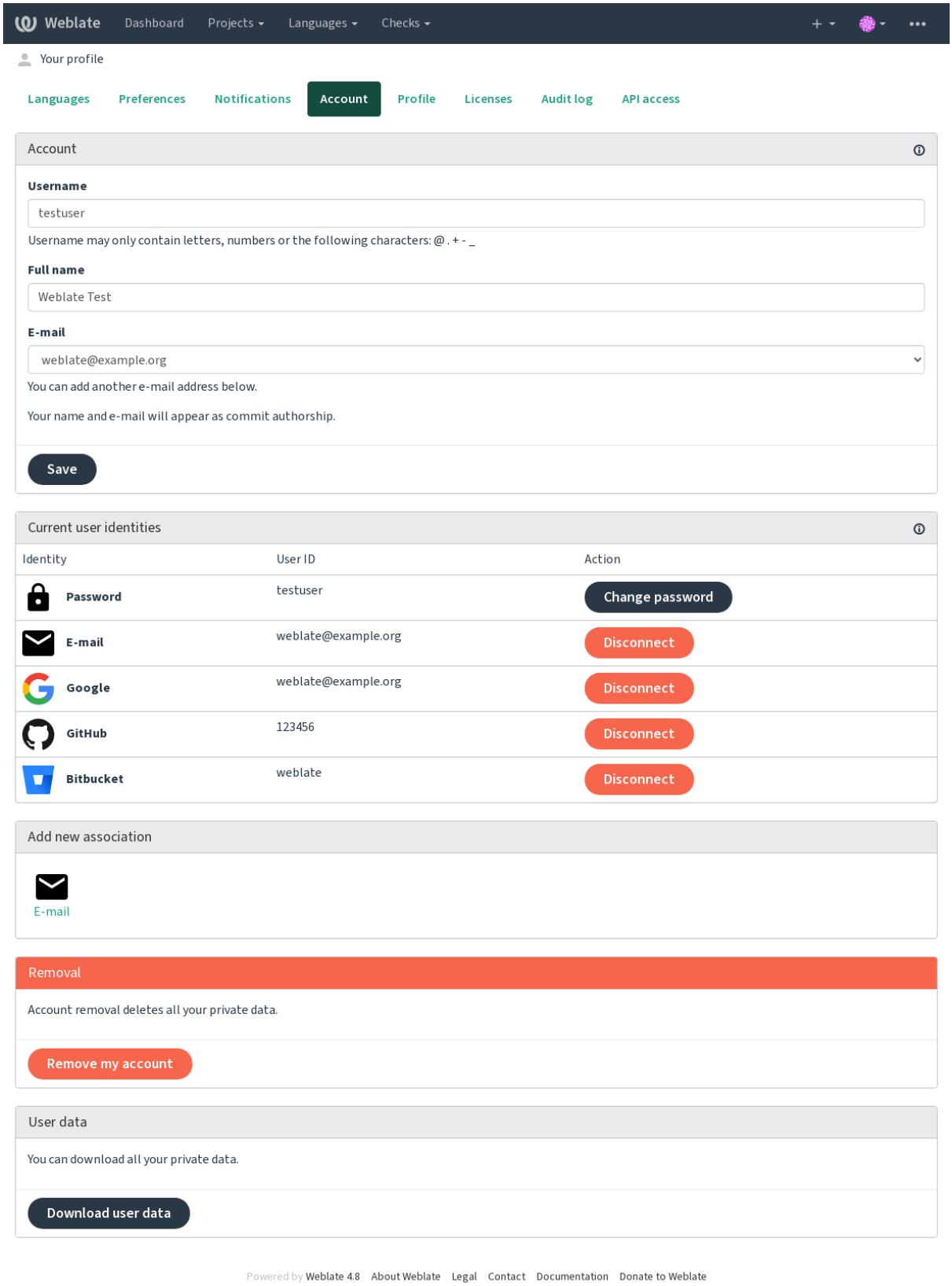
Notă: You will not receive notifications for your own actions.


Weblate
Dashboard
Projects
Languages
Checks
Your profile
Languages
Preferences
Notifications
Account
Profile
Licenses
Audit log
API access
Watched projects
Automatically watch projects on contribution
Whenever you translate a string in a project, you will start watching it.
Watched projects
Search...
Available:
WeblateOrg
Chosen:
WeblateOrg
You can receive notifications for watched projects and they are shown on the dashboard by default.
Add all projects you want to translate to see them as watched projects on the dashboard.
Save
Notification settings
Other projects
Watched projects
Managed projects
Component wide notifications
You will receive a notification for every such event in your watched projects.
Repository failure
Do not notify
Repository operation
Do not notify
Component locking
Do not notify
Changed license
Do not notify
Parse error
Do not notify
Comment on own translation
Instant notification
Mentioned in comment
Instant notification
New language
Do not notify
New translation component
Do not notify
New announcement
Instant notification
New alert
Do not notify
Translation notifications
You will only receive these notifications for your translated languages in your watched projects.
New string
Do not notify
New contributor
Do not notify
New suggestion
Do not notify
New comment
Do not notify
Changed string
Do not notify
Translated string
Do not notify
Approved string
Do not notify
Pending suggestions
Do not notify
Strings needing action
Do not notify
Save
Powered by Weblate 4.8
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.2.7 Cont

The *Account* tab lets you set up basic account details, connect various services you can use to sign in into Weblate, completely remove your account, or download your user data (see *Weblate user data export*).

Notă: The list of services depends on your Weblate configuration, but can be made to include popular sites such as GitLab, GitHub, Google, Facebook, or Bitbucket or other OAuth 2.0 providers.


Weblate
Dashboard
Projects
Languages
Checks
Your profile
Languages
Preferences
Notifications
Account
Profile
Licenses
Audit log
API access
Account
Username
testuser
Username may only contain letters, numbers or the following characters: @ . + - _
Full name
Weblate Test
E-mail
weblate@example.org
You can add another e-mail address below.
Your name and e-mail will appear as commit authorship.
Save
Current user identities
Identity
User ID
Action
Password
testuser
Change password
E-mail
weblate@example.org
Disconnect
Google
weblate@example.org
Disconnect
GitHub
123456
Disconnect
Bitbucket
weblate
Disconnect
Add new association
E-mail
Removal
Account removal deletes all your private data.
Remove my account
User data
You can download all your private data.
Download user data
Powered by Weblate 4.8
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.2.8 Profil

Toate câmpurile de pe această pagină sunt opționale și pot fi șterse în orice moment, iar prin completarea lor, ne dați consimțământul de a partaja aceste date oriunde apare profilul dvs. de utilizator.

Avatar can be shown for each user (depending on `ENABLE_AVATARS`). These images are obtained using https://gravatar.com/.

1.2.9 Licențe

1.2.10 Accesul la API

You can get or reset your API access token here.

1.2.11 Jurnal de audit

Audit log keeps track of the actions performed with your account. It logs IP address and browser for every important action with your account. The critical actions also trigger a notification to a primary e-mail address.

Vezi și:

Running behind reverse proxy

1.3 Translating using Weblate

Thank you for interest in translating using Weblate. Projects can either be set up for direct translation, or by way of accepting suggestions made by users without accounts.

Overall, there are two modes of translation:

- The project accepts direct translations
- The project only accepts suggestions, which are automatically validated once a defined number of votes is reached

Please see *Translation workflows* for more info on translation workflow.

Options for translation project visibility:

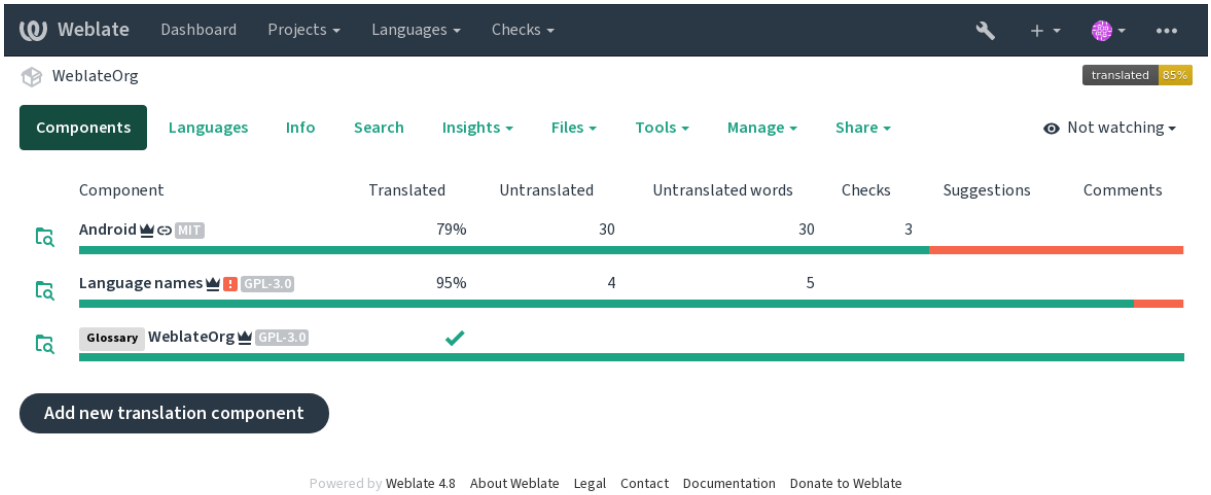
- Publicly visible and anybody can contribute
- Visible only to a certain group of translators

Vezi și:

Controlul accesului, *Translation workflows*

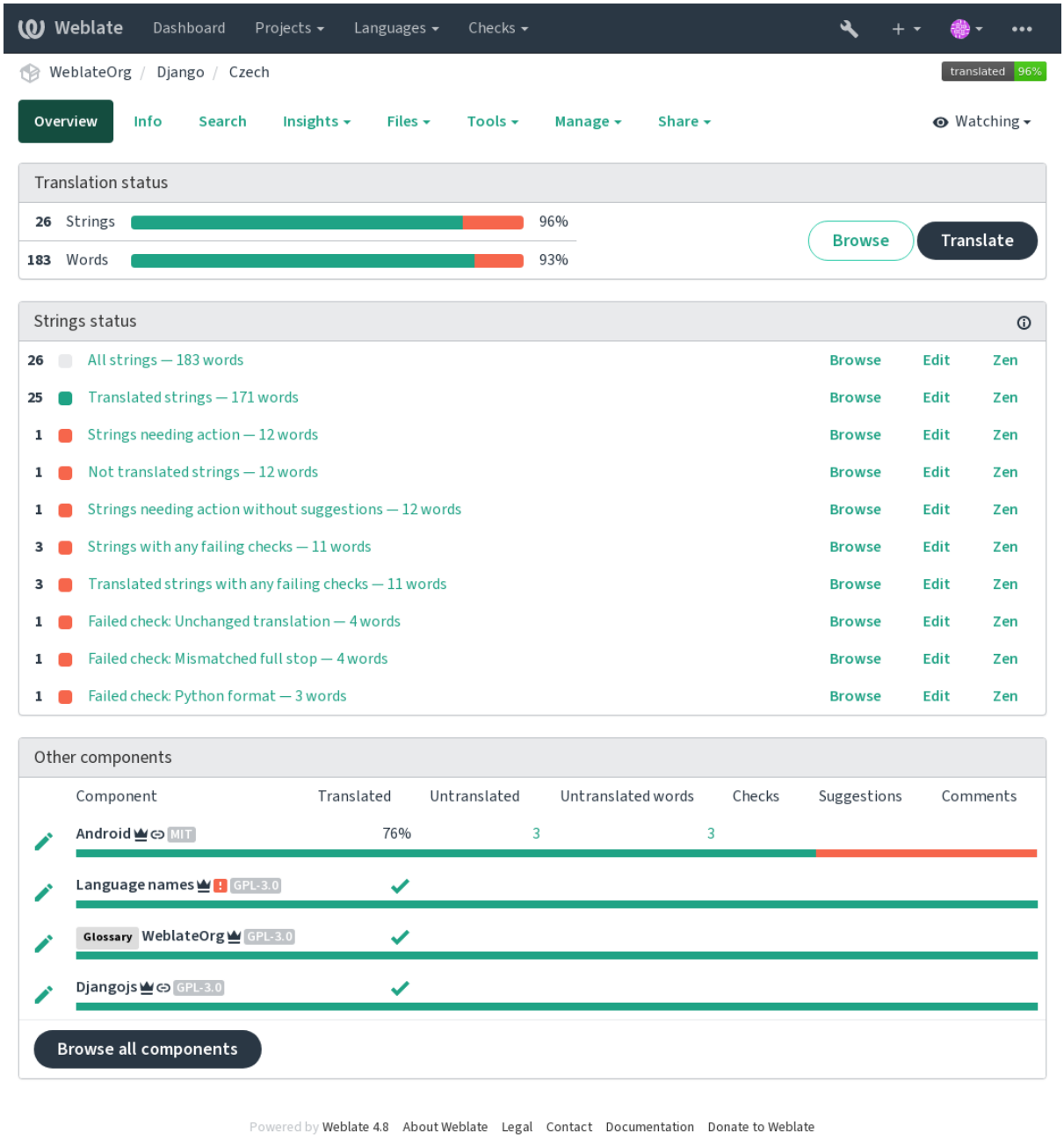
1.3.1 Proiecte de traducere

Translation projects hold related components; resources for the same software, book, or project.



1.3.2 Translation links

Having navigated to a component, a set of links lead to its actual translation. The translation is further divided into individual checks, like *Not translated strings* or *Strings needing action*. If the whole project is translated, without error, *All strings* is still available. Alternatively you can use the search field to find a specific string or term.



1.3.3 Sugestii

Notă: Actual permissions might vary depending on your Weblate configuration.

Anonymous users can only (by default) forward suggestions. Doing so is still available to signed-in users, in cases where uncertainty about the translation arises, prompting other translators to review it.

The suggestions are scanned on a daily basis to remove duplicates and suggestions matching the current translation.

1.3.4 Comentarii

Three types of comments can be posted: for translations, source strings, or to report source string bugs when this functionality is turned on using *Activați revizuirile sursei*. Choose the one suitable to topic you want to discuss. Source string comments are in any event good for providing feedback on the original string, for example that it should be rephrased or to ask questions about it.

You can use Markdown syntax in all comments and mention other users using `@mention`.

Vezi și:

report-source, *Source strings reviews*, *Activați revizuirile sursei*

1.3.5 Variante

Variants are used to group different length variants of the string. The frontend of your project can then use different strings depending on the screen or window size.

Vezi și:

variants, *Variante*

1.3.6 Etichete

Labels are used to categorize strings within a project to further customize the localization workflow (for example to define categories of strings).

Vezi și:

labels

1.3.7 Translating

On the translation page, the source string and an editing area for its translation are shown. Should the translation be plural, multiple source strings and editing areas are shown, each described and labeled in the amount of plural forms the translated language has.

All special whitespace characters are underlined in red and indicated with grey symbols. More than one subsequent space is also underlined in red to alert the translator to a potential formatting issue.

Various bits of extra info can be shown on this page, most of which coming from the project source code (like context, comments or where the message is being used). Translation fields for any secondary languages translators select in the preferences will be shown (see *Limbi secundare*) above the source string.

Below the translation, translators will find suggestion made by others, to be accepted (✓), accepted with changes (✏), or deleted (⍰).

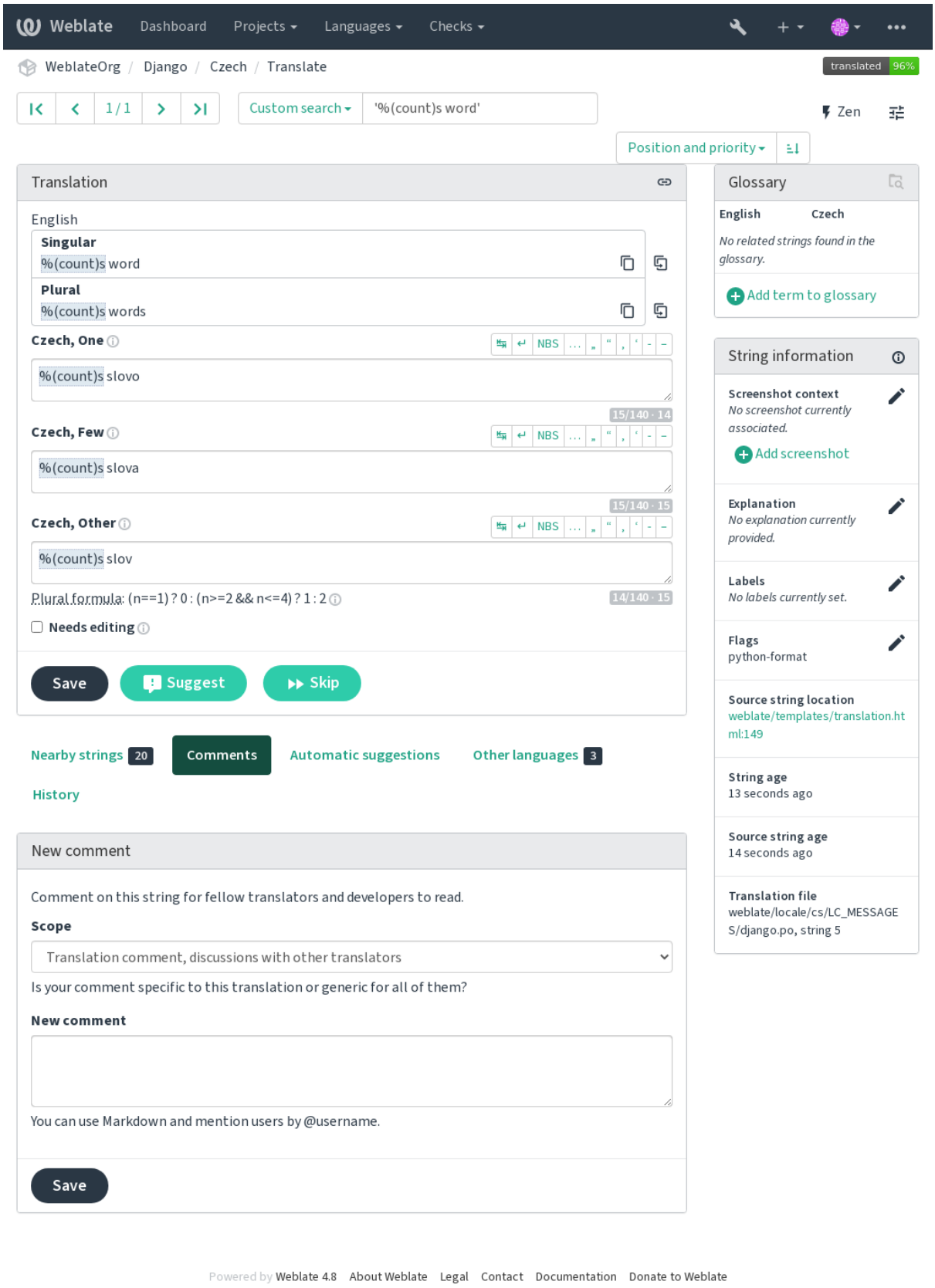
Pluralul

Words changing form to account of their numeric designation are called plurals. Each language has its own definition of plurals. English, for example, supports one. In the singular definition of for example „car”, implicitly one car is referenced, in the plural definition, „cars” two or more cars are referenced (or the concept of cars as a noun). Languages like for example Czech or Arabic have more plurals and also their rules for plurals are different.

Weblate has full support for each of these forms, in each respective language (by translating every plural separately). The number of fields and how it is in turn used in the translated application or project depends on the configured plural formula. Weblate shows the basic info, and the Language Plural Rules by the Unicode Consortium is a more detailed description.

Vezi şi:

Formula plurală



Keyboard shortcuts

Schimbat în versiunea 2.18: The keyboard shortcuts have been revamped in 2.18 to less likely collide with browser or system defaults.

The following keyboard shortcuts can be utilized during translation:

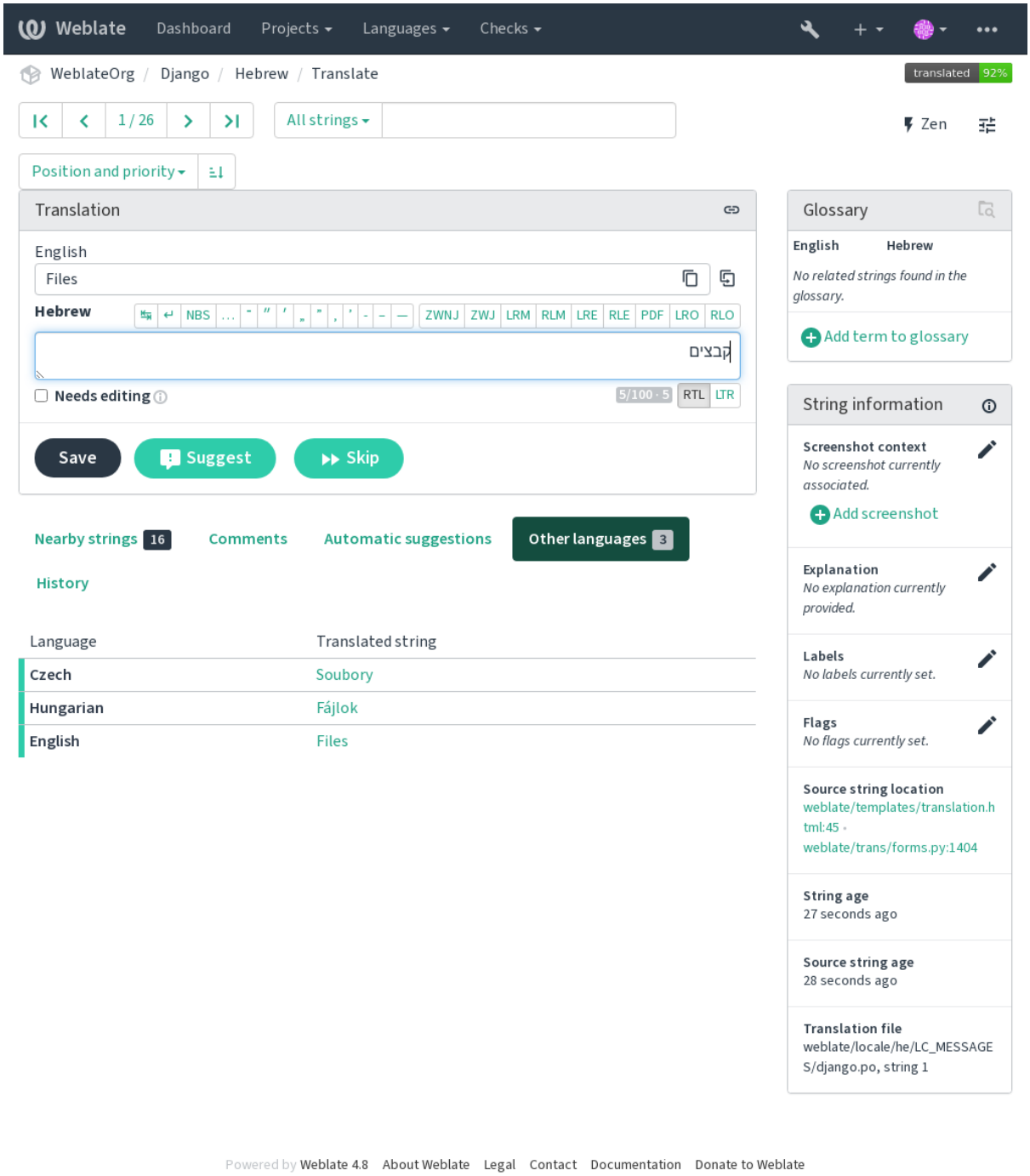
Keyboard shortcut	Descriere
`Alt+Home`	Navigate to first translation in current search.
`Alt+End`	Navigate to last translation in current search.
`Alt+PageUp` or `Ctrl ↑` or `Alt ↑` or `Cmd ↑`	Navigate to previous translation in current search.
`Alt+PageDown` or `Ctrl+↓` or `Alt+↓` or `Cmd+↓`	Navigate to next translation in current search.
`Alt+Enter` or `Ctrl+Enter` or `Cmd+Enter`	Save current translation.
`Ctrl+Shift+Enter` or `Cmd+Shift+Enter`	Unmark translation as needing edit and submit it.
`Ctrl+E` or `Cmd+E`	Focus translation editor.
`Ctrl+U` or `Cmd+U`	Focus comment editor.
`Ctrl+M` or `Cmd+M`	Shows *Automatic suggestions* tab, see *Sugestii automate*.
`Ctrl+1` to `Ctrl+9` or `Cmd+1` to `Cmd+9`	Copies placeable of given number from source string.
`Ctrl+M+1` to `9` or `Cmd+M+1` to `9`	Copy the machine translation of given number to current translation.
`Ctrl+I+1` to `9` or `Cmd+I+1` to `9`	Ignore one item in the list of failing checks.
`Ctrl+J` or `Cmd+J`	Shows the *Nearby strings* tab.
`Ctrl+S` or `Cmd+S`	Focus search field.
`Ctrl+O` or `Cmd+O`	Copy source string.
`Ctrl+Y` or `Cmd+Y`	Toggle the *Needs editing* flag.

Visual keyboard

A small visual keyboard row is shown just above the translation field. This can be useful to keep local punctuation in mind (as the row is local to each language), or have characters otherwise hard to type handy.

The shown symbols factor into three categories:

- User configured characters defined in the *Profilul utilizatorului*
- Per-language characters provided by Weblate (e.g. quotes or RTL specific characters)
- Characters configured using `SPECIAL_CHARS`



Translation context

This contextual description provides related info about the current string.

String attributes Things like message ID, context (`msgctxt`) or location in source code.

Capturi de ecran Screenshots can be uploaded to Weblate to better inform translators of where and how the string is used, see *Visual context for strings*.

Șiruri apropiate Displays neighbouring messages from the translation file. These are usually also used in a similar context and prove useful in keeping the translation consistent.

Alte apariții In case a message appears in multiple places (e.g. multiple components), this tab shows all of them if they are found to be inconsistent (see *Inconsecvent*). You can choose which one to use.

Memorie de traducere Look at similar strings translated in past, see *Memorie de traducere*.

Glosar Displays terms from the project glossary used in the current message.

Modificări recente List of people whom have changed this message recently using Weblate.

Proiect Project info like instructions for translators, or a directory or link to the string in the version control system repository the project uses.

If you want direct links, the translation format has to support it.

Translation history

Every change is by default (unless turned off in component settings) saved in the database, and can be reverted. Optionally one can still also revert anything in the underlying version control system.

Translated string length

Weblate can limit the length of a translation in several ways to ensure the translated string is not too long:

- The default limitation for translation is ten times longer than the source string. This can be turned off by `LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH`. In case you are hitting this, it might be also caused by a monolingual translation erroneously set up as bilingual one, making Weblate mistaking the translation key for the actual source string. See *Bilingual and monolingual formats* for more info.
- Maximal length in characters defined by translation file or flag, see *Lungimea maximă a traducerii*.
- Maximal rendered size in pixels defined by flags, see *Dimensiunea maximă a traducerii*.

1.3.8 Sugestii automate

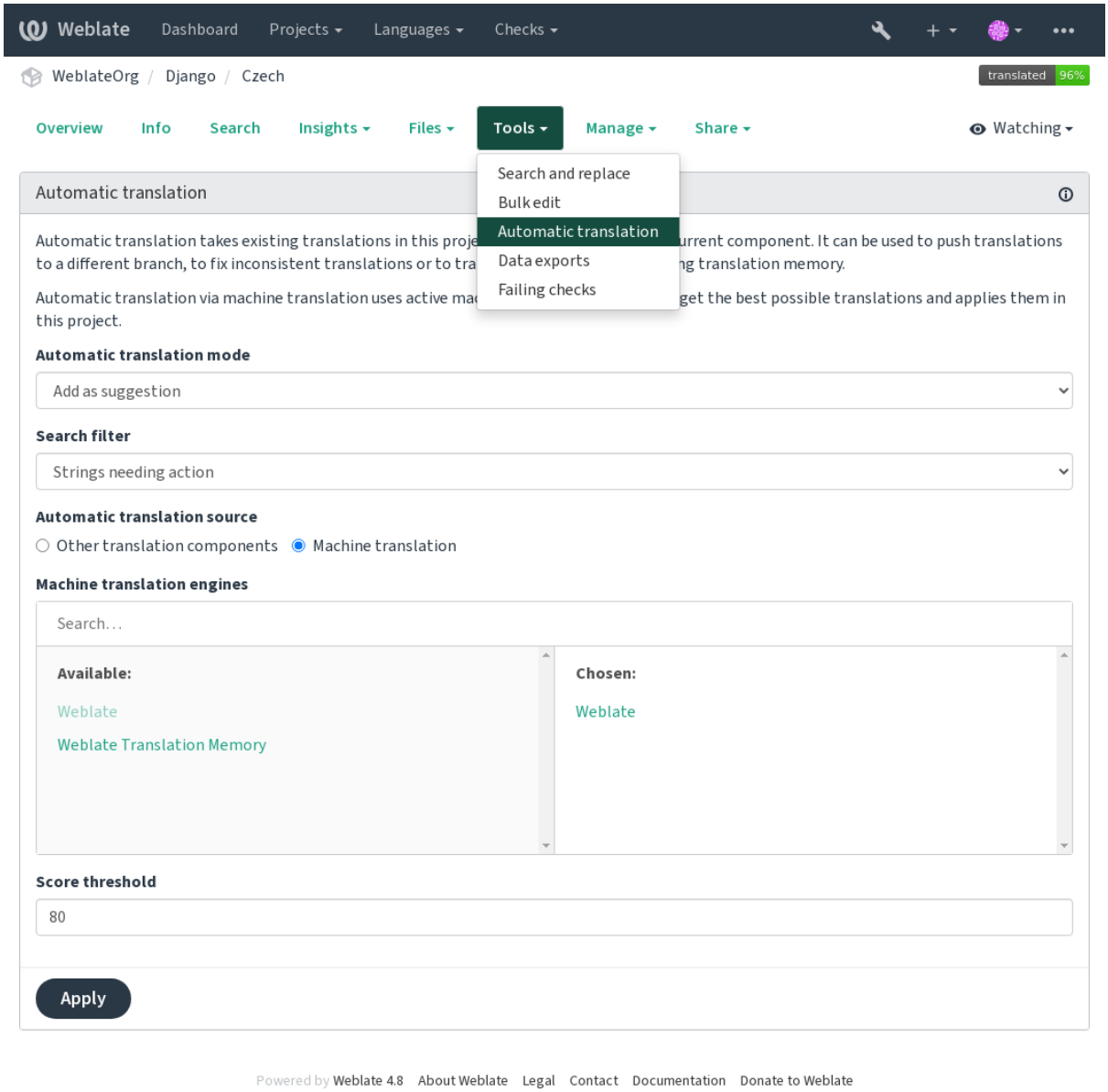
Based on configuration and your translated language, Weblate provides suggestions from several machine translation tools and *Memorie de traducere*. All machine translations are available in a single tab of each translation page.

Vezi și:

You can find the list of supported tools in *Traducere automată*.

1.3.9 Traducere automată

You can use automatic translation to bootstrap translation based on external sources. This tool is called *Automatic translation* accessible in the *Tools* menu, once you have selected a component and a language:



Two modes of operation are possible:

- Using other Weblate components as a source for translations.
- Using selected machine translation services with translations above a certain quality threshold.

You can also choose which strings are to be auto-translated.

> **Atenționare:** Be mindful that this will overwrite existing translations if employed with wide filters such as *All strings*.

Useful in several situations like consolidating translation between different components (for example the application and its website) or when bootstrapping a translation for a new component using existing translations (translation memory).

Vezi și:

Keeping translations same across components

1.3.10 Rate limiting

To avoid abuse of the interface, rate limiting is applied to several operations like searching, sending contact forms or translating. If affected by it, you are blocked for a certain period until you can perform the operation again.

Default limits and fine-tuning is described in the administrative manual, see *Rate limiting*.

1.3.11 Căutare și înlocuire

Change terminology effectively or perform bulk fixing of the strings using *Search and replace* in the *Tools* menu.

Sugestie: Don't worry about messing up the strings. This is a two-step process showing a preview of edited strings before the actual change is confirmed.

1.3.12 Editare în masă

Bulk editing allows performing one operation on number of strings. You define strings by searching for them and set up something to be done for matching ones. The following operations are supported:

- Changing string state (for example to approve all unreviewed strings).
- Adjust translation flags (see *Personalizarea comportamentului cu ajutorul steguleţelor*)
- Adjust string labels (see labels)

Sugestie: This tool is called *Bulk edit* accessible in the *Tools* menu of each project, component or translation.

Vezi și:

Bulk edit addon

1.4 Downloading and uploading translations

You can export files from a translation, make changes, and import them again. This allows working offline, and then merging changes back into the existing translation. This works even if it has been changed in the meantime.

Notă: Available options might be limited by *access control* settings.

1.4.1 Downloading translations

From the project or component dashboard, translatable files can be downloaded in the *Files* menu.

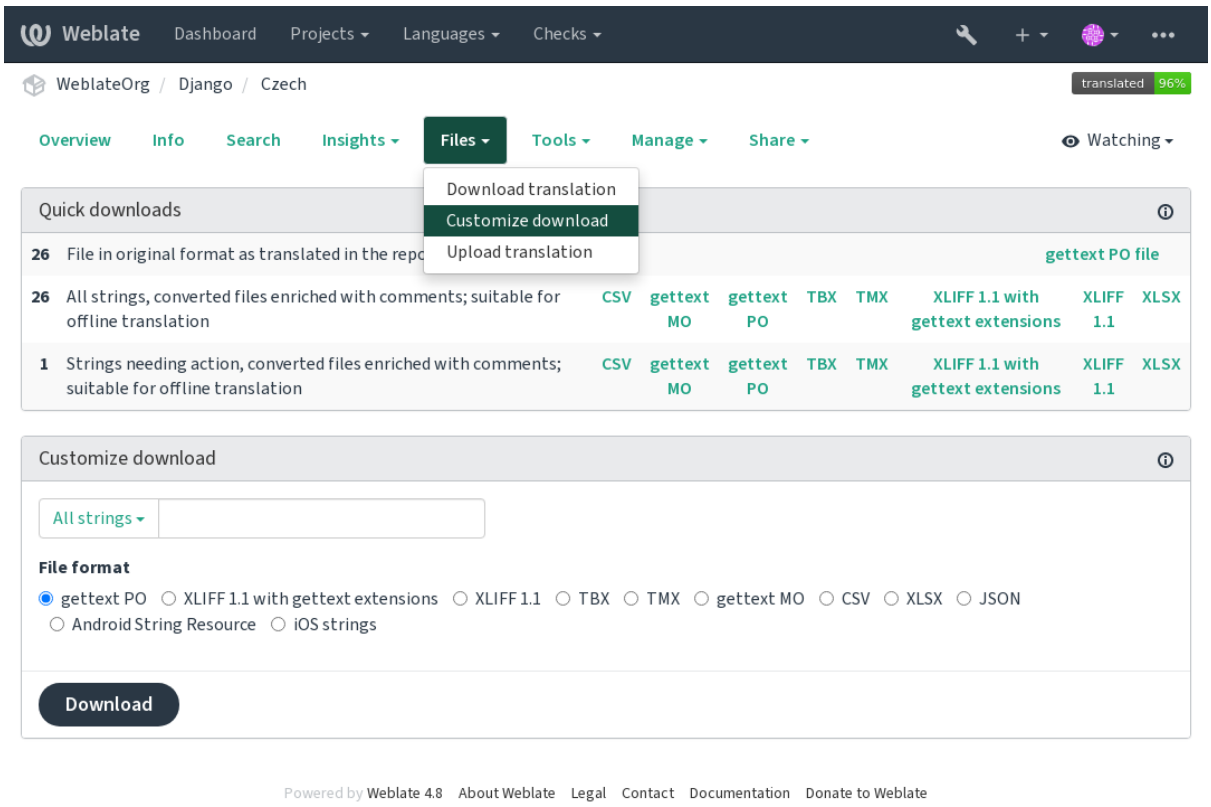
The first option is to download the file in the original format as it is stored in the repository. In this case, any pending changes in the translation are getting committed and the up-to-date file is yield without any conversions.

You can also download the translation converted into one of the widely used localization formats. The converted files will be enriched with data provided in Weblate; such as additional context, comments or flags. Several file formats are available via the *Files* ↓ *Customize download* menu:

- gettext PO
- XLIFF cu extensii gettext
- XLIFF 1.1

- TermBase eXchange
- Translation Memory eXchange
- gettext MO (only available when translation is using gettext PO)
- CSV
- Excel Deschidere XML
- JSON (only available for monolingual translations)
- Android String Resource (only available for monolingual translations)
- iOS strings (only available for monolingual translations)

Sugestie: The content available in the converted files differs based on file format features, you can find overview in *Translation types capabilities*.

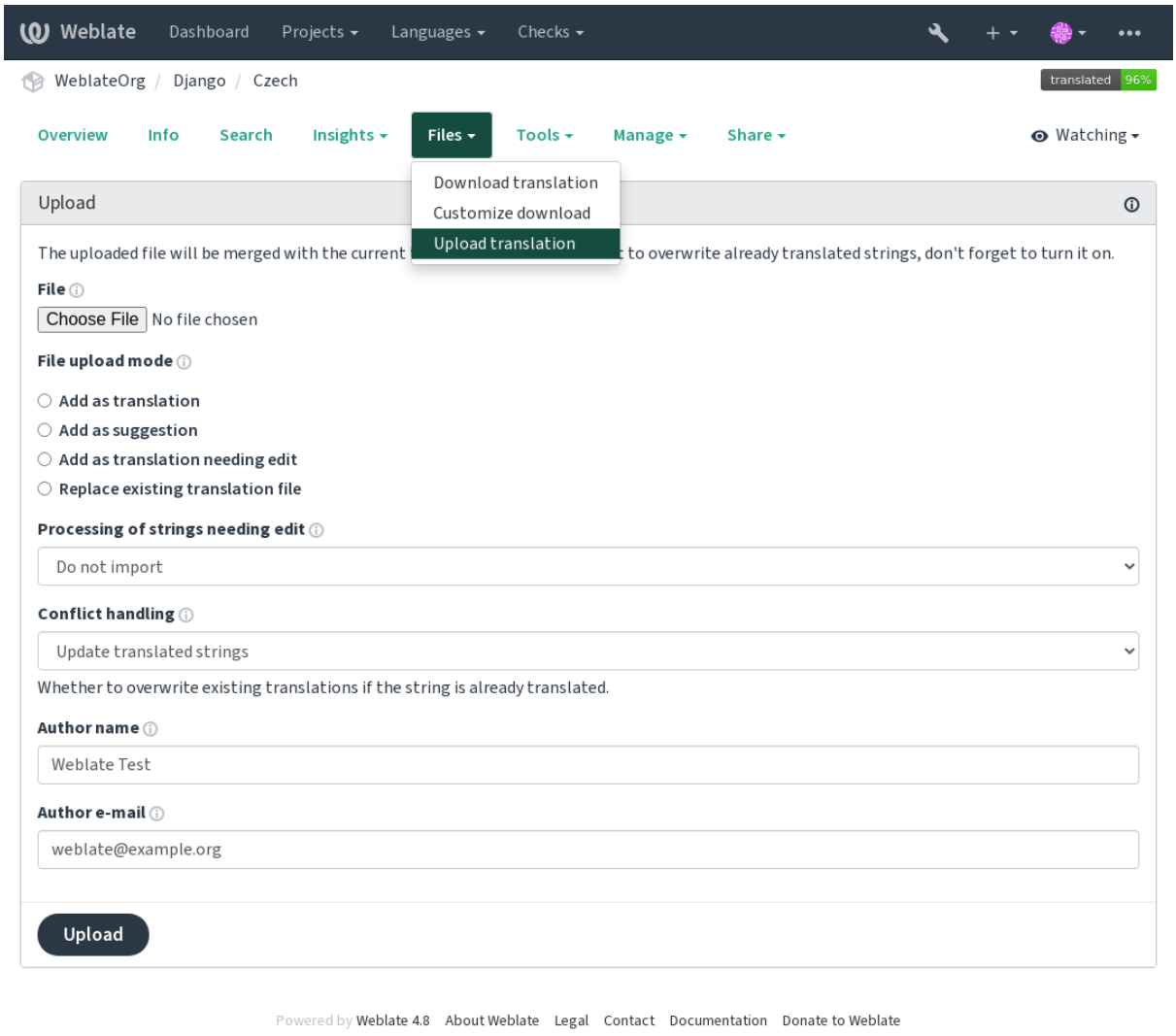


Vezi şi:

GET /api/translations/(string:project)/(string:component)/(string:language)/file/

1.4.2 Uploading translations

When you have made your changes, use *Upload translation* in the *Files* menu.



Supported file formats

Any file in a supported file format can be uploaded, but it is still recommended to use the same file format as the one used for translation, otherwise some features might not be translated properly.

Vezi și:

Supported file formats

The uploaded file is merged to update the translation, overwriting existing entries by default (this can be turned off or on in the upload dialog).

Import methods

These are the choices presented when uploading translation files:

Add as translation (`translate`) Imported translations are added as translations. This is the most common use-case, and the default behavior.

Only translations are used from the uploaded file and no additional content.

Add as suggestion (`suggest`) Imported translations are added as suggestions, do this when you want to have your uploaded strings reviewed.

Only translations are used from the uploaded file and no additional content.

Add as translation needing edit (`fuzzy`) Imported translations are added as translations needing edit. This can be useful when you want translations to be used, but also reviewed.

Only translations are used from the uploaded file and no additional content.

Replace existing translation file (`replace`) Existing file is replaced with new content. This can lead to loss of existing translations, use with caution.

Update source strings (`source`) Updates source strings in bilingual translation file. This is similar to what *Actualizarea fișierelor PO pentru a se potrivi cu POT (msgmerge)* does.

This option is supported only for some file formats.

Add new strings (`add`) Adds new strings to the translation. It skips the one which already exist.

In case you want to both add new strings and update existing translations, upload the file second time with *Add as translation*.

This option is available only with *Gestionați șirurile de caractere* turned on.

Only source, translation and key (context) are used from the uploaded file.

Vezi și:

`POST /api/translations/(string:project)/(string:component)/(string:language)/file/`

Conflicts handling

Defines how to deal with uploaded strings which are already translated.

Strings needing edit

There is also an option for how to handle strings needing edit in the imported file. Such strings can be handle in one of the three following ways: „Do not import”, „Import as string needing edit”, or „Import as translated”.

Overriding authorship

With admin permissions, you can also specify authorship of uploaded file. This can be useful in case you've received the file in another way and want to merge it into existing translations while properly crediting the actual author.

1.5 Glosar

Each project can include one or more glossaries as a shorthand for storing terminology. Glossary easify maintaining consistency of the translation.

A glossary for each language can be managed on its own, but they are stored together as a single component which helps project admins and multilingual translators to maintain some cross-language consistency as well. Terms from the glossary containing words from the currently translated string are displayed in the sidebar of the translation editor.

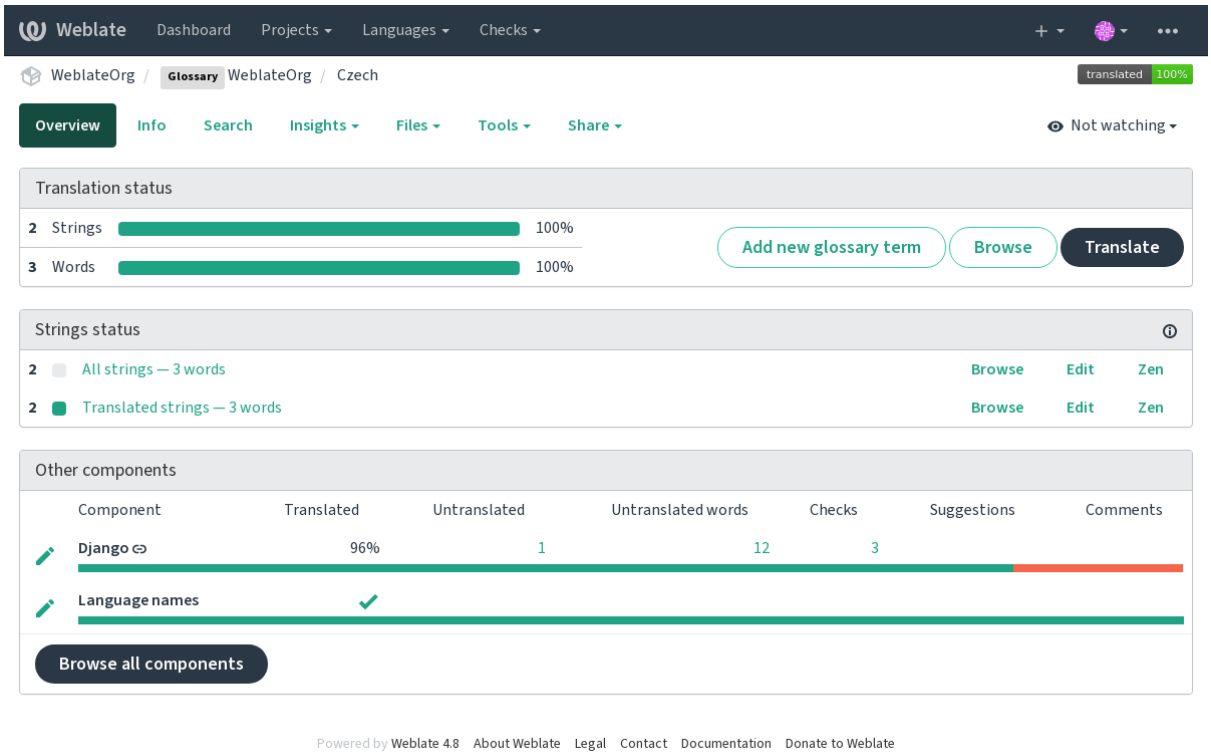
1.5.1 Managing glossaries

Schimbat în versiunea 4.5: Glossaries are now regular translation components and you can use all Weblate features on them — commenting, storing in a remote repository, or adding explanations.

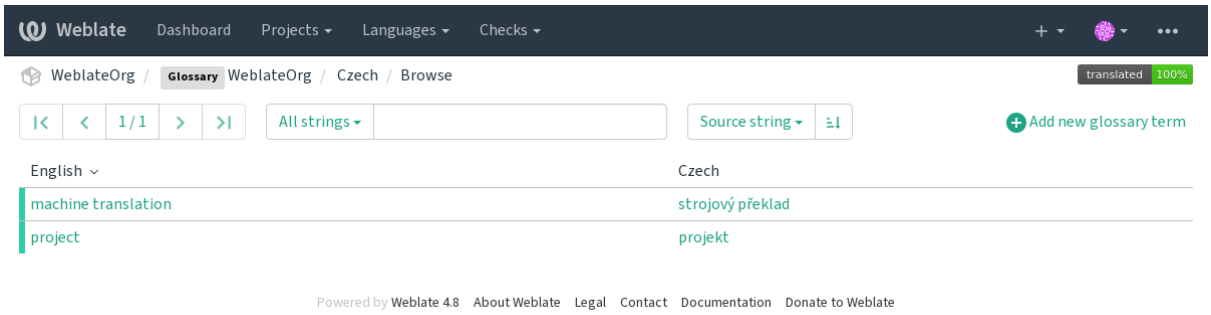
Use any component as a glossary by turning on *Utilizarea ca glosar*. You can create multiple glossaries for one project.

An empty glossary for a given project is automatically created with the project. Glossaries are shared among all components of the same project, and optionally with other projects using *Cota în proiecte* from the respective glossary component.

The glossary component looks like any other component in Weblate with added colored label:



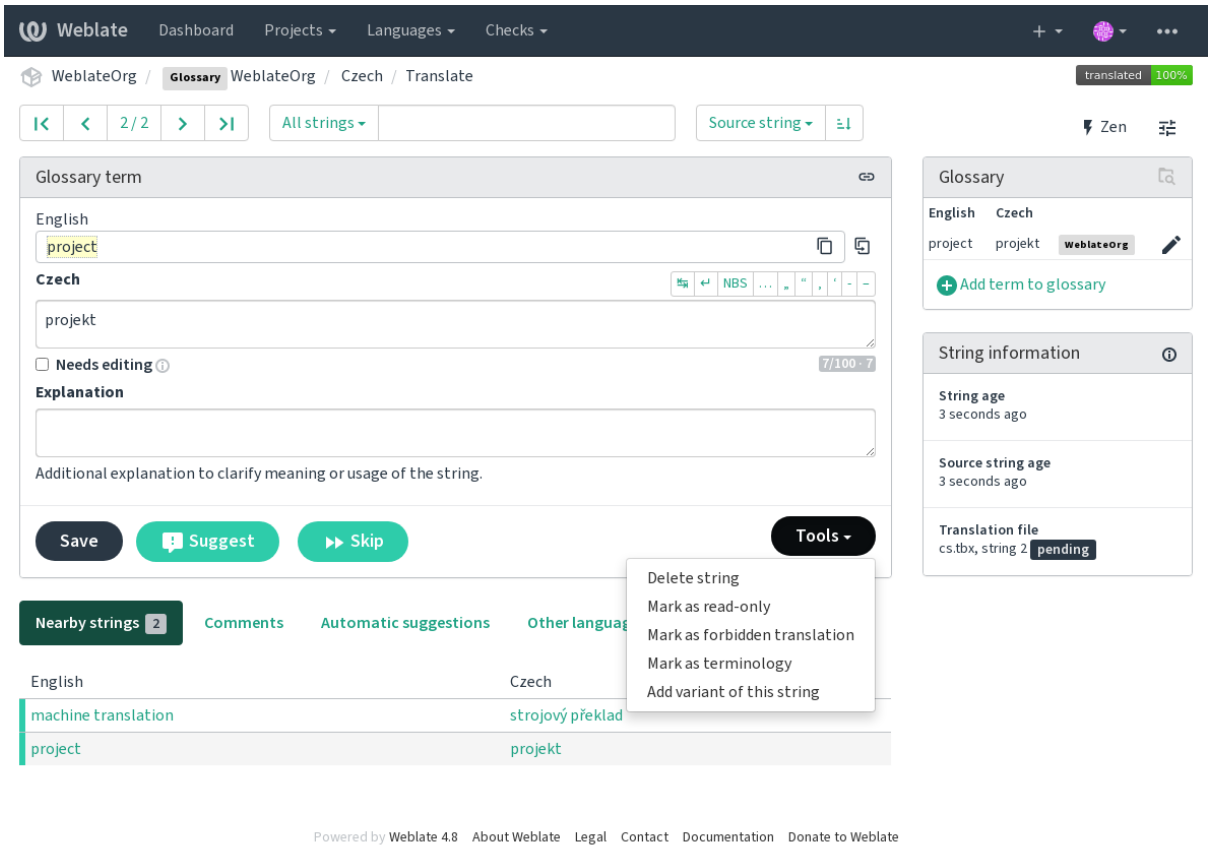
You can browse all glossary terms:



or edit them as any translations.

1.5.2 Glossary terms

Glossary terms are translated the same way regular strings are. You can toggle additional features using the *Tools* menu for each term.



Untranslatable terms

Nou în versiunea 4.5.

Flagging certain glossary term translations `read-only` by bulk-editing, typing in the flag, or by using *Tools* ↓ *Mark as read-only* means they can not be translated. Use this for brand names or other terms that should not be changed in other languages. Such terms are visually highlighted in the glossary sidebar.

Vezi și:

Personalizarea comportamentului cu ajutorul steguleţelor

Forbidden translations

Nou în versiunea 4.5.

Flagging certain glossary term translations as `forbidden`, by bulk-editing, typing in the flag, or by using *Tools* ↓ *Mark as forbidden translation* means they are **not** to be used. Use this to clarify translation when some words are ambiguous or could have unexpected meanings.

Vezi și:

Personalizarea comportamentului cu ajutorul steguleţelor

Terminologie

Nou în versiunea 4.5.

Flagging certain glossary terms as `terminology` by bulk-editing, typing in the flag, or by using *Tools* ↓ *Mark as terminology* adds entries for them to all languages in the glossary. Use this for important terms that should be well thought out, and retain a consistent meaning across all languages.

Vezi și:

Personalizarea comportamentului cu ajutorul steguleţelor

Variante

Variants are a generic way to group strings together. All term variants are listed in the glossary sidebar when translating.

Sugestie: You can use this to add abbreviations or shorter expressions for a term.

Vezi și:

variants

1.6 Verificări și reparații

The quality checks help catch common translator errors, ensuring the translation is in good shape. The checks can be ignored in case of false positives.

Once submitting a translation with a failing check, this is immediately shown to the user:

Weblate Dashboard Projects Languages Checks

WeblateOrg / Django / Czech / Translate translated 96%

The translation has been saved, however there are some newly failing checks: Missing plurals, Python format

1 / 1 Custom search '%(count)s word' Position Zen

Translation

English

Singular
%(count)s word

Plural
%(count)s words

Czech, One

0/140 · 14

Czech, Few

několik slov

12/140 · 15

Czech, Other

%(count)s slov

Plural formula: (n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2

14/140 · 15

Needs editing

Save Suggest Skip

Nearby strings 20 Comments Automatic suggestions Other languages 3 History

New comment

Comment on this string for fellow translators and developers to read.

Scope

Translation comment, discussions with other translators

Is your comment specific to this translation or generic for all of them?

New comment

You can use Markdown and mention users by @username.

Save

Things to check

Python format 1

Following format strings are missing: %(count)s

Dismiss

For all languages

Missing plurals 2

Some plural forms are not translated

Dismiss

For all languages

Glossary

English Czech

No related strings found in the glossary.

Add term to glossary

String information

Screenshot context
No screenshot currently associated.

Add screenshot

Explanation
No explanation currently provided.

Labels
No labels currently set.

Flags
python-format

Source string location
weblate/templates/translation.html:149

String age
20 seconds ago

Source string age
21 seconds ago

Translation file
weblate/locale/cs/LC_MESSAGES/django.po, string 5
pending

Powered by Weblate 4.8 About Weblate Legal Contact Documentation Donate to Weblate

1.6.1 Automatic fixups

In addition to *Quality checks*, Weblate can fix some common errors in translated strings automatically. Use it with caution to not have it add errors.

Vezi și:

`AUTOFIX_LIST`

1.6.2 Quality checks

Weblate employs a wide range of quality checks on strings. The following section describes them all in further detail. There are also language specific checks. Please file a bug if anything is reported in error.

Vezi și:

`CHECK_LIST`, *Personalizarea comportamentului cu ajutorul steguleţelor*

1.6.3 Translation checks

Executed upon every translation change, helping translators maintain good quality translations.

Marcaj BBcode

Summary BBcode în traducere nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.markup.BBCodeCheck`

Flag to ignore `ignore-bbcode`

BBCode represents simple markup, like for example highlighting important parts of a message in bold font, or italics.

This check ensures they are also found in translation.

Notă: The method for detecting BBcode is currently quite simple so this check might produce false positives.

Cuvinte consecutive duplicate

Nou în versiunea 4.1.

Summary Textul conține același cuvânt de două ori la rând:

Domeniul de aplicare translated strings

Check class `weblate.checks.duplicate.DuplicateCheck`

Flag to ignore `ignore-duplicate`

Checks that no consecutive duplicate words occur in a translation. This usually indicates a mistake in the translation.

Sugestie: This check includes language specific rules to avoid false positives. In case it triggers falsely in your case, let us know. See *Reporting issues in Weblate*.

Nu respectă glosarul

Nou în versiunea 4.5.

Summary Traducerea nu respectă termenii definiți într-un glosar.

Domeniul de aplicare translated strings

Check class `weblate.checks.glossary.GlossaryCheck`

Flag to enable `check-glossary`

Flag to ignore `ignore-check-glossary`

This check has to be turned on using `check-glossary` flag (see *Personalizarea comportamentului cu ajutorul steguleţelor*). Please consider following prior to enabling it:

- It does exact string matching, the glossary is expected to contain terms in all variants.
- Checking each string against glossary is expensive, it will slow down any operation in Weblate which involves running checks like importing strings or translating.

Vezi și:

Glosar, *Personalizarea comportamentului cu ajutorul steguleţelor*, *Steaguri de traducere*

Spațiu dublu

Summary Traducerea conține spațiu dublu

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.DoubleSpaceCheck`

Flag to ignore `ignore-double-space`

Checks that double space is present in translation to avoid false positives on other space-related checks.

Check is false when double space is found in source meaning double space is intentional.

Formatted strings

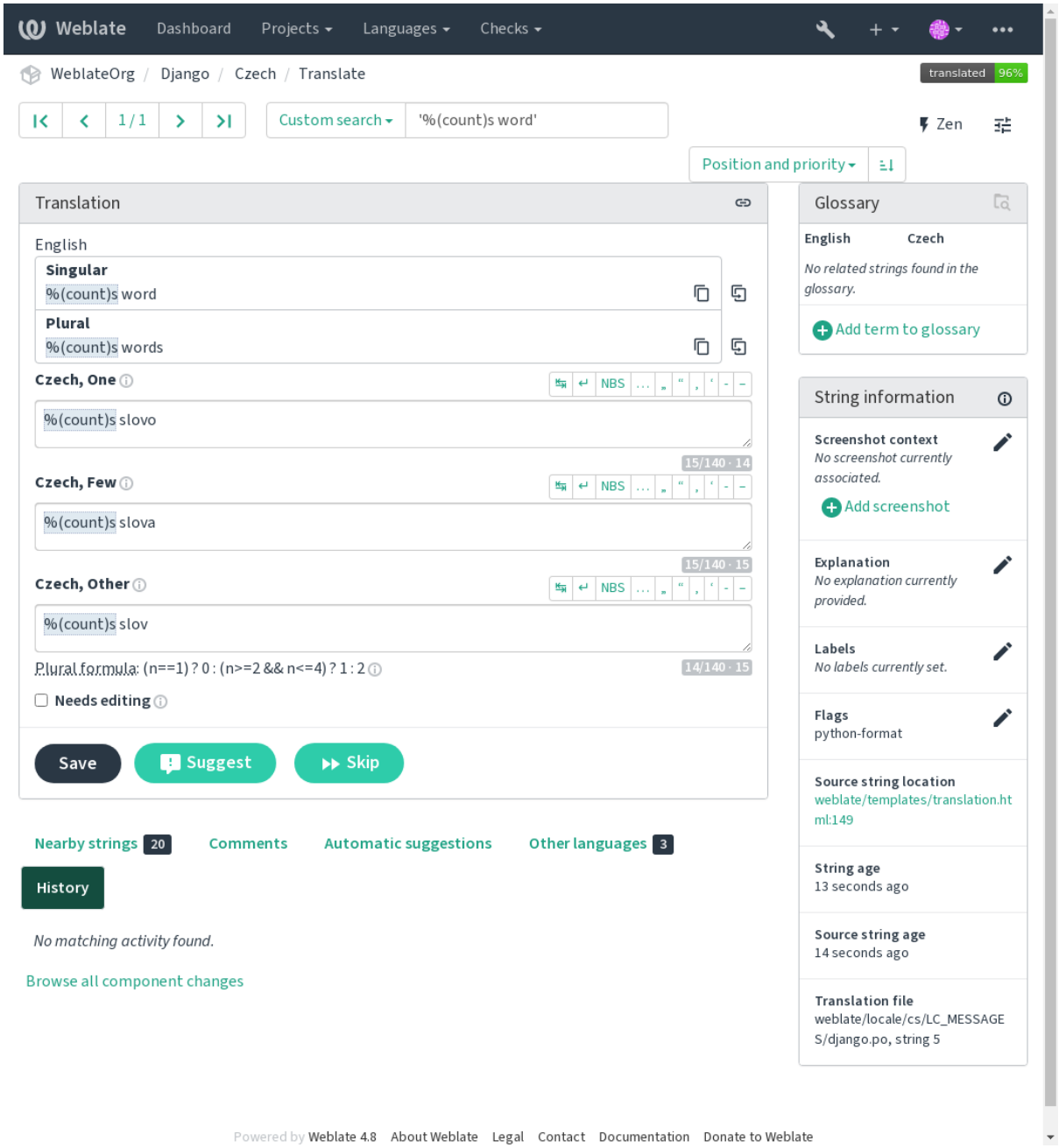
Checks that formatting in strings are replicated between both source and translation. Omitting format strings in translation usually causes severe problems, so the formatting in strings should usually match the source.

Weblate supports checking format strings in several languages. The check is not enabled automatically, only if a string is flagged appropriately (e.g. *c-format* for C format). Gettext adds this automatically, but you will probably have to add it manually for other file formats or if your PO files are not generated by **`xgettext`**.

This can be done per unit (see *Additional info on source strings*) or in *Component configuration*. Having it defined per component is simpler, but can lead to false positives in case the string is not interpreted as a formatting string, but format string syntax happens to be used.

Sugestie: In case specific format check is not available in Weblate, you can use generic *Semne de poziție*.

Besides checking, this will also highlight the formatting strings to easily insert them into translated strings:



Șir de interpolare AngularJS

Summary Șirurile de interpolare AngularJS nu se potrivesc cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.angularjs.AngularJSInterpolationCheck`

Flag to enable `angularjs-format`

Flag to ignore `ignore-angularjs-format`

Named format string example `Your balance is {{amount}} {{ currency }}`

Vezi și:

Formatted strings, AngularJS text interpolation

Format C

Summary Șirul de format C nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.CFormatCheck`

Flag to enable `c-format`

Flag to ignore `ignore-c-format`

Simple format string example `There are %d apples`

Position format string example `Your balance is %1$d %2$s`

Vezi și:

Formatted strings, C format strings, C printf format

C# format

Summary Șirul de format C# nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.CSharpFormatCheck`

Flag to enable `c-sharp-format`

Flag to ignore `ignore-c-sharp-format`

Position format string example `There are {0} apples`

Vezi și:

Formatted strings, C# String Format

Șabloane literale ECMAScript

Summary Literalele șablonului ECMAScript nu se potrivesc cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.ESTemplateLiteralsCheck`

Flag to enable `es-format`

Flag to ignore `ignore-es-format`

Interpolation example `There are ${number} apples`

Vezi și:

Formatted strings, Template literals

Interpolare i18next

Nou în versiunea 4.0.

Summary Interpolarea i18next nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.I18NextInterpolationCheck`

Flag to enable `i18next-interpolation`

Flag to ignore `ignore-i18next-interpolation`

Interpolation example `There are {{number}} apples`

Nesting example `There are $t(number) apples`

Vezi și:

Formatted strings, i18next interpolation

format Java

Summary Șirul de format Java nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.JavaFormatCheck`

Flag to enable `java-format`

Flag to ignore `ignore-java-format`

Simple format string example `There are %d apples`

Position format string example `Your balance is %1$d %2$s`

Vezi și:

Formatted strings, Java Format Strings

Mesaj în format Java

Summary Șirul Java MessageFormat nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.JavaMessageFormatCheck`

Flag to enable uncodintionally `java-messageformat`

Flag to enable autodetection `auto-java-messageformat` enables check only if there is a format string in the source

Flag to ignore `ignore-java-messageformat`

Position format string example `There are {0} apples`

Vezi și:

Formatted strings, Java MessageFormat

Format JavaScript

Summary Șirul de format JavaScript nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.JavaScriptFormatCheck`

Flag to enable `javascript-format`

Flag to ignore `ignore-javascript-format`

Simple format string example `There are %d apples`

Vezi și:

Formatted strings, JavaScript formatting strings

Formatul Lua

Summary Șirul de format Lua nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.LuaFormatCheck`

Flag to enable `lua-format`

Flag to ignore `ignore-lua-format`

Simple format string example `There are %d apples`

Vezi și:

Formatted strings, Lua formatting strings

Object Pascal format

Summary Object Pascal format string does not match source

Domeniul de aplicare translated strings

Check class `weblate.checks.format.ObjectPascalFormatCheck`

Flag to enable `object-pascal-format`

Flag to ignore `ignore-object-pascal-format`

Simple format string example `There are %d apples`

Vezi și:

Formatted strings, Object Pascal formatting strings Free Pascal formatting strings Delphi formatting strings

Procentaje de marcaj

Nou în versiunea 4.0.

Summary Procentajele nu se potrivesc cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.PercentPlaceholdersCheck`

Flag to enable `percent-placeholders`

Flag to ignore `ignore-percent-placeholders`

Simple format string example `There are %number% apples`

Vezi și:

Formatted strings,

Formatul Perl

Summary Șirul de format Perl nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.PerlFormatCheck`

Flag to enable `perl-format`

Flag to ignore `ignore-perl-format`

Simple format string example `There are %d apples`

Position format string example `Your balance is %1$d %2$s`

Vezi și:

Formatted strings, Perl sprintf, Perl Format Strings

Formatul PHP

Summary Șirul de format PHP nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.PHPFormatCheck`

Flag to enable `php-format`

Flag to ignore `ignore-php-format`

Simple format string example `There are %d apples`

Position format string example `Your balance is %1$d %2$s`

Vezi și:

Formatted strings, PHP sprintf documentation, PHP Format Strings

Formatul de bretea Python

Summary Șirul de format Python brace nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.PythonBraceFormatCheck`

Flag to enable `python-brace-format`

Flag to ignore `ignore-python-brace-format`

Simple format string `There are {} apples`

Named format string example `Your balance is {amount} {currency}`

Vezi și:

Formatted strings, Python brace format, Python Format Strings

Format Python

Summary Șirul de format Python nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.PythonFormatCheck`

Flag to enable `python-format`

Flag to ignore `ignore-python-format`

Simple format string `There are %d apples`

Named format string example `Your balance is %(amount)d %(currency)s`

Vezi și:

Formatted strings, Python string formatting, Python Format Strings

Formatul Qt

Summary Șirul de format Qt nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.qt.QtFormatCheck`

Flag to enable `qt-format`

Flag to ignore `ignore-qt-format`

Position format string example `There are %1 apples`

Vezi și:

Formatted strings, Qt QString::arg()

Formatul plural Qt

Summary Șirul de formate Qt plural nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.qt.QtPluralCheck`

Flag to enable `qt-plural-format`

Flag to ignore `ignore-qt-plural-format`

Plural format string example `There are %Ln apple(s)`

Vezi și:

Formatted strings, Qt i18n guide

Formatul Ruby

Summary Șirul de format Ruby nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.ruby.RubyFormatCheck`

Flag to enable `ruby-format`

Flag to ignore `ignore-ruby-format`

Simple format string example `There are %d apples`

Position format string example `Your balance is %1$f %2$s`

Named format string example `Your balance is %+.2<amount>f %<currency>s`

Named template string `Your balance is %{amount} %{currency}`

Vezi și:

Formatted strings, Ruby Kernel#sprintf

Formatul schemei

Summary Șirul de format al schemei nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.SchemeFormatCheck`

Flag to enable `scheme-format`

Flag to ignore `ignore-scheme-format`

Simple format string example `There are ~d apples`

Vezi și:

Formatted strings, Srfi 28, Chicken Scheme format, Guile Scheme formatted output

Formatare Vue I18n

Summary Formatarea Vue I18n nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.format.VueFormattingCheck`

Flag to enable `vue-format`

Flag to ignore `ignore-vue-format`

Named formatting `There are {count} apples`

Rails i18n formatting `There are %{count} apples`

Linked locale messages `@:message.dio @:message.the_world!`

Vezi și:

Formatted strings, Vue I18n Formatting, Vue I18n Linked locale messages

A fost tradus

Summary Acest șir a fost tradus în trecut

Domeniul de aplicare toate șirurile de caractere

Check class `weblate.checks.consistency.TranslatedCheck`

Flag to ignore `ignore-translated`

Means a string has been translated already. This can happen when the translations have been reverted in VCS or lost otherwise.

Inconsecvent

Summary Acest șir are mai multe traduceri în acest proiect sau nu este tradus în anumite componente.

Domeniul de aplicare toate șirurile de caractere

Check class `weblate.checks.consistency.ConsistencyCheck`

Flag to ignore `ignore-inconsistent`

Weblate checks translations of the same string across all translation within a project to help you keep consistent translations.

The check fails on differing translations of one string within a project. This can also lead to inconsistencies in displayed checks. You can find other translations of this string on the *Other occurrences* tab.

Sugestie: For performance reasons, the check might not find all inconsistencies, it limits number of matches.

Notă: This check also fires in case the string is translated in one component and not in another. It can be used as a quick way to manually handle strings which are not translated in some components just by clicking on the *Use this translation* button displayed on each line in the *Other occurrences* tab.

You can use *Traducere automată* addon to automate translating of newly added strings which are already translated in another component.

Vezi și:

Keeping translations same across components

Scrisoarea Kashida folosită

Nou în versiunea 3.5.

Summary Literele decorative kashida nu trebuie să fie folosite

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.KashidaCheck`

Flag to ignore `ignore-kashida`

The decorative Kashida letters should not be used in translation. These are also known as Tatweel.

Vezi și:

Kashida on Wikipedia

Legături Markdown

Nou în versiunea 3.5.

Summary Legăturile Markdown nu se potrivesc cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.markup.MarkdownLinkCheck`

Flag to enable `md-text`

Flag to ignore `ignore-md-link`

Markdown links do not match source.

Vezi și:

Markdown links

Referințe Markdown

Nou în versiunea 3.5.

Summary Referințele de legătură Markdown nu se potrivesc cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.markup.MarkdownRefLinkCheck`

Flag to enable `md-text`

Flag to ignore `ignore-md-reflink`

Markdown link references do not match source.

Vezi și:

Markdown links

Sintaxa Markdown

Nou în versiunea 3.5.

Summary Sintaxa Markdown nu se potrivește cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.markup.MarkdownSyntaxCheck`

Flag to enable `md-text`

Flag to ignore `ignore-md-syntax`

Sintaxa Markdown nu se potrivește cu sursa

Vezi și:

Markdown span elements

Lungimea maximă a traducerii

Summary Traducerea nu trebuie să depășească lungimea dată

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.MaxLengthCheck`

Flag to enable `max-length`

Flag to ignore `ignore-max-length`

Checks that translations are of acceptable length to fit available space. This only checks for the length of translation characters.

Unlike the other checks, the flag should be set as a `key:value` pair like `max-length:100`.

Sugestie: This check looks at number of chars, what might not be the best metric when using proportional fonts to render the text. The *Dimensiunea maximă a traducerii* check does check actual rendering of the text.

The `replacements:` flag might be also useful to expand placeables before checking the string.

Dimensiunea maximă a traducerii

Summary Textul redat în traducere nu trebuie să depășească dimensiunea dată

Domeniul de aplicare translated strings

Check class `weblate.checks.render.MaxSizeCheck`

Flag to enable `max-size`

Flag to ignore `ignore-max-size`

Nou în versiunea 3.7.

Translation rendered text should not exceed given size. It renders the text with line wrapping and checks if it fits into given boundaries.

This check needs one or two parameters - maximal width and maximal number of lines. In case the number of lines is not provided, one line text is considered.

You can also configure used font by `font-*` directives (see *Personalizarea comportamentului cu ajutorul steguleților*), for example following translation flags say that the text rendered with ubuntu font size 22 should fit into two lines and 500 pixels:

```
max-size:500:2, font-family:ubuntu, font-size:22
```

Sugestie: You might want to set `font-*` directives in *Component configuration* to have the same font configured for all strings within a component. You can override those values per string in case you need to customize it per string.

The `replacements:` flag might be also useful to expand placeables before checking the string.

Vezi și:

Gestionarea fonturilor, *Personalizarea comportamentului cu ajutorul steguleților*, *Lungimea maximă a traducerii*

Mismatched \n

Summary Number of \n in translation does not match source

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.EscapedNewlineCountingCheck`

Flag to ignore `ignore-escaped-newline`

Usually escaped newlines are important for formatting program output. Check fails if the number of `\n` literals in translation do not match the source.

Două puncte nepotrivite

Summary Sursa și traducerea nu se termină amândouă cu două puncte

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.EndColonCheck`

Flag to ignore `ignore-end-colon`

Checks that colons are replicated between both source and translation. The presence of colons is also checked for various languages where they do not belong (Chinese or Japanese).

Vezi și:

Colon on Wikipedia

Elipsa nepotrivită

Summary Sursa și traducerea nu se termină amândouă cu o elipsă

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.EndEllipsisCheck`

Flag to ignore `ignore-end-ellipsis`

Checks that trailing ellipses are replicated between both source and translation. This only checks for real ellipsis (…) not for three dots (`...`).

An ellipsis is usually rendered nicer than three dots in print, and sounds better with text-to-speech.

Vezi și:

Ellipsis on Wikipedia

Semnul exclamării nepotrivit

Summary Sursa și traducerea nu se termină amândouă cu semnul exclamării

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.EndExclamationCheck`

Flag to ignore `ignore-end-exclamation`

Checks that exclamations are replicated between both source and translation. The presence of exclamation marks is also checked for various languages where they do not belong (Chinese, Japanese, Korean, Armenian, Limbu, Myanmar or Nko).

Vezi și:

Exclamation mark on Wikipedia

Punct final nepotrivit

Summary Sursa și traducerea nu se termină amândouă cu punct și virgulă

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.EndStopCheck`

Flag to ignore `ignore-end-stop`

Checks that full stops are replicated between both source and translation. The presence of full stops is checked for various languages where they do not belong (Chinese, Japanese, Devanagari or Urdu).

Vezi și:

Full stop on Wikipedia

Semn de întrebare nepotrivit

Summary Sursa și traducerea nu se termină amândouă cu un semn de întrebare

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.EndQuestionCheck`

Flag to ignore `ignore-end-question`

Checks that question marks are replicated between both source and translation. The presence of question marks is also checked for various languages where they do not belong (Armenian, Arabic, Chinese, Korean, Japanese, Ethiopic, Vai or Coptic).

Vezi și:

Question mark on Wikipedia

Punct și virgulă nepotrivit

Summary Sursa și traducerea nu se termină amândouă cu punct și virgulă

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.EndSemicolonCheck`

Flag to ignore `ignore-end-semicolon`

Checks that semicolons at the end of sentences are replicated between both source and translation. This can be useful to keep formatting of entries such as desktop files.

Vezi și:

Semicolon on Wikipedia

Întreruperi de linie nepotrivite

Summary Numărul de linii noi din traducere nu corespunde sursei

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.NewLineCountCheck`

Flag to ignore `ignore-newline-count`

Usually newlines are important for formatting program output. Check fails if the number of `\n` literals in translation do not match the source.

Lipsește pluralul

Summary Unele forme de plural nu sunt traduse

Domeniul de aplicare translated strings

Check class `weblate.checks.consistency.PluralsCheck`

Flag to ignore `ignore-plurals`

Checks that all plural forms of a source string have been translated. Specifics on how each plural form is used can be found in the string definition.

Failing to fill in plural forms will in some cases lead to displaying nothing when the plural form is in use.

Semne de poziție

Nou în versiunea 3.9.

Summary Traducerea lipsește unele placeholders

Domeniul de aplicare translated strings

Check class `weblate.checks.placeholders.PlaceholderCheck`

Flag to enable `placeholders`

Flag to ignore `ignore-placeholders`

Schimbat în versiunea 4.3: You can use regular expression as placeholder.

Translation is missing some placeholders. These are either extracted from the translation file or defined manually using `placeholders` flag, more can be separated with colon, strings with space can be quoted:

```
placeholders:$URL$:$TARGET$:"some long text"
```

In case you have some syntax for placeholders, you can use a regular expression:

```
placeholders:r"%[^% ]%"
```

Vezi și:

Personalizarea comportamentului cu ajutorul steguleţelor

Spațierea punctuației

Nou în versiunea 3.9.

Summary Lipsesc spațiile neîntreruptibile înainte de semnul dublu de punctuație

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.PunctuationSpacingCheck`

Flag to ignore `ignore-punctuation-spacing`

Checks that there is non breakable space before double punctuation sign (exclamation mark, question mark, semicolon and colon). This rule is used only in a few selected languages like French or Breton, where space before double punctuation sign is a typographic rule.

Vezi și:

French and English spacing on Wikipedia

Expresie regulată

Nou în versiunea 3.9.

Summary Traducerea nu se potrivește cu expresia regulată:

Domeniul de aplicare translated strings

Check class `weblate.checks.placeholders.RegexCheck`

Flag to enable `regex`

Flag to ignore `ignore-regex`

Translation does not match regular expression. The expression is either extracted from the translation file or defined manually using `regex` flag:

```
regex:^foo|bar$
```

Același plural

Summary Unele forme de plural sunt traduse în același mod

Domeniul de aplicare translated strings

Check class `weblate.checks.consistency.SamePluralsCheck`

Flag to ignore `ignore-same-plurals`

Check that fails if some plural forms are duplicated in the translation. In most languages they have to be different.

Începutul liniei noi

Summary Sursa și traducerea nu încep ambele cu o linie nouă

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.BeginNewlineCheck`

Flag to ignore `ignore-begin-newline`

Newlines usually appear in source strings for good reason, omissions or additions can lead to formatting problems when the translated text is put to use.

Vezi și:

Linie nouă de urmărire

Spații de pornire

Summary Sursa și traducerea nu încep ambele cu același număr de spații

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.BeginSpaceCheck`

Flag to ignore `ignore-begin-space`

A space in the beginning of a string is usually used for indentation in the interface and thus important to keep.

Linie nouă de urmărire

Summary Sursa și traducerea nu se termină ambele cu o linie nouă

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.EndNewlineCheck`

Flag to ignore `ignore-end-newline`

Newlines usually appear in source strings for good reason, omissions or additions can lead to formatting problems when the translated text is put to use.

Vezi și:

Începutul liniei noi

Spațiu de urmărire

Summary Sursa și traducerea nu se termină amândouă cu un spațiu

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.EndSpaceCheck`

Flag to ignore `ignore-end-space`

Checks that trailing spaces are replicated between both source and translation.

Trailing space is usually utilized to space out neighbouring elements, so removing it might break layout.

Traducere neschimbată

Summary Sursa și traducerea sunt identice

Domeniul de aplicare translated strings

Check class `weblate.checks.same.SameCheck`

Flag to ignore `ignore-same`

Happens if the source and corresponding translation strings is identical, down to at least one of the plural forms. Some strings commonly found across all languages are ignored, and various markup is stripped. This reduces the number of false positives.

This check can help find strings mistakenly untranslated.

The default behavior of this check is to exclude words from the built-in blacklist from the checking. These are words which are frequently not being translated. This is useful to avoid false positives on short strings, which consist only of single word which is same in several languages. This blacklist can be disabled by adding `strict-same` flag to string or component.

Vezi și:

Component configuration, *Personalizarea comportamentului cu ajutorul steguleţelor*

HTML nesigur

Nou în versiunea 3.9.

Summary Traducerea folosește marcaje HTML nesigure

Domeniul de aplicare translated strings

Check class `weblate.checks.markup.SafeHTMLCheck`

Flag to enable `safe-html`

Flag to ignore `ignore-safe-html`

The translation uses unsafe HTML markup. This check has to be enabled using `safe-html` flag (see *Personalizarea comportamentului cu ajutorul steguleţelor*). There is also accompanied autofixer which can automatically sanitize the markup.

Vezi şi:

The HTML check is performed by the Bleach library developed by Mozilla.

URL

Nou în versiunea 3.5.

Summary Traducerea nu conține un URL

Domeniul de aplicare translated strings

Check class `weblate.checks.markup.URLCheck`

Flag to enable `url`

Flag to ignore `ignore-url`

The translation does not contain an URL. This is triggered only in case the unit is marked as containing URL. In that case the translation has to be a valid URL.

Marcarea XML

Summary Etichetele XML din traducere nu se potrivesc cu sursa

Domeniul de aplicare translated strings

Check class `weblate.checks.markup.XMLTagsCheck`

Flag to ignore `ignore-xml-tags`

This usually means the resulting output will look different. In most cases this is not a desired result from changing the translation, but occasionally it is.

Checks that XML tags are replicated between both source and translation.

Notă: This check is disabled by the `safe-html` flag as the HTML cleanup done by it can produce HTML markup which is not valid XML.

Sintaxa XML

Nou în versiunea 2.8.

Summary Traducerea nu este validă XML

Domeniul de aplicare translated strings

Check class `weblate.checks.markup.XMLValidityCheck`

Flag to ignore `ignore-xml-invalid`

The XML markup is not valid.

Notă: This check is disabled by the `safe-html` flag as the HTML cleanup done by it can produce HTML markup which is not valid XML.

Spațiu cu lățime zero

Summary Traducerea conține un caracter suplimentar de spațiu cu lățimea zero

Domeniul de aplicare translated strings

Check class `weblate.checks.chars.ZeroWidthSpaceCheck`

Flag to ignore `ignore-zero-width-space`

Zero-width space (<U+200B>) characters are used to break messages within words (word wrapping).

As they are usually inserted by mistake, this check is triggered once they are present in translation. Some programs might have problems when this character is used.

Vezi și:

Zero width space on Wikipedia

1.6.4 Source checks

Source checks can help developers improve the quality of source strings.

Elipsă

Summary Șirul de caractere utilizează trei puncte (....) în loc de un caracter de elipsă (…)

Domeniul de aplicare source strings

Check class `weblate.checks.source.EllipsisCheck`

Flag to ignore `ignore-ellipsis`

This fails when the string uses three dots (`...`) when it should use an ellipsis character (...).

Using the Unicode character is in most cases the better approach and looks better rendered, and may sound better with text-to-speech.

Vezi și:

Ellipsis on Wikipedia

Lungă netradusă

Nou în versiunea 4.1.

Summary Șirul nu a mai fost tradus de mult timp

Domeniul de aplicare source strings

Check class `weblate.checks.source.LongUntranslatedCheck`

Flag to ignore `ignore-long-untranslated`

When the string has not been translated for a long time, it is can indicate problem in a source string making it hard to translate.

Controale multiple eșuate

Summary Traducerile în mai multe limbi au verificări eșuate

Domeniul de aplicare source strings

Check class `weblate.checks.source.MultipleFailingCheck`

Flag to ignore `ignore-multiple-failures`

Numerous translations of this string have failing quality checks. This is usually an indication that something could be done to improve the source string.

This check failing can quite often be caused by a missing full stop at the end of a sentence, or similar minor issues which translators tend to fix in translation, while it would be better to fix it in the source string.

Variabile multiple fără nume

Nou în versiunea 4.1.

Summary Există mai multe variabile fără nume în șir, ceea ce face imposibilă reordonarea lor de către traducători

Domeniul de aplicare source strings

Check class `weblate.checks.format.MultipleUnnamedFormatsCheck`

Flag to ignore `ignore-unnamed-format`

There are multiple unnamed variables in the string, making it impossible for translators to reorder them.

Consider using named variables instead to allow translators to reorder them.

Nepluralizat

Summary Șirul este folosit la plural, dar nu se folosesc formele de plural

Domeniul de aplicare source strings

Check class `weblate.checks.source.OptionalPluralCheck`

Flag to ignore `ignore-optional-plural`

The string is used as a plural, but does not use plural forms. In case your translation system supports this, you should use the plural aware variant of it.

For example with Gettext in Python it could be:

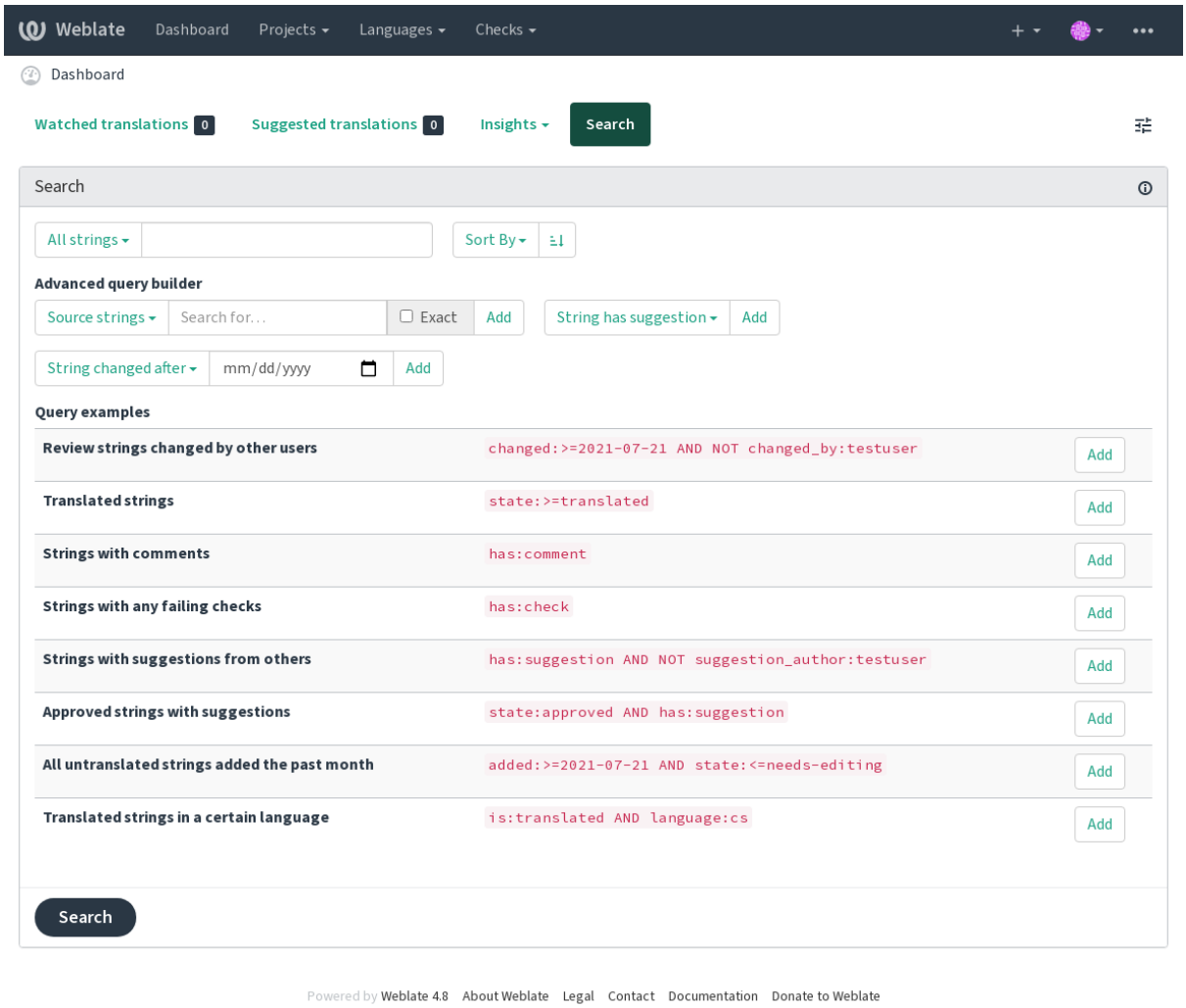
```
from gettext import ngettext

print ngettext("Selected %d file", "Selected %d files", files) % files
```

1.7 Searching

Nou în versiunea 3.9.

Advanced queries using boolean operations, parentheses, or field specific lookup can be used to find the strings you want.

When no field is defined, the lookup happens on *Source*, *Translate* and *Context* fields.



1.7.1 Simple search

Any phrase typed into the search box is split into words. Strings containing any of them are shown. To look for an exact phrase, put „the searchphrase" into quotes (both single («) and double („) quotes will work): `"this is a quoted string"` or `'another quoted string'`.

1.7.2 Fields

`source:TEXT` Source string case insensitive search.

`target:TEXT` Target string case insensitive search.

`context:TEXT` Context string case insensitive search.

`key:TEXT` Key string case insensitive search.

`note:TEXT` Comment string case insensitive search.

`location:TEXT` Location string case insensitive search.

`priority:NUMBER` String priority.

`added:DATETIME` Timestamp for when the string was added to Weblate.

`state:TEXT` State search (`approved`, `translated`, `needs-editing`, `empty`, `read-only`), supports *Field operators*.

`pending:BOOLEAN` String pending for flushing to VCS.

`has:TEXT` Search for string having attributes - `plural`, `context`, `suggestion`, `comment`, `check`, `dismissed-check`, `translation`, `variant`, `screenshot`, `flags`, `explanation`, `glossary`, `note`.

`is:TEXT` Search for string states (`pending`, `translated`, `untranslated`).

`language:TEXT` String target language.

`component:TEXT` Component slug or name case insensitive search, see *Component slug* and *Denumirea componentei*.

`project:TEXT` Project slug, see *URL slug*.

`changed_by:TEXT` String was changed by author with given username.

`changed:DATETIME` String content was changed on date, supports *Field operators*.

`change_time:DATETIME` String was changed on date, supports *Field operators*, unlike `changed` this includes event which don't change content and you can apply custom action filtering using `change_action`.

`change_action:TEXT` Filters on change action, useful together with `change_time`. Accepts English name of the change action, either quoted and with spaces or lowercase and spaces replaced by a hyphen. See *Searching for changes* for examples.

`check:TEXT` String has failing check.

`dismissed_check:TEXT` String has dismissed check.

`comment:TEXT` Search in user comments.

`comment_author:TEXT` Filter by comment author.

`suggestion:TEXT` Search in suggestions.

`suggestion_author:TEXT` Filter by suggestion author.

`explanation:TEXT` Search in explanations.

1.7.3 Boolean operators

You can combine lookups using `AND`, `OR`, `NOT` and parentheses to form complex queries. For example: `state:translated AND (source:hello OR source:bar)`

1.7.4 Field operators

You can specify operators, ranges or partial lookups for date or numeric searches:

`state:>=translated` State is `translated` or better (`approved`).

`changed:2019` Changed in year 2019.

`changed:[2019-03-01 to 2019-04-01]` Changed between two given dates.

1.7.5 Exact operators

You can do an exact match query on different string fields using = operator. For example, to search for all source strings exactly matching `hello world`, use: `source:="hello world"`. For searching single word expressions, you can skip quotes. For example, to search for all source strings matching `hello`, you can use: `source:=hello`.

1.7.6 Searching for changes

Nou în versiunea 4.4.

Searching for history events can be done using `change_action` and `change_time` operators.

For example, searching for strings marked for edit in 2018 can be entered as `change_time:2018 AND change_action:marked-for-edit` or `change_time:2018 AND change_action:"Marked for edit"`.

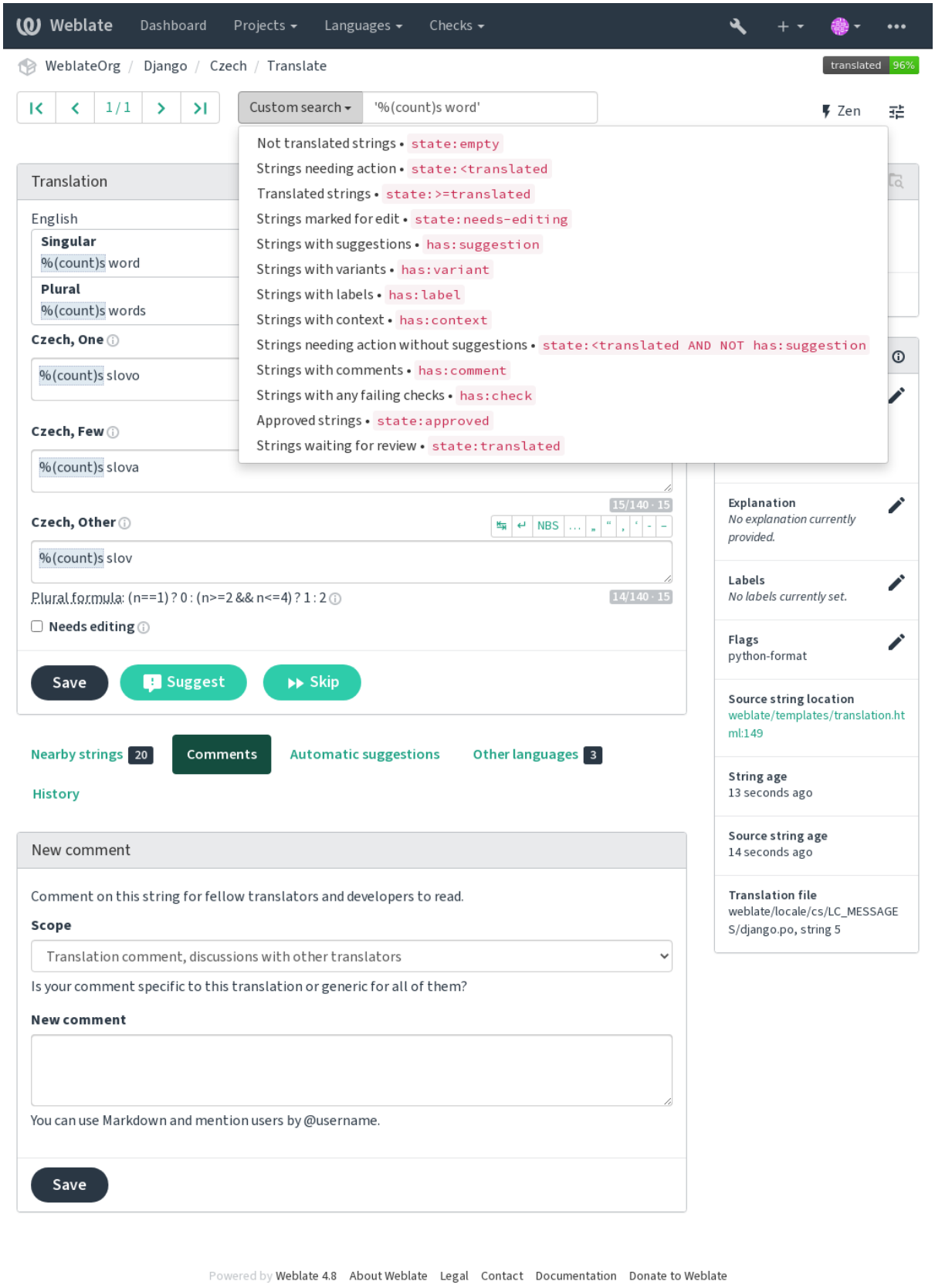
1.7.7 Regular expressions

Anywhere text is accepted you can also specify a regular expression as `r"regexp"`.

For example, to search for all source strings which contain any digit between 2 and 5, use `source:r"[2-5]"`.

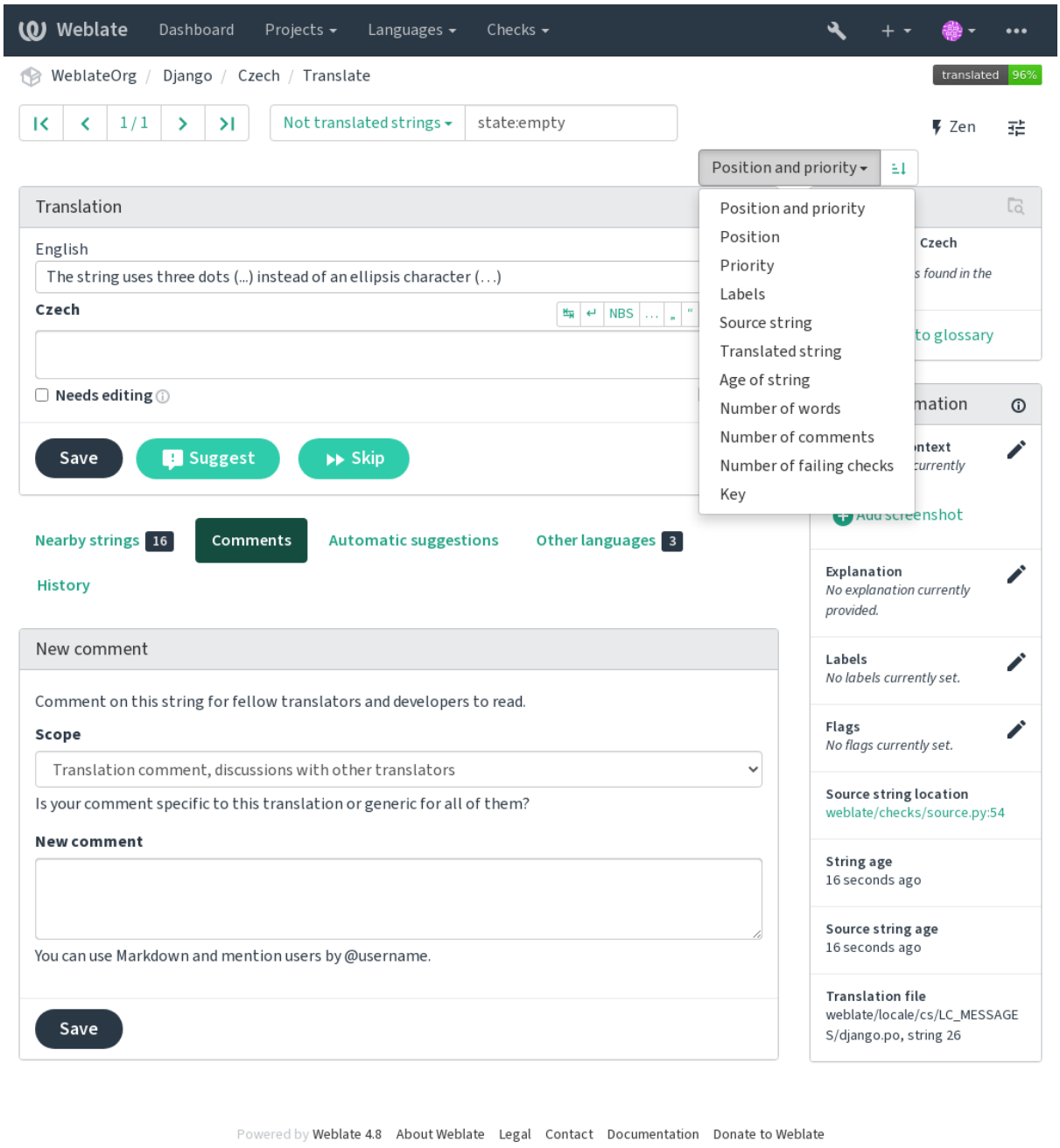
1.7.8 Predefined queries

You can select out of predefined queries on the search page, this allows you to quickly access the most frequent searches:


Weblate
Dashboard
Projects
Languages
Checks
WeblateOrg / Django / Czech / Translate
translated 96%
1 / 1
Custom search
'%(count)s word'
Zen
Not translated strings • state:empty
Strings needing action • state:<translated
Translated strings • state:>=translated
Strings marked for edit • state:needs-editing
Strings with suggestions • has:suggestion
Strings with variants • has:variant
Strings with labels • has:label
Strings with context • has:context
Strings needing action without suggestions • state:<translated AND NOT has:suggestion
Strings with comments • has:comment
Strings with any failing checks • has:check
Approved strings • state:approved
Strings waiting for review • state:translated
Translation
English
Singular
%(count)s word
Plural
%(count)s words
Czech, One
%(count)s slovo
Czech, Few
%(count)s slova
15/140 · 15
Czech, Other
%(count)s slov
Plural formula: (n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2
14/140 · 15
Needs editing
Save
Suggest
Skip
Nearby strings 20
Comments
Automatic suggestions
Other languages 3
History
New comment
Comment on this string for fellow translators and developers to read.
Scope
Translation comment, discussions with other translators
Is your comment specific to this translation or generic for all of them?
New comment
You can use Markdown and mention users by @username.
Save
Explanation
No explanation currently provided.
Labels
No labels currently set.
Flags
python-format
Source string location
weblate/templates/translation.html:149
String age
13 seconds ago
Source string age
14 seconds ago
Translation file
weblate/locale/cs/LC_MESSAGES/django.po, string 5
Powered by Weblate 4.8
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.7.9 Ordering the results

There are many options to order the strings according to your needs:



1.8 Translation workflows

Using Weblate is a process that brings your users closer to you, by bringing you closer to your translators. It is up to you to decide how many of its features you want to make use of.

The following is not a complete list of ways to configure Weblate. You can base other workflows on the most usual examples listed here.

1.8.1 Translation access

The *access control* is not discussed in detail as a whole in the workflows, as most of its options can be applied to any workflow. Please consult the respective documentation on how to manage access to translations.

In the following chapters, *any user* means a user who has access to the translation. It can be any authenticated user if the project is public, or a user that has a *Translate* permission for the project.

1.8.2 Translation states

Each translated string can be in one of following states:

Netradus Translation is empty, it might or not be stored in the file, depending on the file format.

Necesită editare Translation needs editing, this is usually the result of a source string change, fuzzy matching or translator action. The translation is stored in the file, depending on the file format it might be marked as needing edit (for example as it gets a `fuzzy` flag in the Gettext file).

În așteptare pentru revizuire Translation is made, but not reviewed. It is stored in the file as a valid translation.

Aprobat Translation has been approved in the review. It can no longer be changed by translators, but only by reviewers. Translators can only add suggestions to it.

Sugestii Suggestions are stored in Weblate only and not in the translation file.

The states are represented in the translation files when possible.

Sugestie: In case file format you use does not support storing states, you might want to use *Marcați traducerile neschimbate ca fiind „Needs editing”* addon to flag unchanged strings as needing editing.

Vezi și:

Translation types capabilities, *Translation workflows*

1.8.3 Direct translation

This is most usual setup for smaller teams, anybody can directly translate. This is also the default setup in Weblate.

- *Any user* can edit translations.
- Suggestions are optional ways to suggest changes, when translators are not sure about the change.

Setting	Value	Notă
Activați recenzii	oprit	Configured at project level.
Enable suggestions	pe	It is useful for users to be able to suggest when they are not sure.
Votul prin sugestie	oprit	
Sugestii de autoacceptare	0	
Translators group	*Users*	Or *Translate* with *per-project access control*.
Reviewers group	N/A	Not used.

1.8.4 Peer review

With this workflow, anybody can add suggestions, and need approval from additional member(s) before it is accepted as a translation.

- *Any user* can add suggestions.
- *Any user* can vote for suggestions.
- Suggestions become translations when given a predetermined number of votes.

Setting	Value	Notă
Activați recenzii	oprit	Configured at project level.
Enable suggestions	pe	
Votul prin sugestie	oprit	
Sugestii de autoacceptare	1	You can set higher value to require more peer reviews.
Translators group	*Users*	Or *Translate* with *per-project access control*.
Reviewers group	N/A	Not used, all translators review.

1.8.5 Dedicated reviewers

Nou în versiunea 2.18: The proper review workflow is supported since Weblate 2.18.

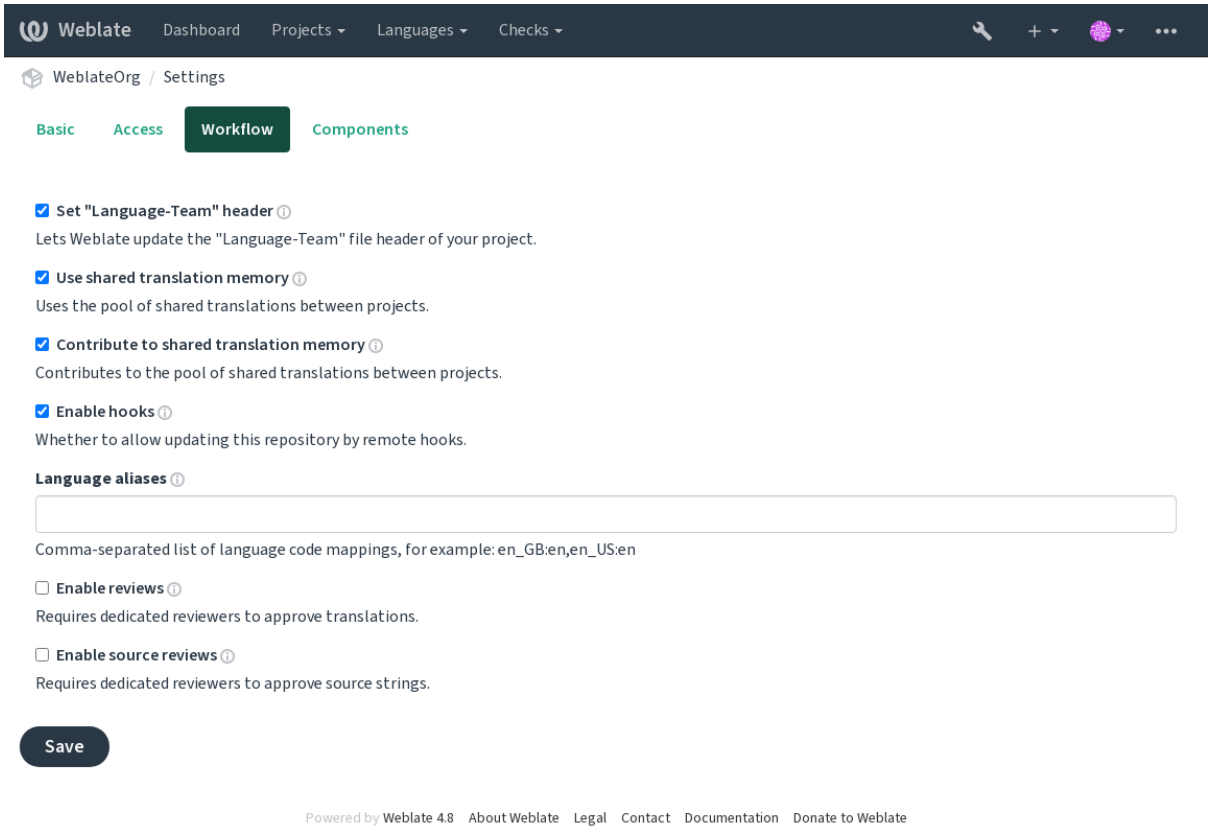
With dedicated reviewers you have two groups of users, one able to submit translations, and one able to review them to ensure translations are consistent and that the quality is good.

- *Any user* can edit unapproved translations.
- *Reviewer* can approve / unapprove strings.
- *Reviewer* can edit all translations (including approved ones).
- Suggestions can also be used to suggest changes for approved strings.

Setting	Value	Notă
Activați recenzii	pe	Configured at project level.
Enable suggestions	oprit	It is useful for users to be able to suggest when they are not sure.
Votul prin sugestie	oprit	
Sugestii de autoacceptare	0	
Translators group	*Users*	Or *Translate* with *per-project access control*.
Reviewers group	*Reviewers*	Or *Review* with *per-project access control*.

1.8.6 Turning on reviews

Reviews can be turned on in the project configuration, from the *Workflow* subpage of project settings (to be found in the *Manage* → *Settings* menu):

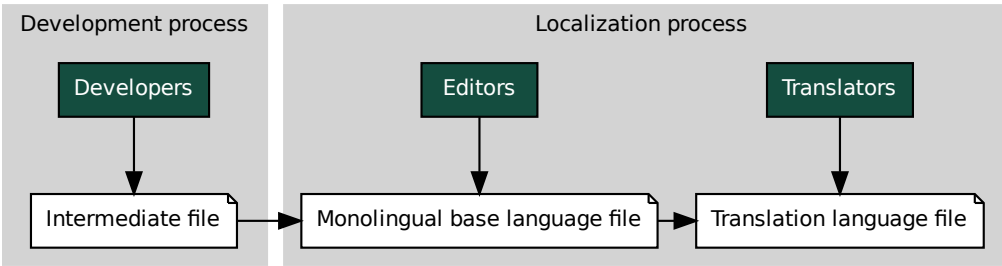


Notă: Depending on Weblate configuration, the setting might not be available to you. For example on Hosted Weblate this is not available for projects hosted for free.

1.8.7 Quality gateway for the source strings

In many cases the original source language strings are coming from developers, because they write the code and provide initial strings. However developers are often not a native speakers in the source language and do not provide desired quality of the source strings. The intermediate translation can help you in addressing this - there is additional quality gateway for the strings between developers and translators and users.

By setting *Fișier de limbaj intermediar*, this file will be used as source for the strings, but it will be edited to source language to polish it. Once the string is ready in the source language, it will be also available for translators to translate into additional languages.



Vezi și:

Fișier de limbaj intermediar, *Fișier de limbă de bază monolingvă*, *Bilingual and monolingual formats*

1.8.8 Source strings reviews

With *Activați revizuirile sursei* enabled, the review process can be applied on the source strings. Once enabled, users can report issues in the source strings. The actual process depends on whether you use bilingual or monolingual formats.

For monolingual formats, the source string review behaves similarly as with *Dedicated reviewers* - once issue is reported on the source string, it is marked as *Needs editing*.

The bilingual formats do not allow direct editing of the source strings (these are typically extracted directly from the source code). In this case *Source needs review* label is attached to strings reported by translators. You should review such strings and either edit them in the source or remove the label.

Vezi și:

Bilingual and monolingual formats, *Dedicated reviewers*, labels, *Comentarii*

1.9 Frequently Asked Questions

1.9.1 Configurație

How to create an automated workflow?

Weblate can handle all the translation things semi-automatically for you. If you give it push access to your repository, the translations can happen without interaction, unless some merge conflict occurs.

1. Set up your Git repository to tell Weblate when there is any change, see *Cârlige de notificare* for info on how to do it.
2. Set a push URL at your *Component configuration* in Weblate, this allows Weblate to push changes to your repository.
3. Turn on *Împingeți pe comitere* on your *Component configuration* in Weblate, this will make Weblate push changes to your repository whenever they happen at Weblate.

Vezi și:

Traducere continuă, *Avoiding merge conflicts*

How to access repositories over SSH?

Please see *Accessing repositories* for info on setting up SSH keys.

How to fix merge conflicts in translations?

Merge conflicts happen from time to time when the translation file is changed in both Weblate and the upstream repository concurrently. You can usually avoid this by merging Weblate translations prior to making changes in the translation files (e.g. before running msgmerge). Just tell Weblate to commit all pending translations (you can do it in *Repository maintenance* in the *Manage* menu) and merge the repository (if automatic push is not on).

If you've already encountered a merge conflict, the easiest way to solve all conflicts locally on your machine, is to add Weblate as a remote repository, merge it into upstream and fix any conflicts. Once you push changes back, Weblate will be able to use the merged version without any other special actions.

Notă: Depending on your setup, access to the Weblate repository might require authentication. When using the built-in *Git exporter* in Weblate, you authenticate with your username and the API key.

```
# Commit all pending changes in Weblate, you can do this in the UI as well:
wlc commit
# Lock the translation in Weblate, again this can be done in the UI as well:
wlc lock
# Add Weblate as remote:
git remote add weblate https://hosted.weblate.org/git/project/component/
# You might need to include credentials in some cases:
git remote add weblate https://username:APIKEY@hosted.weblate.org/git/project/
↪component/

# Update weblate remote:
git remote update weblate

# Merge Weblate changes:
git merge weblate/main

# Resolve conflicts:
edit …
git add …
…
git commit

# Push changes to upstream repository, Weblate will fetch merge from there:
git push

# Open Weblate for translation:
wlc unlock
```

If you're using multiple branches in Weblate, you can do the same to all of them:

```
# Add and update Weblate remotes
git remote add weblate-one https://hosted.weblate.org/git/project/one/
git remote add weblate-second https://hosted.weblate.org/git/project/second/
git remote update weblate-one weblate-second

# Merge QA_4_7 branch:
git checkout QA_4_7
git merge weblate-one/QA_4_7
... # Resolve conflicts
git commit

# Merge main branch:
git checkout main
git merge weblates-second/main
... # Resolve conflicts
git commit

# Push changes to the upstream repository, Weblate will fetch the merge from there:
git push
```

In case of gettext PO files, there is a way to merge conflicts in a semi-automatic way:

Fetch and keep a local clone of the Weblate Git repository. Also get a second fresh local clone of the upstream Git repository (i. e. you need two copies of the upstream Git repository: An intact and a working copy):

```
# Add remote:
git remote add weblate /path/to/weblate/snapshot/
```

(continues on next page)

(continuare din pagina precedentă)

```
# Update Weblate remote:
git remote update weblate

# Merge Weblate changes:
git merge weblate/main

# Resolve conflicts in the PO files:
for PO in `find . -name '*.po'` ; do
    msgcat --use-first /path/to/weblate/snapshot/$PO\
               /path/to/upstream/snapshot/$PO -o $PO.merge
    msgmerge --previous --lang=${PO%.po} $PO.merge domain.pot -o $PO
    rm $PO.merge
    git add $PO
done
git commit

# Push changes to the upstream repository, Weblate will fetch merge from there:
git push
```

Vezi și:

How to export the Git repository that Weblate uses?, *Traducere continuă*, *Avoiding merge conflicts*, *Weblate Client*

How do I translate several branches at once?

Weblate supports pushing translation changes within one *Project configuration*. For every *Component configuration* which has it turned on (the default behavior), the change made is automatically propagated to others. This way translations are kept synchronized even if the branches themselves have already diverged quite a lot, and it is not possible to simply merge translation changes between them.

Once you merge changes from Weblate, you might have to merge these branches (depending on your development workflow) discarding differences:

```
git merge -s ours origin/maintenance
```

Vezi și:

Keeping translations same across components

How to translate multi-platform projects?

Weblate supports a wide range of file formats (see *Supported file formats*) and the easiest approach is to use the native format for each platform.

Once you have added all platform translation files as components in one project (see *Adding translation projects and components*), you can utilize the translation propagation feature (turned on by default, and can be turned off in the *Component configuration*) to translate strings for all platforms at once.

Vezi și:

Keeping translations same across components

How to export the Git repository that Weblate uses?

There is nothing special about the repository, it lives under the `DATA_DIR` directory and is named `vcs/<project>/<component>/`. If you have SSH access to this machine, you can use the repository directly.

For anonymous access, you might want to run a Git server and let it serve the repository to the outside world.

Alternatively, you can use *Git exporter* inside Weblate to automate this.

What are the options for pushing changes back upstream?

This heavily depends on your setup, Weblate is quite flexible in this area. Here are examples of some workflows used with Weblate:

- Weblate automatically pushes and merges changes (see *How to create an automated workflow?*).
- You manually tell Weblate to push (it needs push access to the upstream repository).
- Somebody manually merges changes from the Weblate git repository into the upstream repository.
- Somebody rewrites history produced by Weblate (e.g. by eliminating merge commits), merges changes, and tells Weblate to reset the content in the upstream repository.

Of course you are free to mix all of these as you wish.

How can I limit Weblate access to only translations, without exposing source code to it?

You can use git submodule for separating translations from source code while still having them under version control.

1. Create a repository with your translation files.
2. Add this as a submodule to your code:

```
git submodule add git@example.com:project-translations.git path/to/translations
```

3. Link Weblate to this repository, it no longer needs access to the repository containing your source code.
4. You can update the main repository with translations from Weblate by:

```
git submodule update --remote path/to/translations
```

Please consult the git submodule documentation for more details.

How can I check whether my Weblate is set up properly?

Weblate includes a set of configuration checks which you can see in the admin interface, just follow the *Performance report* link in the admin interface, or open the `/manage/performance/` URL directly.

Why are all commits committed by Weblate <noreply@weblate.org>?

This is the default committer name, configured when you create a translation component. You can change it in the administration at any time.

The author of every commit (if the underlying VCS supports it) is still recorded correctly as the user that made the translation.

Vezi și:

Component configuration

1.9.2 Usage

How do I review the translations of others?

- There are several review based workflows available in Weblate, see *Translation workflows*.
- You can subscribe to any changes made in *Notificări* and then check others contributions as they come in by e-mail.
- There is a review tool available at the bottom of the translation view, where you can choose to browse translations made by others since a given date.

Vezi și:

Translation workflows

How do I provide feedback on a source string?

On context tabs below translation, you can use the *Comments* tab to provide feedback on a source string, or discuss it with other translators.

Vezi și:

report-source, *Comentarii*

How can I use existing translations while translating?

- All translations within Weblate can be used thanks to shared translation memory.
- You can import existing translation memory files into Weblate.
- Use the import functionality to load compendium as translations, suggestions or translations needing review. This is the best approach for a one-time translation using a compendium or a similar translation database.
- You can set up *tmserver* with all databases you have and let Weblate use it. This is good when you want to use it several times during translation.
- Another option is to translate all related projects in a single Weblate instance, which will make it automatically pick up translations from other projects as well.

Vezi și:

Traducere automată, *Sugestii automate*, *Memorie de traducere*

Does Weblate update translation files besides translations?

Weblate tries to limit changes in translation files to a minimum. For some file formats it might unfortunately lead to reformatting the file. If you want to keep the file formatted your way, please use a pre-commit hook for that.

Vezi și:

updating-target-files

Where do language definitions come from and how can I add my own?

The basic set of language definitions is included within Weblate and Translate-toolkit. This covers more than 150 languages and includes info about plural forms or text direction.

You are free to define your own languages in the administrative interface, you just need to provide info about it.

Vezi și:

Language definitions

Can Weblate highlight changes in a fuzzy string?

Weblate supports this, however it needs the data to show the difference.

For Gettext PO files, you have to pass the parameter `--previous` to **msgmerge** when updating PO files, for example:

```
msgmerge --previous -U po/cs.po po/phpmyadmin.pot
```

For monolingual translations, Weblate can find the previous string by ID, so it shows the differences automatically.

Why does Weblate still show old translation strings when I've updated the template?

Weblate does not try to manipulate the translation files in any way other than allowing translators to translate. So it also does not update the translatable files when the template or source code have been changed. You simply have to do this manually and push changes to the repository, Weblate will then pick up the changes automatically.

Notă: It is usually a good idea to merge changes done in Weblate before updating translation files, as otherwise you will usually end up with some conflicts to merge.

For example with gettext PO files, you can update the translation files using the **msgmerge** tool:

```
msgmerge -U locale/cs/LC_MESSAGES/django.mo locale/django.pot
```

In case you want to do the update automatically, you can install addon *Actualizarea fișierelor PO pentru a se potrivi cu POT (msgmerge)*.

Vezi și:

updating-target-files

1.9.3 Troubleshooting

Requests sometimes fail with „too many open files" error

This happens sometimes when your Git repository grows too much and you have many of them. Compressing the Git repositories will improve this situation.

The easiest way to do this is to run:

```
# Go to DATA_DIR directory
cd data/vcs
# Compress all Git repositories
for d in */* ; do
    pushd $d
    git gc
    popd
done
```

Vezi și:

DATA_DIR

When accessing the site I get a „Bad Request (400)” error

This is most likely caused by an improperly configured *ALLOWED_HOSTS*. It needs to contain all hostnames you want to access on your Weblate. For example:

```
ALLOWED_HOSTS = ["weblate.example.com", "weblate", "localhost"]
```

Vezi și:

Allowed hosts setup

What does mean „There are more files for the single language (en)”?

This typically happens when you have translation file for source language. Weblate keeps track of source strings and reserves source language for this. The additional file for same language is not processed.

- In case the translation to the source language is desired, please change the *Limba sursă* in the component settings.
- In case the translation file for the source language is not needed, please remove it from the repository.
- In case the translation file for the source language is needed, but should be ignored by Weblate, please adjust the *Filtru lingvistic* to exclude it.

Sugestie: You might get similar error message for other languages as well. In that case the most likely reason is that several files map to single language in Weblate.

This can be caused by using obsolete language codes together with new one (`ja` and `jp` for Japanese) or including both country specific and generic codes (`fr` and `fr_FR`). See *Parsing language codes* for more details.

1.9.4 Funcționalități

Does Weblate support other VCSes than Git and Mercurial?

Weblate currently does not have native support for anything other than *Git* (with extended support for *GitHub*, *Gerrit* and *Subversion*) and *Mercurial*, but it is possible to write backends for other VCSes.

You can also use *Git remote helpers* in Git to access other VCSes.

Weblate also supports VCS-less operation, see *Local files*.

Notă: For native support of other VCSes, Weblate requires using distributed VCS, and could probably be adjusted to work with anything other than Git and Mercurial, but somebody has to implement this support.

Vezi și:

Integrarea controlului versiunilor

How does Weblate credit translators?

Every change made in Weblate is committed into VCS under the translators name. This way every single change has proper authorship, and you can track it down using the standard VCS tools you use for code.

Additionally, when the translation file format supports it, the file headers are updated to include the translator's name.

Vezi și:

`list_translators`, ../devel/reporting

Why does Weblate force showing all PO files in a single tree?

Weblate was designed in a way that every PO file is represented as a single component. This is beneficial for translators, so they know what they are actually translating.

Schimbat în versiunea 4.2: Translators can translate all the components of a project into a specific language as a whole.

Why does Weblate use language codes such sr_Latn or zh_Hant?

These are language codes defined by **RFC 5646** to better indicate that they are really different languages instead previously wrongly used modifiers (for `@latin` variants) or country codes (for Chinese).

Weblate still understands legacy language codes and will map them to current one - for example `sr@latin` will be handled as `sr_Latn` or `zh@CN` as `zh_Hans`.

Notă: Weblate defaults to POSIX style language codes with underscore, see *Language definitions* for more details.

Vezi și:

Language definitions, *Stil de cod lingvistic*, *Adding new translations*

1.10 Supported file formats

Weblate supports most translation format understood by translate-toolkit, however each format being slightly different, some issues with formats that are not well tested can arise.

Vezi și:

Translation Related File Formats

Notă: When choosing a file format for your application, it's better to stick some well established format in the toolkit/platform you use. This way your translators can additionally use whatever tools they are used to, and will more likely contribute to your project.

1.10.1 Bilingual and monolingual formats

Both monolingual and bilingual formats are supported. Bilingual formats store two languages in single file—source and translation (typical examples are *GNU gettext*, *XLIFF* or *Apple iOS strings*). On the other side, monolingual formats identify the string by ID, and each language file contains only the mapping of those to any given language (typically *Android string resources*). Some file formats are used in both variants, see the detailed description below.

For correct use of monolingual files, Weblate requires access to a file containing complete list of strings to translate with their source—this file is called *Fișier de limbă de bază monolingvă* within Weblate, though the naming might vary in your paradigm.

Additionally this workflow can be extended by utilizing *Fișier de limbaj intermediar* to include strings provided by developers, but not to be used as is in the final strings.

1.10.2 Automatic detection

Weblate can automatically detect several widespread file formats, but this detection can harm your performance and will limit features specific to given file format (for example automatic addition of new translations).

1.10.3 Translation types capabilities

Capabilities of all supported formats:

Format	Linguality[Pagina 65, 1]	Plurals[Pagina 65, 2]	Comments[Pagina 65, 3]	Context[Pagina 65, 4]	Location[Pagina 65, 5]	Flags[Pagina 65, 8]	Additional states[Pagina 65, 6]
GNU gettext	bilingual	yes	yes	yes	yes	yes[9]	needs editing
Monolingual gettext	mono	yes	yes	yes	yes	yes[?]	needs editing
XLIFF	both	yes	yes	yes	yes	yes[10]	needs editing, approved
Java properties	both	no	yes	no	no	no	
mi18n lang files	mono	no	yes	no	no	no	
Proprietăți GWT	mono	yes	yes	no	no	no	
Joomla translations	mono	no	yes	no	yes	no	
Qt Linguist .ts	both	yes	yes	no	yes	yes[?]	needs editing
Android string resources	mono	yes	yes[7]	no	no	yes[?]	
Apple iOS strings	bilingual	no	yes	no	no	no	
Șiruri PHP	mono	no[11]	yes	no	no	no	
JSON files	mono	no	no	no	no	no	
JSON i18next files	mono	yes	no	no	no	no	

continues on next page

Tabelul 1 – continuare din pagina precedentă

Format	Linguality[Pagina 65, 1]	Plurals[Pagina 65, 2]	Comments[Pagina 65, 3]	Context[Pagina 65, 4]	Location[Pagina 65, 5]	Flags[Pagina 65, 8]	Additional states[Pagina 65, 6]
go-i18n JSON files	mono	yes	no	no	no	no	
ARB File	mono	yes	yes	no	no	no	
WebExtension JSON	mono	yes	yes	no	no	no	
.XML resource files	mono	no	yes	no	no	yes[?]	
CSV files	both	no	yes	yes	yes	no	needs editing
YAML files	mono	no	yes	no	no	no	
Ruby YAML files	mono	yes	yes	no	no	no	
DTD files	mono	no	no	no	no	no	
Flat XML files	mono	no	no	no	no	yes[?]	
Windows RC files	mono	no	yes	no	no	no	
Excel Deschidere XML	mono	no	yes	yes	yes	no	needs editing
Fișiere de metadate ale magazinului de aplicații	mono	no	no	no	no	no	
Subtitle files	mono	no	no	no	yes	no	
HTML files	mono	no	no	no	no	no	
OpenDocument Format	mono	no	no	no	no	no	
IDML Format	mono	no	no	no	no	no	
INI translations	mono	no	no	no	no	no	
Inno Setup INI translations	mono	no	no	no	no	no	
TermBase eXchange format	bilingual	no	yes	no	no	yes[?]	
Text files	mono	no	no	no	no	no	
Stringsdict format	mono	yes	yes	no	no	no	
Fluent format	mono	no[12]	yes	no	no	no	

Șiruri de caractere numai pentru citire

Nou în versiunea 3.10.

Read-only strings from translation files will be included, but can not be edited in Weblate. This feature is natively supported by few formats (*XLIFF* and *Android string resources*), but can be emulated in others by adding a `read-only` flag, see *Personalizarea comportamentului cu ajutorul steguleţelor*.

1.10.4 GNU gettext

Most widely used format for translating libre software.

Contextual info stored in the file is supported by adjusting its headers or linking to corresponding source files.

The bilingual gettext PO file typically looks like this:

```
#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "Monday"
msgstr "Pondělí"

#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "Tuesday"
msgstr "Úterý"

#: weblate/accounts/avatar.py:163
msgctxt "No known user"
msgid "None"
msgstr "Žádný"
```

Typical Weblate *Component configuration*	
Masca de fișier	`po/*.po`
Fișier de limbă de bază monolingvă	*Empty*
Șablon pentru traduceri noi	`po/messages.pot`
Formatul fișierului	*Gettext PO file*

Vezi și:

devel/gettext, devel/sphinx, Gettext on Wikipedia, PO Files, *Actualizarea variabilei ALL_LINGUAS în fișierul „configure”*, *Personalizați ieșirea gettext*, *Actualizarea fișierului LINGUAS*, *Generarea de fișiere MO*, *Actualizarea fișierelor PO pentru a se potrivi cu POT (msgmerge)*

[1] See *Bilingual and monolingual formats*
[2] Plurals are necessary to properly localize strings with variable count.
[3] Comments can be used to pass additional info about the string to translate.
[4] Context is used to differentiate identical strings used in different scopes (for example *Sun* can be used as an abbreviated name of the day „Sunday” or as the name of our closest star).
[5] Location of a string in source code might help proficient translators figure out how the string is used.
[8] See *Personalizarea comportamentului cu ajutorul steguleţelor*
[6] Additional states supported by the file format in addition to „Not translated” and „Translated”.
[9] The gettext type comments are used as flags.
[10] The flags are extracted from the non-standard attribute `weblate-flags` for all XML based formats. Additionally `max-length:N` is supported through the `maxwidth` attribute as defined in the XLIFF standard, see *Specifying translation flags*.
[7] XML comment placed before the `<string>` element, parsed as a developer comment.
[11] The plurals are supported only for Laravel which uses in string syntax to define them, see Localization in Laravel.
[12] Plurals are handled in the syntax of the strings and not exposed as plurals in Weblate.

Monolingual gettext

Some projects decide to use gettext as monolingual formats—they code just the IDs in their source code and the string then needs to be translated to all languages, including English. This is supported, though you have to choose this file format explicitly when importing components into Weblate.

The monolingual gettext PO file typically looks like this:

```
#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "day-monday"
msgstr "Pondělí"

#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "day-tuesday"
msgstr "Úterý"

#: weblate/accounts/avatar.py:163
msgid "none-user"
msgstr "Žádný"
```

While the base language file will be:

```
#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "day-monday"
msgstr "Monday"

#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "day-tuesday"
msgstr "Tuesday"

#: weblate/accounts/avatar.py:163
msgid "none-user"
msgstr "None"
```

Typical Weblate *Component configuration*	
Masca de fișier	`po/*.po`
Fișier de limbă de bază monolingvă	`po/en.po`
Șablon pentru traduceri noi	`po/messages.pot`
Formatul fișierului	*Gettext PO file (monolingual)*

1.10.5 XLIFF

XML-based format created to standardize translation files, but in the end it is one of many standards, in this area.

XML Localization Interchange File Format (XLIFF) is usually used as bilingual, but Weblate supports it as monolingual as well.

Vezi și:

XML Localization Interchange File Format (XLIFF) specification

Translation states

Schimbat în versiunea 3.3: Weblate ignored the `state` attribute prior to the 3.3 release.

The `state` attribute in the file is partially processed and mapped to the „Needs edit" state in Weblate (the following states are used to flag the string as needing edit if there is a target present: `new`, `needs-translation`, `needs-adaptation`, `needs-l10n`). Should the `state` attribute be missing, a string is considered translated as soon as a `<target>` element exists.

If the translation string has `approved="yes"`, it will also be imported into Weblate as „Approved", anything else will be imported as „Waiting for review" (which matches the XLIFF specification).

While saving, Weblate doesn't add those attributes unless necessary:

- The `state` attribute is only added in case string is marked as needing edit.
- The `approved` attribute is only added in case string has been reviewed.
- In other cases the attributes are not added, but they are updated in case they are present.

That means that when using the XLIFF format, it is strongly recommended to turn on the Weblate review process, in order to see and change the approved state of strings.

Similarly upon importing such files (in the upload form), you should choose *Import as translated* under *Processing of strings needing edit*.

Vezi și:

Dedicated reviewers

Whitespace and newlines in XLIFF

Generally types or amounts of whitespace is not differentiated between in XML formats. If you want to keep it, you have to add the `xml:space="preserve"` flag to the string.

De exemplu:

```
    <trans-unit id="10" approved="yes">
        <source xml:space="preserve">hello</source>
        <target xml:space="preserve">Hello, world!
</target>
    </trans-unit>
```

Specifying translation flags

You can specify additional translation flags (see *Personalizarea comportamentului cu ajutorul steguleţelor*) by using the `weblate-flags` attribute. Weblate also understands `maxwidth` and `font` attributes from the XLIFF specification:

```
<trans-unit id="10" maxwidth="100" size-unit="pixel" font="ubuntu;22;bold">
   <source>Hello %s</source>
</trans-unit>
<trans-unit id="20" maxwidth="100" size-unit="char" weblate-flags="c-format">
   <source>Hello %s</source>
</trans-unit>
```

The `font` attribute is parsed for font family, size and weight, the above example shows all of that, though only font family is required. Any whitespace in the font family is converted to underscore, so `Source Sans Pro` becomes `Source_Sans_Pro`, please keep that in mind when naming the font group (see *Gestionarea fonturilor*).

String keys

Weblate identifies the units in the XLIFF file by `resname` attribute in case it is present and falls back to `id` (together with `file` tag if present).

The `resname` attribute is supposed to be human friendly identifier of the unit making it more suitable for Weblate to display instead of `id`. The `resname` has to be unique in the whole XLIFF file. This is required by Weblate and is not covered by the XLIFF standard - it does not put any uniqueness restrictions on this attribute.

Typical Weblate *Component configuration* for bilingual XLIFF	
Masca de fișier	`localizations/*.xliff`
Fișier de limbă de bază monolingvă	*Empty*
Șablon pentru traduceri noi	`localizations/en-US.xliff`
Formatul fișierului	*XLIFF Translation File*

Typical Weblate *Component configuration* for monolingual XLIFF	
File mask	`localizations/*.xliff`
Fișier de limbă de bază monolingvă	`localizations/en-US.xliff`
Șablon pentru traduceri noi	`localizations/en-US.xliff`
Formatul fișierului	*XLIFF Translation File*

Vezi și:

XLIFF on Wikipedia, XLIFF, font attribute in XLIFF 1.2, maxwidth attribute in XLIFF 1.2

1.10.6 Java properties

Native Java format for translations.

Java properties are usually used as monolingual translations.

Weblate supports ISO-8859-1, UTF-8 and UTF-16 variants of this format. All of them support storing all Unicode characters, it is just differently encoded. In the ISO-8859-1, the Unicode escape sequences are used (for example `zkou\u0161ka`), all others encode characters directly either in UTF-8 or UTF-16.

Notă: Loading escape sequences works in UTF-8 mode as well, so please be careful choosing the correct encoding set to match your application needs.

Typical Weblate *Component configuration*	
Masca de fișier	`src/app/Bundle_*.properties`
Fișier de limbă de bază monolingvă	`src/app/Bundle.properties`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*Java Properties (ISO-8859-1)*

Vezi și:

Java properties on Wikipedia, Mozilla and Java properties files, *mi18n lang files*, *Proprietăți GWT*, updating-target-files, *Formatează fișierul de proprietăți Java*, *Curățați fișierele de traducere*

1.10.7 mi18n lang files

Nou în versiunea 4.7.

File format used for JavaScript localization by mi18n. Syntactically it matches *Java properties*.

Typical Weblate *Component configuration*	
Masca de fișier	`*.lang`
Fișier de limbă de bază monolingvă	`en-US.lang`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*mi18n lang file*

Vezi și:

mi18n Mozilla and Java properties files, *Java properties*, updating-target-files, *Formatează fișierul de proprietăți Java*, *Curățați fișierele de traducere*

1.10.8 Proprietăți GWT

Native GWT format for translations.

GWT properties are usually used as monolingual translations.

Typical Weblate *Component configuration*	
Masca de fișier	`src/app/Bundle_*.properties`
Fișier de limbă de bază monolingvă	`src/app/Bundle.properties`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*GWT Properties*

Vezi și:

GWT localization guide, GWT Internationalization Tutorial, Mozilla and Java properties files, updating-target-files, *Formatează fișierul de proprietăți Java*, *Curățați fișierele de traducere*

1.10.9 INI translations

Nou în versiunea 4.1.

INI file format for translations.

INI translations are usually used as monolingual translations.

Typical Weblate *Component configuration*	
Masca de fișier	`language/*.ini`
Fișier de limbă de bază monolingvă	`language/en.ini`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*INI File*

Notă: Weblate only extracts keys from sections within an INI file. In case your INI file lacks sections, you might want to use *Joomla translations* or *Java properties* instead.

Vezi și:

INI Files, *Java properties*, *Joomla translations*, *Inno Setup INI translations*

1.10.10 Inno Setup INI translations

Nou în versiunea 4.1.

Inno Setup INI file format for translations.

Inno Setup INI translations are usually used as monolingual translations.

Notă: The only notable difference to *INI translations* is in supporting `%n` and `%t` placeholders for line break and tab.

Typical Weblate *Component configuration*	
Masca de fișier	`language/*.islu`
Fișier de limbă de bază monolingvă	`language/en.islu`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*Inno Setup INI File*

Notă: Only Unicode files (`.islu`) are currently supported, ANSI variant (`.isl`) is currently not supported.

Vezi și:

INI Files, *Joomla translations*, *INI translations*

1.10.11 Joomla translations

Nou în versiunea 2.12.

Native Joomla format for translations.

Joomla translations are usually used as monolingual translations.

Typical Weblate *Component configuration*	
Masca de fișier	`language/*/com_foobar.ini`
Fișier de limbă de bază monolingvă	`language/en-GB/com_foobar.ini`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*Joomla Language File*

Vezi și:

Specification of Joomla language files, Mozilla and Java properties files, *INI translations*, *Inno Setup INI translations*

1.10.12 Qt Linguist .ts

Translation format used in Qt based applications.

Qt Linguist files are used as both bilingual and monolingual translations.

Typical Weblate *Component configuration* when using as bilingual	
Masca de fișier	`i18n/app.*.ts`
Fișier de limbă de bază monolingvă	*Empty*
Șablon pentru traduceri noi	`i18n/app.de.ts`
Formatul fișierului	*Qt Linguist Translation File*

Typical Weblate *Component configuration* when using as monolingual	
Masca de fișier	`i18n/app.*.ts`
Fișier de limbă de bază monolingvă	`i18n/app.en.ts`
Șablon pentru traduceri noi	`i18n/app.en.ts`
Formatul fișierului	*Qt Linguist Translation File*

Vezi și:

Qt Linguist manual, Qt .ts, *Bilingual and monolingual formats*

1.10.13 Android string resources

Android specific file format for translating applications.

Android string resources are monolingual, the *Fișier de limbă de bază monolingvă* is stored in a different location from the others `res/values/strings.xml`.

Typical Weblate *Component configuration*	
Masca de fișier	`res/values-*/strings.xml`
Fișier de limbă de bază monolingvă	`res/values/strings.xml`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*Android String Resource*

Vezi și:

Android string resources documentation, Android string resources

---

Notă: Android *string-array* structures are not currently supported. To work around this, you can break your string arrays apart:

```
<string-array name="several_strings">
    <item>First string</item>
    <item>Second string</item>
</string-array>
```

become:

```
<string-array name="several_strings">
    <item>@string/several_strings_0</item>
    <item>@string/several_strings_1</item>
</string-array>
<string name="several_strings_0">First string</string>
<string name="several_strings_1">Second string</string>
```

The *string-array* that points to the *string* elements should be stored in a different file, and not be made available for translation.

This script may help pre-process your existing strings.xml files and translations: https://gist.github.com/paour/11291062

1.10.14 Apple iOS strings

Apple specific file format for translating applications, used for both iOS and iPhone/iPad application translations.

Apple iOS strings are usually used as bilingual translations.

Typical Weblate *Component configuration*	
Masca de fișier	`Resources/*.lproj/Localizable.strings`
Fișier de limbă de bază monolingvă	`Resources/en.lproj/Localizable.strings` or `Resources/Base.lproj/Localizable.strings`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*iOS Strings (UTF-8)*

Vezi și:

Stringsdict format, Apple „strings files” documentation, Mac OSX strings

1.10.15 Șiruri PHP

PHP translations are usually monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

Example file:

```
<?php
$LANG['foo'] = 'bar';
$LANG['foo1'] = 'foo bar';
$LANG['foo2'] = 'foo bar baz';
$LANG['foo3'] = 'foo bar baz bag';
```

Typical Weblate *Component configuration*	
Masca de fișier	`lang/*/texts.php`
Fișier de limbă de bază monolingvă	`lang/en/texts.php`
Șablon pentru traduceri noi	`lang/en/texts.php`
Formatul fișierului	*PHP strings*

Laravel PHP șiruri de caractere

Schimbat în versiunea 4.1.

The Laravel PHP localization files are supported as well with plurals:

```
<?php
return [
    'welcome' => 'Welcome to our application',
    'apples' => 'There is one apple|There are many apples',
];
```

Vezi și:

PHP, Localization in Laravel

1.10.16 JSON files

Nou în versiunea 2.0.

Schimbat în versiunea 2.16: Since Weblate 2.16 and with translate-toolkit at-least 2.2.4, nested structure JSON files are supported as well.

Schimbat în versiunea 4.3: The structure of JSON file is properly preserved even for complex situations which were broken in prior releases.

JSON format is used mostly for translating applications implemented in JavaScript.

Weblate currently supports several variants of JSON translations:

- Simple key / value files, used for example by *vue-i18n* or *react-intl*.
- Files with nested keys.
- *JSON i18next files*
- *go-i18n JSON files*
- *WebExtension JSON*
- *ARB File*

JSON translations are usually monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

Example file:

```
{
  "Hello, world!\n": "Ahoj světe!\n",
  "Orangutan has %d banana.\n": "",
  "Try Weblate at https://demo.weblate.org/!\n": "",
  "Thank you for using Weblate.": ""
}
```

Nested files are supported as well (see above for requirements), such a file can look like:

```
{
  "weblate": {
    "hello": "Ahoj světe!\n",
    "orangutan": "",
    "try": "",
    "thanks": ""
  }
}
```

Sugestie: The *JSON file* and *JSON nested structure file* can both handle same type of files. Both preserve existing JSON structure when translating.

The only difference between them is when adding new strings using Weblate. The nested structure format parses the newly added key and inserts the new string into the matching structure. For example `app.name` key is inserted as:

```
{
   "app": {
      "name": "Weblate"
   }
}
```

Typical Weblate *Component configuration*	
Masca de fișier	`langs/translation-*.json`
Fișier de limbă de bază monolingvă	`langs/translation-en.json`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*JSON nested structure file*

Vezi și:

JSON, updating-target-files, *Personalizați ieșirea JSON*, *Curățați fișierele de traducere*,

1.10.17 JSON i18next files

Schimbat în versiunea 2.17: Since Weblate 2.17 and with translate-toolkit at-least 2.2.5, i18next JSON files with plurals are supported as well.

i18next is an internationalization framework written in and for JavaScript. Weblate supports its localization files with features such as plurals.

i18next translations are monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

Notă: Weblate supports the i18next JSON v3 format. The v2 and v1 variants are mostly compatible, with exception of how plurals are handled.

Example file:

```
{
  "hello": "Hello",
  "apple": "I have an apple",
  "apple_plural": "I have {{count}} apples",
  "apple_negative": "I have no apples"
}
```

Typical Weblate *Component configuration*	
Masca de fișier	`langs/*.json`
Fișier de limbă de bază monolingvă	`langs/en.json`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*i18next JSON file*

Vezi și:

JSON, i18next JSON Format, updating-target-files, *Personalizați ieșirea JSON*, *Curățați fișierele de traducere*

1.10.18 go-i18n JSON files

Nou în versiunea 4.1.

go-i18n translations are monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

Notă: Weblate supports the go-i18n JSON v1 format, for flat JSON formats please use *JSON files*. The v2 format with hash is currently not supported.

Typical Weblate *Component configuration*	
Masca de fișier	`langs/*.json`
Fișier de limbă de bază monolingvă	`langs/en.json`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*go-i18n JSON file*

Vezi și:

JSON, go-i18n, updating-target-files, *Personalizați ieșirea JSON*, *Curățați fișierele de traducere*,

1.10.19 ARB File

Nou în versiunea 4.1.

ARB translations are monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

Typical Weblate *Component configuration*	
Masca de fișier	`lib/l10n/intl_*.arb`
Fișier de limbă de bază monolingvă	`lib/l10n/intl_en.arb`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*ARB file*

Vezi și:

JSON, Application Resource Bundle Specification, Internationalizing Flutter apps, updating-target-files, *Personalizați ieșirea JSON*, *Curățați fișierele de traducere*

1.10.20 WebExtension JSON

Nou în versiunea 2.16: This is supported since Weblate 2.16 and with translate-toolkit at-least 2.2.4.

File format used when translating extensions for Mozilla Firefox or Google Chromium.

Notă: While this format is called JSON, its specification allows to include comments, which are not part of JSON specification. Weblate currently does not support file with comments.

Example file:

```
{
  "hello": {
    "message": "Ahoj světe!\n",
    "description": "Description",
    "placeholders": {
      "url": {
        "content": "$1",
        "example": "https://developer.mozilla.org"
      }
    }
  },
  "orangutan": {
    "message": "",
    "description": "Description"
  },
  "try": {
    "message": "",
```

(continues on next page)

(continuare din pagina precedentă)

```
    "description": "Description"
  },
  "thanks": {
    "message": "",
    "description": "Description"
  }
}
```

Typical Weblate *Component configuration*	
Masca de fișier	`_locales/*/messages.json`
Fișier de limbă de bază monolingvă	`_locales/en/messages.json`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*WebExtension JSON file*

Vezi și:

JSON, Google chrome.i18n, Mozilla Extensions Internationalization

1.10.21 .XML resource files

Nou în versiunea 2.3.

A .XML resource (.resx) file employs a monolingual XML file format used in Microsoft .NET applications. It is interchangeable with .resw, when using identical syntax to .resx.

Typical Weblate *Component configuration*	
Masca de fișier	`Resources/Language.*.resx`
Fișier de limbă de bază monolingvă	`Resources/Language.resx`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*.NET resource file*

Vezi și:

.NET Resource files (.resx), updating-target-files, *Curățați fișierele de traducere*

1.10.22 CSV files

Nou în versiunea 2.4.

CSV files can contain a simple list of source and translation. Weblate supports the following files:

- Files with header defining fields (`location`, `source`, `target`, `ID`, `fuzzy`, `context`, `translator_comments`, `developer_comments`). This is the recommended approach, as it is the least error prone. Choose *CSV file* as a file format.
- Files with two fields—source and translation (in this order). Choose *Simple CSV file* as a file format.
- Headerless files with fields in order defined by the translate-toolkit: `location`, `source`, `target`, `ID`, `fuzzy`, `context`, `translator_comments`, `developer_comments`. Choose *CSV file* as a file format.
- Remember to define *Fișier de limbă de bază monolingvă* when your files are monolingual (see *Bilingual and monolingual formats*).

Atenționare: The CSV format currently automatically detects the dialect of the CSV file. In some cases the automatic detection might fail and you will get mixed results. This is especially true for CSV files with newlines in the values. As a workaround it is recommended to omit quoting characters.

Example file:

```
Thank you for using Weblate.,Děkujeme za použití Weblate.
```

Typical Weblate *Component configuration* for bilingual CSV	
Masca de fișier	`locale/*.csv`
Fișier de limbă de bază monolingvă	*Empty*
Șablon pentru traduceri noi	`locale/en.csv`
Formatul fișierului	*CSV file*

Typical Weblate *Component configuration* for monolingual CSV	
Masca de fișier	`locale/*.csv`
Fișier de limbă de bază monolingvă	`locale/en.csv`
Șablon pentru traduceri noi	`locale/en.csv`
Formatul fișierului	*Simple CSV file*

Vezi și:

CSV

1.10.23 YAML files

Nou în versiunea 2.9.

The plain YAML files with string keys and values. Weblate also extract strings from lists or dictionaries.

Example of a YAML file:

```yaml
weblate:
  hello: ""
  orangutan": ""
  try": ""
  thanks": ""
```

Typical Weblate *Component configuration*	
Masca de fișier	`translations/messages.*.yml`
Fișier de limbă de bază monolingvă	`translations/messages.en.yml`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*YAML file*

Vezi și:

YAML, *Ruby YAML files*

1.10.24 Ruby YAML files

Nou în versiunea 2.9.

Ruby i18n YAML files with language as root node.

Example Ruby i18n YAML file:

```yaml
cs:
  weblate:
    hello: ""
    orangutan: ""
```

(continues on next page)

(continuare din pagina precedentă)

```
    try: ""
    thanks: ""
```

Typical Weblate *Component configuration*	
Masca de fișier	`translations/messages.*.yml`
Fișier de limbă de bază monolingvă	`translations/messages.en.yml`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*Ruby YAML file*

Vezi și:

YAML, *YAML files*

1.10.25 DTD files

Nou în versiunea 2.18.

Example DTD file:

```
<!ENTITY hello "">
<!ENTITY orangutan "">
<!ENTITY try "">
<!ENTITY thanks "">
```

Typical Weblate *Component configuration*	
Masca de fișier	`locale/*.dtd`
Fișier de limbă de bază monolingvă	`locale/en.dtd`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*DTD file*

Vezi și:

Mozilla DTD format

1.10.26 Flat XML files

Nou în versiunea 3.9.

Example of a flat XML file:

```
<?xml version='1.0' encoding='UTF-8'?>
<root>
  <str key="hello_world">Hello World!</str>
  <str key="resource_key">Translated value.</str>
</root>
```

Typical Weblate *Component configuration*	
Masca de fișier	`locale/*.xml`
Fișier de limbă de bază monolingvă	`locale/en.xml`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*Flat XML file*

Vezi și:

Flat XML

1.10.27 Windows RC files

Schimbat în versiunea 4.1: Support for Windows RC files has been rewritten.

Notă: Support for this format is currently in beta, feedback from testing is welcome.

Example Windows RC file:

```
LANGUAGE LANG_CZECH, SUBLANG_DEFAULT

STRINGTABLE
BEGIN
    IDS_MSG1                "Hello, world!\n"
    IDS_MSG2                "Orangutan has %d banana.\n"
    IDS_MSG3                "Try Weblate at http://demo.weblate.org/!\n"
    IDS_MSG4                "Thank you for using Weblate."
END
```

Typical Weblate *Component configuration*	
Masca de fișier	`lang/*.rc`
Fișier de limbă de bază monolingvă	`lang/en-US.rc`
Șablon pentru traduceri noi	`lang/en-US.rc`
Formatul fișierului	*RC file*

Vezi și:

Windows RC files

1.10.28 Fișiere de metadate ale magazinului de aplicații

Nou în versiunea 3.5.

Metadata used for publishing apps in various app stores can be translated. Currently the following tools are compatible:

- Triple-T gradle-play-publisher
- Fastlane
- F-Droid

The metadata consists of several textfiles, which Weblate will present as separate strings to translate.

Typical Weblate *Component configuration*	
Masca de fișier	`fastlane/android/metadata/*`
Fișier de limbă de bază monolingvă	`fastlane/android/metadata/en-US`
Șablon pentru traduceri noi	`fastlane/android/metadata/en-US`
Formatul fișierului	*App store metadata files*

Sugestie: In case you don't want to translate certain strings (for example changelogs), mark them read-only (see *Personalizarea comportamentului cu ajutorul steguleților*). This can be automated by the *Editare în masă*.

1.10.29 Subtitle files

Nou în versiunea 3.7.

Weblate can translate various subtitle files:

- SubRip subtitle file (`*.srt`)
- MicroDVD subtitle file (`*.sub`)
- Advanced Substation Alpha subtitles file (`*.ass`)
- Substation Alpha subtitle file (`*.ssa`)

Typical Weblate *Component configuration*	
Masca de fișier	`path/*.srt`
Fișier de limbă de bază monolingvă	`path/en.srt`
Șablon pentru traduceri noi	`path/en.srt`
Formatul fișierului	*SubRip subtitle file*

Vezi și:

Subtitles

1.10.30 Excel Deschidere XML

Nou în versiunea 3.2.

Excel Open XML (.xlsx) files can be imported and exported.

When uploading XLSX files for translation, be aware that only the active worksheet is considered, and there must be at least a column called `source` (which contains the source string) and a column called `target` (which contains the translation). Additionally there should be the column called `context` (which contains the context path of the translation string). If you use the XLSX download for exporting the translations into an Excel workbook, you already get a file with the correct file format.

1.10.31 HTML files

Nou în versiunea 4.1.

Notă: Support for this format is currently in beta, feedback from testing is welcome.

The translatable content is extracted from the HTML files and offered for the translation.

Vezi și:

HTML

1.10.32 Text files

Nou în versiunea 4.6.

Notă: Support for this format is currently in beta, feedback from testing is welcome.

The translatable content is extracted from the plain text files and offered for the translation. Each paragraph is translated as a separate string.

There are three flavors of this format:

- Fișier text simplu
- Fișier text DokuWiki
- Fișier text MediaWiki

Vezi și:

Simple Text Documents

1.10.33 OpenDocument Format

Nou în versiunea 4.1.

Notă: Support for this format is currently in beta, feedback from testing is welcome.

The translatable content is extracted from the OpenDocument files and offered for the translation.

Vezi și:

OpenDocument Format

1.10.34 IDML Format

Nou în versiunea 4.1.

Notă: Support for this format is currently in beta, feedback from testing is welcome.

The translatable content is extracted from the Adobe InDesign Markup Language files and offered for the translation.

1.10.35 TermBase eXchange format

Nou în versiunea 4.5.

TBX is an XML format for the exchange of terminology data.

Typical Weblate *Component configuration*	
Masca de fișier	`tbx/*.tbx`
Fișier de limbă de bază monolingvă	*Empty*
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*TermBase eXchange file*

Vezi și:

TBX on Wikipedia, TBX, *Glosar*

1.10.36 Stringsdict format

Nou în versiunea 4.8.

Notă: Support for this format is currently in beta, feedback from testing is welcome.

XML based format used by Apple which is able to store plural forms of a string.

Typical Weblate *Component configuration*	
Masca de fișier	`Resources/*.lproj/Localizable.stringsdict`
Fișier de limbă de bază monolingvă	`Resources/en.lproj/Localizable.stringsdict` or `Resources/Base.lproj/Localizable.stringsdict`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*Stringsdict file*

Vezi și:

Apple iOS strings, Stringsdict File Format,

1.10.37 Fluent format

Nou în versiunea 4.8.

Notă: Support for this format is currently in beta, feedback from testing is welcome.

Fluent is a monolingual text format that focuses on asymmetric localization: a simple string in one language can map to a complex multi-variant translation in another language.

Typical Weblate *Component configuration*	
Masca de fișier	`locales/*/messages.ftl`
Fișier de limbă de bază monolingvă	`locales/en/messages.ftl`
Șablon pentru traduceri noi	*Empty*
Formatul fișierului	*Fluent file*

Vezi și:

Project Fluent website

1.10.38 Supporting other formats

Most formats supported by translate-toolkit which support serializing can be easily supported, but they did not (yet) receive any testing. In most cases some thin layer is needed in Weblate to hide differences in behavior of different translate-toolkit storages.

To add support for a new format, the preferred approach is to first implement support for it in the translate-toolkit.

Vezi și:

Translation Related File Formats

1.11 Integrarea controlului versiunilor

Weblate currently supports *Git* (with extended support for *GitHub*, *Gerrit* and *Subversion*) and *Mercurial* as version control back-ends.

1.11.1 Accessing repositories

The VCS repository you want to use has to be accessible to Weblate. With a publicly available repository you just need to enter the correct URL (for example `https://github.com/WeblateOrg/weblate.git`), but for private repositories or for push URLs the setup is more complex and requires authentication.

Accessing repositories from Hosted Weblate

For Hosted Weblate there is a dedicated push user registered on GitHub, Bitbucket, Codeberg and GitLab (with the username *weblate*, e-mail `hosted@weblate.org` and, named *Weblate push user*). You need to add this user as a collaborator and give it appropriate permission to your repository (read-only is okay for cloning, write is required for pushing). Depending on service and your organization settings, this happens immediately, or requires confirmation on the Weblate side.

The *weblate* user on GitHub accepts invitations automatically within five minutes. Manual processing might be needed on the other services, so please be patient.

Once the *weblate* user is added, you can configure *Depozitul de cod sursă* and *URL de împingere a depozitului* using the SSH protocol (for example `git@github.com:WeblateOrg/weblate.git`).

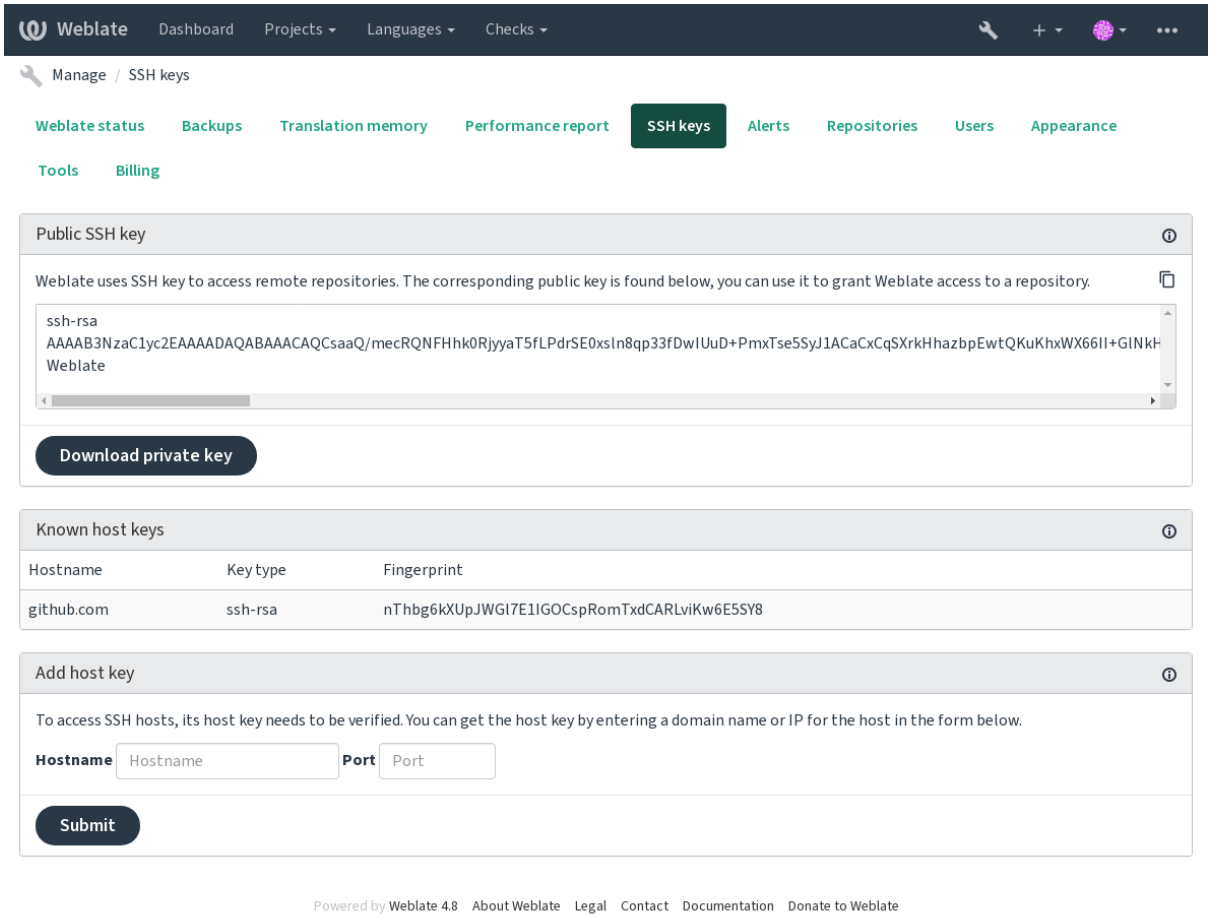
SSH repositories

The most frequently used method to access private repositories is based on SSH. Authorize the public Weblate SSH key (see *Weblate SSH key*) to access the upstream repository this way.

> **Atenționare:** On GitHub, each key can only be used once, see *GitHub repositories* and *Accessing repositories from Hosted Weblate*.

Weblate also stores the host key fingerprint upon first connection, and fails to connect to the host should it be changed later (see *Verifying SSH host keys*).

In case adjustment is needed, do so from the Weblate admin interface:



Weblate SSH key

The Weblate public key is visible to all users browsing the *About* page.

Admins can generate or display the public key currently used by Weblate in the connection (from *SSH keys*) on the admin interface landing page.

Notă: The corresponding private SSH key can not currently have a password, so make sure it is well protected.

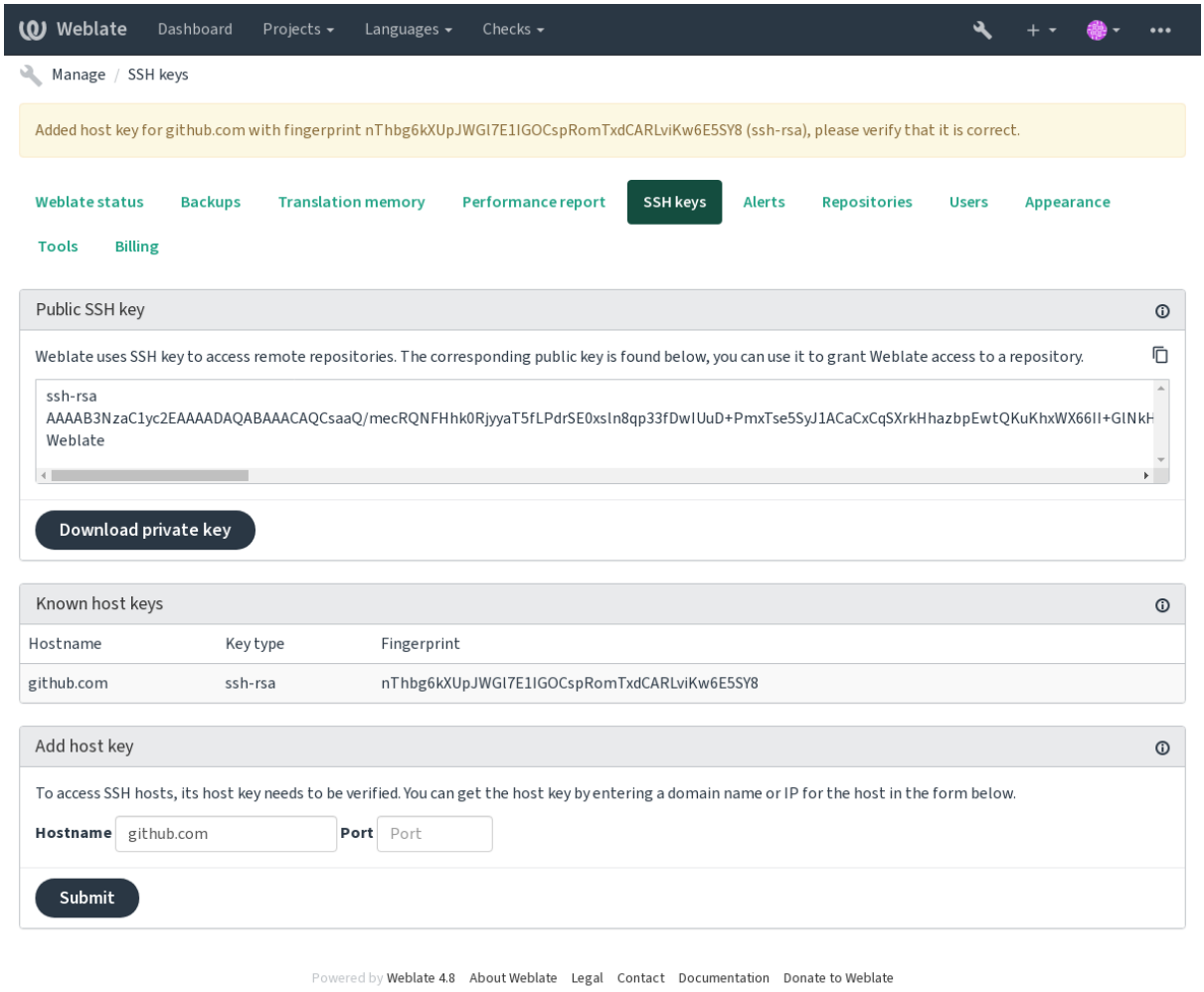
Sugestie: Make a backup of the generated private Weblate SSH key.

Verifying SSH host keys

Weblate automatically stores the SSH host keys on first access and remembers them for further use.

In case you want to verify the key fingerprint before connecting to the repository, add the SSH host keys of the servers you are going to access in *Add host key*, from the same section of the admin interface. Enter the hostname you are going to access (e.g. `gitlab.com`), and press *Submit*. Verify its fingerprint matches the server you added.

The added keys with fingerprints are shown in the confirmation message:



GitHub repositories

Access via SSH is possible (see *SSH repositories*), but in case you need to access more than one repository, you will hit a GitHub limitation on allowed SSH key usage (since each key can be used only once).

In case the *Împingeți ramura* is not set, the project is forked and changes pushed through a fork. In case it is set, changes are pushed to the upstream repository and chosen branch.

For smaller deployments, use HTTPS authentication with a personal access token and your GitHub account, see Creating an access token for command-line use.

For bigger setups, it is usually better to create a dedicated user for Weblate, assign it the public SSH key generated in Weblate (see *Weblate SSH key*) and grant it access to all the repositories you want to translate. This approach is also used for Hosted Weblate, there is dedicated *weblate* user for that.

Vezi și:

Accessing repositories from Hosted Weblate

Weblate internal URLs

Share one repository setup between different components by referring to its placement as `weblate://project/component` in other(linked) components. This way linked components use the VCS repository configuration of the main(referenced) component.

Atenționare: Removing main component also removes linked components.

Weblate automatically adjusts the repository URL when creating a component if it finds a component with a matching repository setup. You can override this in the last step of the component configuration.

Reasons to use this:

- Saves disk space on the server, the repository is stored just once.
- Makes the updates faster, only one repository is updated.
- There is just single exported repository with Weblate translations (see *Git exporter*).
- Some addons can operate on multiple components sharing one repository, for example *Comenzi Git Squash*.

HTTPS repositories

To access protected HTTPS repositories, include the username and password in the URL. Don't worry, Weblate will strip this info when the URL is shown to users (if even allowed to see the repository URL at all).

For example the GitHub URL with authentication added might look like: `https://user:your_access_token@github.com/WeblateOrg/weblate.git`.

Notă: If your username or password contains special characters, those have to be URL encoded, for example `https://user%40example.com:%24password%23@bitbucket.org/`....

Using proxy

If you need to access HTTP/HTTPS VCS repositories using a proxy server, configure the VCS to use it.

This can be done using the `http_proxy`, `https_proxy`, and `all_proxy` environment variables, (as described in the cURL documentation) or by enforcing it in the VCS configuration, for example:

```
git config --global http.proxy http://user:password@proxy.example.com:80
```

Notă: The proxy configuration needs to be done under user running Weblate (see also *Filesystem permissions*) and with `HOME=$DATA_DIR/home` (see *`DATA_DIR`*), otherwise Git executed by Weblate will not use it.

Vezi și:

The cURL manpage, Git config documentation

1.11.2 Git

Sugestie: Weblate needs Git 2.12 or newer.

Vezi și:

See *Accessing repositories* for info on how to access different kinds of repositories.

Git cu forța push

This behaves exactly like Git itself, the only difference being that it always force pushes. This is intended only in the case of using a separate repository for translations.

Atenționare: Use with caution, as this easily leads to lost commits in your upstream repository.

Customizing Git configuration

Weblate invokes all VCS commands with `HOME=$DATA_DIR/home` (see *DATA_DIR*), therefore editing the user configuration needs to be done in `DATA_DIR/home/.git`.

Git remote helpers

You can also use Git remote helpers for additionally supporting other version control systems, but be prepared to debug problems this may lead to.

At this time, helpers for Bazaar and Mercurial are available within separate repositories on GitHub: git-remote-hg and git-remote-bzr. Download them manually and put somewhere in your search path (for example `~/bin`). Make sure you have the corresponding version control systems installed.

Once you have these installed, such remotes can be used to specify a repository in Weblate.

To clone the `gnuhello` project from Launchpad using Bazaar:

```
bzr::lp:gnuhello
```

For the `hello` repository from selenic.com using Mercurial:

```
hg::http://selenic.com/repo/hello
```

Atenționare: The inconvenience of using Git remote helpers is for example with Mercurial, the remote helper sometimes creates a new tip when pushing changes back.

1.11.3 GitHub

Nou în versiunea 2.3.

This adds a thin layer atop *Git* using the GitHub API to allow pushing translation changes as pull requests, instead of pushing directly to the repository.

Git pushes changes directly to a repository, while *GitHub* creates pull requests. The latter is not needed for merely accessing Git repositories.

Vezi și:

Pushing changes from Weblate

Pushing changes to GitHub as pull requests

If not wanting to push translations to a GitHub repository, they can be sent as either one or many pull requests instead.

You need to configure API credentials to make this work.

Vezi și:

`GITHUB_USERNAME`, `GITHUB_TOKEN`, `GITHUB_CREDENTIALS`

1.11.4 GitLab

Nou în versiunea 3.9.

This just adds a thin layer atop *Git* using the GitLab API to allow pushing translation changes as merge requests instead of pushing directly to the repository.

There is no need to use this to access Git repositories, ordinary *Git* works the same, the only difference is how pushing to a repository is handled. With *Git* changes are pushed directly to the repository, while *GitLab* creates merge request.

Vezi și:

Pushing changes from Weblate

Pushing changes to GitLab as merge requests

If not wanting to push translations to a GitLab repository, they can be sent as either one or many merge requests instead.

You need to configure API credentials to make this work.

Vezi și:

`GITLAB_USERNAME`, `GITLAB_TOKEN`, `GITLAB_CREDENTIALS`

1.11.5 Pagure

Nou în versiunea 4.3.2.

This just adds a thin layer atop *Git* using the Pagure API to allow pushing translation changes as merge requests instead of pushing directly to the repository.

There is no need to use this to access Git repositories, ordinary *Git* works the same, the only difference is how pushing to a repository is handled. With *Git* changes are pushed directly to the repository, while *Pagure* creates merge request.

Vezi și:

Pushing changes from Weblate

Pushing changes to Pagure as merge requests

If not wanting to push translations to a Pagure repository, they can be sent as either one or many merge requests instead.

You need to configure API credentials to make this work.

Vezi și:

`PAGURE_USERNAME`, `PAGURE_TOKEN`, `PAGURE_CREDENTIALS`

1.11.6 Gerrit

Nou în versiunea 2.2.

Adds a thin layer atop *Git* using the git-review tool to allow pushing translation changes as Gerrit review requests, instead of pushing them directly to the repository.

The Gerrit documentation has the details on the configuration necessary to set up such repositories.

1.11.7 Mercurial

Nou în versiunea 2.1.

Mercurial is another VCS you can use directly in Weblate.

Notă: It should work with any Mercurial version, but there are sometimes incompatible changes to the command-line interface which breaks Weblate integration.

Vezi și:

See *Accessing repositories* for info on how to access different kinds of repositories.

1.11.8 Subversion

Nou în versiunea 2.8.

Weblate uses git-svn to interact with subversion repositories. It is a Perl script that lets subversion be used by a Git client, enabling users to maintain a full clone of the internal repository and commit locally.

Notă: Weblate tries to detect Subversion repository layout automatically - it supports both direct URLs for branch or repositories with standard layout (branches/, tags/ and trunk/). More info about this is to be found in the git-svn documentation. If your repository does not have a standard layout and you encounter errors, try including the branch name in the repository URL and leaving branch empty.

Schimbat în versiunea 2.19: Before this, only repositories using the standard layout were supported.

Subversion credentials

Weblate expects you to have accepted the certificate up-front (and your credentials if needed). It will look to insert them into the `DATA_DIR` directory. Accept the certificate by using *svn* once with the *$HOME* environment variable set to the `DATA_DIR`:

```
# Use DATA_DIR as configured in Weblate settings.py, it is /app/data in the Docker
HOME=${DATA_DIR}/home svn co https://svn.example.com/example
```

Vezi și:

`DATA_DIR`

1.11.9 Local files

1.11.10 Git

Sugestie: Underneath, this uses *Git*. It requires Git installed and allows you to switch to using Git natively with full history of your translations.

Nou în versiunea 3.8.

Weblate can also operate without a remote VCS. The initial translations are imported by uploading them. Later you can replace individual files by file upload, or add translation strings directly from Weblate (currently available only for monolingual translations).

In the background Weblate creates a Git repository for you and all changes are tracked in. In case you later decide to use a VCS to store the translations, you already have a repository within Weblate can base your integration on.

1.12 Weblate's REST API

Nou în versiunea 2.6: The REST API is available since Weblate 2.6.

The API is accessible on the `/api/` URL and it is based on Django REST framework. You can use it directly or by *Weblate Client*.

1.12.1 Authentication and generic parameters

The public project API is available without authentication, though unauthenticated requests are heavily throttled (by default to 100 requests per day), so it is recommended to use authentication. The authentication uses a token, which you can get in your profile. Use it in the `Authorization` header:

ANY /

Generic request behaviour for the API, the headers, status codes and parameters here apply to all endpoints as well.

Query Parameters

- **`format`** – Response format (overrides Accept). Possible values depends on REST framework setup, by default `json` and `api` are supported. The latter provides web browser interface for API.
- **`page`** – Returns given page of paginated results (use *next* and *previous* fields in response to automate the navigation).

Request Headers

- Accept – the response content type depends on Accept header
- Authorization – optional token to authenticate as `Authorization: Token YOUR-TOKEN`

Response Headers

- Content-Type – this depends on Accept header of request
- Allow – list of allowed HTTP methods on object

Response JSON Object

- **`detail`** (*`string`*) – verbose description of the result (for HTTP status codes other than 200 OK)
- **`count`** (*`int`*) – total item count for object lists

- **next** (*string*) – next page URL for object lists
- **previous** (*string*) – previous page URL for object lists
- **results** (*array*) – results for object lists
- **url** (*string*) – URL to access this resource using API
- **web_url** (*string*) – URL to access this resource using web browser

Status Codes

- 200 OK – when request was correctly handled
- 201 Created – when a new object was created successfully
- 204 No Content – when an object was deleted successfully
- 400 Bad Request – when form parameters are missing
- 403 Forbidden – when access is denied
- 429 Too Many Requests – when throttling is in place

Authentication examples

Example request:

```
GET /api/ HTTP/1.1
Host: example.com
Accept: application/json, text/javascript
Authorization: Token YOUR-TOKEN
```

Example response:

```
HTTP/1.0 200 OK
Date: Fri, 25 Mar 2016 09:46:12 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, HEAD, OPTIONS

{
    "projects":"http://example.com/api/projects/",
    "components":"http://example.com/api/components/",
    "translations":"http://example.com/api/translations/",
    "languages":"http://example.com/api/languages/"
}
```

CURL example:

```
curl \
    -H "Authorization: Token TOKEN" \
    https://example.com/api/
```

Passing Parameters Examples

For the POST method the parameters can be specified either as form submission (*application/x-www-form-urlencoded*) or as JSON (*application/json*).

Form request example:

```
POST /api/projects/hello/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Token TOKEN

operation=pull
```

JSON request example:

```
POST /api/projects/hello/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"operation":"pull"}
```

CURL example:

```
curl \
    -d operation=pull \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/components/hello/weblate/repository/
```

CURL JSON example:

```
curl \
    --data-binary '{"operation":"pull"}' \
    -H "Content-Type: application/json" \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/components/hello/weblate/repository/
```

API rate limiting

The API requests are rate limited; the default configuration limits it to 100 requests per day for anonymous users and 5000 requests per hour for authenticated users.

Rate limiting can be adjusted in the `settings.py`; see Throttling in Django REST framework documentation for more details how to configure it.

The status of rate limiting is reported in following headers:

`X-RateLimit-Limit`	Rate limiting limit of requests to perform
`X-RateLimit-Remaining`	Remaining limit of requests
`X-RateLimit-Reset`	Number of seconds until ratelimit window resets

Schimbat în versiunea 4.1: Added ratelimiting status headers.

Vezi și:

Rate limiting, *Rate limiting*

1.12.2 API Entry Point

GET /api/

The API root entry point.

Example request:

```
GET /api/ HTTP/1.1
Host: example.com
Accept: application/json, text/javascript
Authorization: Token YOUR-TOKEN
```

Example response:

```
HTTP/1.0 200 OK
Date: Fri, 25 Mar 2016 09:46:12 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, HEAD, OPTIONS

{
    "projects":"http://example.com/api/projects/",
    "components":"http://example.com/api/components/",
    "translations":"http://example.com/api/translations/",
    "languages":"http://example.com/api/languages/"
}
```

1.12.3 Utilizatori

Nou în versiunea 4.0.

GET /api/users/

Returns a list of users if you have permissions to see manage users. If not, then you get to see only your own details.

Vezi și:

Users object attributes are documented at *GET /api/users/(str:username)/*.

POST /api/users/

Creates a new user.

Parameters

- **username** (*string*) – Nume utilizator
- **full_name** (*string*) – User full name
- **email** (*string*) – User email
- **is_superuser** (*boolean*) – Is user superuser? (optional)
- **is_active** (*boolean*) – Is user active? (optional)

GET /api/users/(str: *username*)/

Returns information about users.

Parameters

- **username** (*string*) – User's username

Response JSON Object

- **username** (*string*) – username of a user
- **full_name** (*string*) – full name of a user
- **email** (*string*) – email of a user
- **is_superuser** (*boolean*) – whether the user is a super user
- **is_active** (*boolean*) – whether the user is active
- **date_joined** (*string*) – date the user is created
- **groups** (*array*) – link to associated groups; see *GET /api/groups/(int:id)/*

Example JSON data:

```
{
    "email": "user@example.com",
    "full_name": "Example User",
    "username": "exampleusername",
    "groups": [
        "http://example.com/api/groups/2/",
        "http://example.com/api/groups/3/"
    ],
    "is_superuser": true,
    "is_active": true,
    "date_joined": "2020-03-29T18:42:42.617681Z",
    "url": "http://example.com/api/users/exampleusername/",
    "statistics_url": "http://example.com/api/users/exampleusername/statistics/
↪"
}
```

PUT /api/users/(str: *username*)/

Changes the user parameters.

Parameters

- **username** (*string*) – User's username

Response JSON Object

- **username** (*string*) – username of a user
- **full_name** (*string*) – full name of a user
- **email** (*string*) – email of a user
- **is_superuser** (*boolean*) – whether the user is a super user
- **is_active** (*boolean*) – whether the user is active
- **date_joined** (*string*) – date the user is created

PATCH /api/users/(str: *username*)/

Changes the user parameters.

Parameters

- **username** (*string*) – User's username

Response JSON Object

- **username** (*string*) – username of a user
- **full_name** (*string*) – full name of a user
- **email** (*string*) – email of a user
- **is_superuser** (*boolean*) – whether the user is a super user
- **is_active** (*boolean*) – whether the user is active
- **date_joined** (*string*) – date the user is created

DELETE /api/users/(**str:** *username*)**/**
Deletes all user information and marks the user inactive.

Parameters

- **username** (*string*) – User's username

POST /api/users/(**str:** *username*)**/groups/**
Associate groups with a user.

Parameters

- **username** (*string*) – User's username

Form Parameters

- **string group_id** – The unique group ID

GET /api/users/(**str:** *username*)**/statistics/**
List statistics of a user.

Parameters

- **username** (*string*) – User's username

Response JSON Object

- **translated** (*int*) – Number of translations by user
- **suggested** (*int*) – Number of suggestions by user
- **uploaded** (*int*) – Number of uploads by user
- **commented** (*int*) – Number of comments by user
- **languages** (*int*) – Number of languages user can translate

GET /api/users/(**str:** *username*)**/notifications/**
List subscriptions of a user.

Parameters

- **username** (*string*) – User's username

POST /api/users/(**str:** *username*)**/notifications/**
Associate subscriptions with a user.

Parameters

- **username** (*string*) – User's username

Request JSON Object

- **notification** (*string*) – Name of notification registered
- **scope** (*int*) – Scope of notification from the available choices
- **frequency** (*int*) – Frequency choices for notifications

GET /api/users/(**str:** *username*)**/notifications/**
int: *subscription_id***/** Get a subscription associated with a user.

Parameters

- **username** (*string*) – User's username
- **subscription_id** (*int*) – ID of notification registered

PUT /api/users/(**str:** *username*)**/notifications/**
int: *subscription_id***/** Edit a subscription associated with a user.

Parameters

- **username** (*string*) – User's username

- **subscription_id** (*int*) – ID of notification registered

Request JSON Object

- **notification** (*string*) – Name of notification registered
- **scope** (*int*) – Scope of notification from the available choices
- **frequency** (*int*) – Frequency choices for notifications

PATCH /api/users/(**str:** *username*)**/notifications/**
int: *subscription_id***/** Edit a subscription associated with a user.

Parameters

- **username** (*string*) – User's username
- **subscription_id** (*int*) – ID of notification registered

Request JSON Object

- **notification** (*string*) – Name of notification registered
- **scope** (*int*) – Scope of notification from the available choices
- **frequency** (*int*) – Frequency choices for notifications

DELETE /api/users/(**str:** *username*)**/notifications/**
int: *subscription_id***/** Delete a subscription associated with a user.

Parameters

- **username** (*string*) – User's username
- **subscription_id** – Name of notification registered
- **subscription_id** – int

1.12.4 Grupuri

Nou în versiunea 4.0.

GET /api/groups/
Returns a list of groups if you have permissions to see manage groups. If not, then you get to see only the groups the user is a part of.

Vezi și:

Group object attributes are documented at *GET /api/groups/(int:id)/*.

POST /api/groups/
Creates a new group.

Parameters

- **name** (*string*) – Numele grupului
- **project_selection** (*int*) – Group of project selection from given options
- **language_selection** (*int*) – Group of languages selected from given options

GET /api/groups/(**int:** *id*)**/**
Returns information about group.

Parameters

- **id** (*int*) – Group's ID

Response JSON Object

- **name** (*string*) – name of a group
- **project_selection** (*int*) – integer corresponding to group of projects

- **language_selection** (*int*) – integer corresponding to group of languages
- **roles** (*array*) – link to associated roles; see *GET /api/roles/(int:id)/*
- **projects** (*array*) – link to associated projects; see *GET /api/projects/(string:project)/*
- **components** (*array*) – link to associated components; see *GET /api/components/(string:project)/(string:component)/*
- **componentlist** (*array*) – link to associated componentlist; see *GET /api/component-lists/(str:slug)/*

Example JSON data:

```
{
    "name": "Guests",
    "project_selection": 3,
    "language_selection": 1,
    "url": "http://example.com/api/groups/1/",
    "roles": [
        "http://example.com/api/roles/1/",
        "http://example.com/api/roles/2/"
    ],
    "languages": [
        "http://example.com/api/languages/en/",
        "http://example.com/api/languages/cs/",
    ],
    "projects": [
        "http://example.com/api/projects/demo1/",
        "http://example.com/api/projects/demo/"
    ],
    "componentlist": "http://example.com/api/component-lists/new/",
    "components": [
        "http://example.com/api/components/demo/weblate/"
    ]
}
```

PUT /api/groups/(int: *id*)/

Changes the group parameters.

Parameters

- **id** (*int*) – Group's ID

Response JSON Object

- **name** (*string*) – name of a group
- **project_selection** (*int*) – integer corresponding to group of projects
- **language_selection** (*int*) – integer corresponding to group of Languages

PATCH /api/groups/(int: *id*)/

Changes the group parameters.

Parameters

- **id** (*int*) – Group's ID

Response JSON Object

- **name** (*string*) – name of a group
- **project_selection** (*int*) – integer corresponding to group of projects
- **language_selection** (*int*) – integer corresponding to group of languages

DELETE /api/groups/(int: *id*)/

Deletes the group.

Parameters

- **id** (*int*) – Group's ID

POST /api/groups/(**int:** *id*)**/roles/**
Associate roles with a group.

Parameters

- **id** (*int*) – Group's ID

Form Parameters

- **string role_id** – The unique role ID

POST /api/groups/(**int:** *id*)**/components/**
Associate components with a group.

Parameters

- **id** (*int*) – Group's ID

Form Parameters

- **string component_id** – The unique component ID

DELETE /api/groups/(**int:** *id*)**/components/**
int: *component_id* Delete component from a group.

Parameters

- **id** (*int*) – Group's ID
- **component_id** (*int*) – The unique component ID

POST /api/groups/(**int:** *id*)**/projects/**
Associate projects with a group.

Parameters

- **id** (*int*) – Group's ID

Form Parameters

- **string project_id** – The unique project ID

DELETE /api/groups/(**int:** *id*)**/projects/**
int: *project_id* Delete project from a group.

Parameters

- **id** (*int*) – Group's ID
- **project_id** (*int*) – The unique project ID

POST /api/groups/(**int:** *id*)**/languages/**
Associate languages with a group.

Parameters

- **id** (*int*) – Group's ID

Form Parameters

- **string language_code** – The unique language code

DELETE /api/groups/(**int:** *id*)**/languages/**
string: *language_code* Delete language from a group.

Parameters

- **id** (*int*) – Group's ID
- **language_code** (*string*) – The unique language code

POST /api/groups/(int: *id*)/componentlists/
Associate componentlists with a group.

Parameters

- **id** (*int*) – Group's ID

Form Parameters

- **string component_list_id** – The unique componentlist ID

DELETE /api/groups/(int: *id*)/componentlists/
int: *component_list_id* Delete componentlist from a group.

Parameters

- **id** (*int*) – Group's ID
- **component_list_id** (*int*) – The unique componentlist ID

1.12.5 Roluri

GET /api/roles/
Returns a list of all roles associated with user. If user is superuser, then list of all existing roles is returned.

Vezi și:

Roles object attributes are documented at *GET /api/roles/(int:id)/*.

POST /api/roles/
Creates a new role.

Parameters

- **name** (*string*) – Role name
- **permissions** (*array*) – List of codenames of permissions

GET /api/roles/(int: *id*)/
Returns information about a role.

Parameters

- **id** (*int*) – Role ID

Response JSON Object

- **name** (*string*) – Role name
- **permissions** (*array*) – list of codenames of permissions

Example JSON data:

```
{
    "name": "Access repository",
    "permissions": [
        "vcs.access",
        "vcs.view"
    ],
    "url": "http://example.com/api/roles/1/",
}
```

PUT /api/roles/(int: *id*)/
Changes the role parameters.

Parameters

- **id** (*int*) – Role's ID

Response JSON Object

- **name** (*string*) – Role name
- **permissions** (*array*) – list of codenames of permissions

PATCH /api/roles/(int: *id*)/
Changes the role parameters.

Parameters

- **id** (*int*) – Role's ID

Response JSON Object

- **name** (*string*) – Role name
- **permissions** (*array*) – list of codenames of permissions

DELETE /api/roles/(int: *id*)/
Deletes the role.

Parameters

- **id** (*int*) – Role's ID

1.12.6 Limbi

GET /api/languages/
Returns a list of all languages.

Vezi și:

Language object attributes are documented at *GET /api/languages/(string:language)/*.

POST /api/languages/
Creates a new language.

Parameters

- **code** (*string*) – Numele limbii
- **name** (*string*) – Numele limbii
- **direction** (*string*) – Direcția textului
- **plural** (*object*) – Language plural formula and number

GET /api/languages/(string: *language*)/
Returns information about a language.

Parameters

- **language** (*string*) – Codul limbii

Response JSON Object

- **code** (*string*) – Codul limbii
- **direction** (*string*) – Direcția textului
- **plural** (*object*) – Object of language plural information
- **aliases** (*array*) – Array of aliases for language

Example JSON data:

```
{
    "code": "en",
    "direction": "ltr",
    "name": "English",
    "plural": {
```

(continues on next page)

(continuare din pagina precedentă)

```
        "id": 75,
        "source": 0,
        "number": 2,
        "formula": "n != 1",
        "type": 1
    },
    "aliases": [
        "english",
        "en_en",
        "base",
        "source",
        "eng"
    ],
    "url": "http://example.com/api/languages/en/",
    "web_url": "http://example.com/languages/en/",
    "statistics_url": "http://example.com/api/languages/en/statistics/"
}
```

PUT /api/languages/(string: *language*)/

Changes the language parameters.

Parameters

- **language** (*string*) – Language's code

Request JSON Object

- **name** (*string*) – Numele limbii
- **direction** (*string*) – Direcția textului
- **plural** (*object*) – Language plural details

PATCH /api/languages/(string: *language*)/

Changes the language parameters.

Parameters

- **language** (*string*) – Language's code

Request JSON Object

- **name** (*string*) – Numele limbii
- **direction** (*string*) – Direcția textului
- **plural** (*object*) – Language plural details

DELETE /api/languages/(string: *language*)/

Deletes the language.

Parameters

- **language** (*string*) – Language's code

GET /api/languages/(string: *language*)/statistics/

Returns statistics for a language.

Parameters

- **language** (*string*) – Codul limbii

Response JSON Object

- **total** (*int*) – total number of strings
- **total_words** (*int*) – total number of words
- **last_change** (*timestamp*) – last changes in the language

- **recent_changes** (*int*) – total number of changes
- **translated** (*int*) – number of translated strings
- **translated_percent** (*float*) – percentage of translated strings
- **translated_words** (*int*) – number of translated words
- **translated_words_percent** (*int*) – percentage of translated words
- **translated_chars** (*int*) – number of translated characters
- **translated_chars_percent** (*int*) – percentage of translated characters
- **total_chars** (*int*) – number of total characters
- **fuzzy** (*int*) – number of fuzzy (marked for edit) strings
- **fuzzy_percent** (*int*) – percentage of fuzzy (marked for edit) strings
- **failing** (*int*) – number of failing strings
- **failing** – percentage of failing strings

1.12.7 Proiecte

GET /api/projects/

Returns a list of all projects.

Vezi și:

Project object attributes are documented at *GET /api/projects/(string:project)/*.

POST /api/projects/

Nou în versiunea 3.9.

Creates a new project.

Parameters

- **name** (*string*) – Denumire proiect
- **slug** (*string*) – Project slug
- **web** (*string*) – Website-ul proiectului

GET /api/projects/(string: *project*)/

Returns information about a project.

Parameters

- **project** (*string*) – Lipsă URL a proiectului

Response JSON Object

- **name** (*string*) – numele proiectului
- **slug** (*string*) – project slug
- **web** (*string*) – project website
- **components_list_url** (*string*) – URL to components list; see *GET /api/projects/(string:project)/components/*
- **repository_url** (*string*) – URL to repository status; see *GET /api/projects/(string:project)/repository/*
- **changes_list_url** (*string*) – URL to changes list; see *GET /api/projects/(string:project)/changes/*
- **translation_review** (*boolean*) – *Activați recenzii*

- **source_review** (*boolean*) – *Activați revizuirile sursei*
- **set_language_team** (*boolean*) – *Setați antetul „Language-Team”*
- **enable_hooks** (*boolean*) – *Activați cârligele*
- **instructions** (*string*) – *Instrucțiuni de traducere*
- **language_aliases** (*string*) – *Aliasuri lingvistice*

Example JSON data:

```
{
    "name": "Hello",
    "slug": "hello",
    "url": "http://example.com/api/projects/hello/",
    "web": "https://weblate.org/",
    "web_url": "http://example.com/projects/hello/"
}
```

PATCH /api/projects/(string: *project*)/

Nou în versiunea 4.3.

Edit a project by a PATCH request.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

PUT /api/projects/(string: *project*)/

Nou în versiunea 4.3.

Edit a project by a PUT request.

Parameters

- **project** (*string*) – Lipsă URL a proiectului

DELETE /api/projects/(string: *project*)/

Nou în versiunea 3.9.

Deletes a project.

Parameters

- **project** (*string*) – Lipsă URL a proiectului

GET /api/projects/(string: *project*)/changes/

Returns a list of project changes. This is essentially a project scoped `GET /api/changes/` accepting same params.

Parameters

- **project** (*string*) – Lipsă URL a proiectului

Response JSON Object

- **results** (*array*) – array of component objects; see `GET /api/changes/(int:id)/`

GET /api/projects/(string: *project*)/repository/

Returns information about VCS repository status. This endpoint contains only an overall summary for all repositories for the project. To get more detailed status use `GET /api/components/(string:project)/(string:component)/repository/`.

Parameters

- **project** (*string*) – Lipsă URL a proiectului

Response JSON Object

- **needs_commit** (*boolean*) – whether there are any pending changes to commit
- **needs_merge** (*boolean*) – whether there are any upstream changes to merge
- **needs_push** (*boolean*) – whether there are any local changes to push

Example JSON data:

```
{
    "needs_commit": true,
    "needs_merge": false,
    "needs_push": true
}
```

POST /api/projects/(string: *project*)/repository/

Performs given operation on the VCS repository.

Parameters

- **project** (*string*) – Lipsă URL a proiectului

Request JSON Object

- **operation** (*string*) – Operation to perform: one of `push`, `pull`, `commit`, `reset`, `cleanup`, `file-sync`

Response JSON Object

- **result** (*boolean*) – result of the operation

CURL example:

```
curl \
    -d operation=pull \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/projects/hello/repository/
```

JSON request example:

```
POST /api/projects/hello/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"operation":"pull"}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{"result":true}
```

GET /api/projects/(string: *project*)/components/

Returns a list of translation components in the given project.

Parameters

- **project** (*string*) – Lipsă URL a proiectului

Response JSON Object

- **results** (*array*) – array of component objects; see *GET /api/components/(string:project)/(string:component)/*

POST /api/projects/(string: *project*)/components/

Nou în versiunea 3.9.

Schimbat în versiunea 4.3: The `zipfile` and `docfile` parameters are now accepted for VCS-less components, see *Local files*.

Schimbat în versiunea 4.6: The cloned repositories are now automatically shared within a project using *Weblate internal URLs*. Use `disable_autoshare` to turn off this.

Creates translation components in the given project.

Sugestie: Use *Weblate internal URLs* when creating multiple components from a single VCS repository.

Notă: Most of the component creation happens in the background. Check the `task_url` attribute of created component and follow the progress there.

Parameters

- **project** (*string*) – Lipsă URL a proiectului

Form Parameters

- **file zipfile** – ZIP file to upload into Weblate for translations initialization
- **file docfile** – Document de tradus
- **boolean disable_autoshare** – Disables automatic repository sharing via *Weblate internal URLs*.

Response JSON Object

- **result** (*object*) – Created component object; see *GET /api/components/(string:project)/(string:component)/*

JSON can not be used when uploading the files using the `zipfile` and `docfile` parameters. The data has to be uploaded as *multipart/form-data*.

CURL form request example:

```
curl \
    --form docfile=@strings.html \
    --form name=Weblate \
    --form slug=weblate \
    --form file_format=html \
    --form new_lang=add \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/projects/hello/components/
```

CURL JSON request example:

```
curl \
    --data-binary '{
        "branch": "main",
        "file_format": "po",
        "filemask": "po/*.po",
        "git_export": "",
        "license": "",
```

(continues on next page)

(continuare din pagina precedentă)

```
        "license_url": "",
        "name": "Weblate",
        "slug": "weblate",
        "repo": "file:///home/nijel/work/weblate-hello",
        "template": "",
        "new_base": "",
        "vcs": "git"
    }' \
    -H "Content-Type: application/json" \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/projects/hello/components/
```

JSON request example:

```
POST /api/projects/hello/components/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{
    "branch": "main",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "Weblate",
    "slug": "weblate",
    "repo": "file:///home/nijel/work/weblate-hello",
    "template": "",
    "new_base": "",
    "vcs": "git"
}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{
    "branch": "main",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "Weblate",
    "slug": "weblate",
    "project": {
        "name": "Hello",
        "slug": "hello",
        "source_language": {
            "code": "en",
```

(continues on next page)

(continuare din pagina precedentă)

```
            "direction": "ltr",
            "name": "English",
            "url": "http://example.com/api/languages/en/",
            "web_url": "http://example.com/languages/en/"
        },
        "url": "http://example.com/api/projects/hello/",
        "web": "https://weblate.org/",
        "web_url": "http://example.com/projects/hello/"
    },
    "repo": "file:///home/nijel/work/weblate-hello",
    "template": "",
    "new_base": "",
    "url": "http://example.com/api/components/hello/weblate/",
    "vcs": "git",
    "web_url": "http://example.com/projects/hello/weblate/"
}
```

GET /api/projects/(string: *project*)/languages/

Returns paginated statistics for all languages within a project.

Nou în versiunea 3.8.

Parameters

- **project** (*string*) – Lipsă URL a proiectului

Response JSON Object

- **results** (*array*) – array of translation statistics objects
- **language** (*string*) – language name
- **code** (*string*) – language code
- **total** (*int*) – total number of strings
- **translated** (*int*) – number of translated strings
- **translated_percent** (*float*) – percentage of translated strings
- **total_words** (*int*) – total number of words
- **translated_words** (*int*) – number of translated words
- **words_percent** (*float*) – percentage of translated words

GET /api/projects/(string: *project*)/statistics/

Returns statistics for a project.

Nou în versiunea 3.8.

Parameters

- **project** (*string*) – Lipsă URL a proiectului

Response JSON Object

- **total** (*int*) – total number of strings
- **translated** (*int*) – number of translated strings
- **translated_percent** (*float*) – percentage of translated strings
- **total_words** (*int*) – total number of words
- **translated_words** (*int*) – number of translated words
- **words_percent** (*float*) – percentage of translated words

1.12.8 Componente

GET /api/components/

Returns a list of translation components.

Vezi și:

Component object attributes are documented at `GET /api/components/(string:project)/(string:component)/`.

GET /api/components/(string: *project*)/(string: *component*)/

Returns information about translation component.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Response JSON Object

- **project** (*object*) – the translation project; see `GET /api/projects/(string:project)/`
- **name** (*string*) – *Denumirea componentei*
- **slug** (*string*) – *Component slug*
- **vcs** (*string*) – *Sistem de control al versiunilor*
- **repo** (*string*) – *Depozitul de cod sursă*
- **git_export** (*string*) – *URL-ul depozitului exportat*
- **branch** (*string*) – *Ramura de depozit*
- **push_branch** (*string*) – *Împingeți ramura*
- **filemask** (*string*) – *File mask*
- **template** (*string*) – *Fișier de limbă de bază monolingvă*
- **edit_template** (*string*) – *Editarea fișierului de bază*
- **intermediate** (*string*) – *Fișier de limbaj intermediar*
- **new_base** (*string*) – *Șablon pentru traduceri noi*
- **file_format** (*string*) – *Formatul fișierului*
- **license** (*string*) – *Licență de traducere*
- **agreement** (*string*) – *Acord de colaborare*
- **new_lang** (*string*) – *Adăugarea unei noi traduceri*
- **language_code_style** (*string*) – *Stil de cod lingvistic*
- **source_language** (*object*) – source language object; see `GET /api/languages/(string:language)/`
- **push** (*string*) – *URL de împingere a depozitului*
- **check_flags** (*string*) – *Steaguri de traducere*
- **priority** (*string*) – *Prioritate*
- **enforced_checks** (*string*) – *Controale forțate*
- **restricted** (*string*) – *Restricted access*
- **repoweb** (*string*) – *Navigator de depozite*
- **report_source_bugs** (*string*) – *Adresa de raportare a erorilor din șirul sursă*

- **merge_style** (*string*) – *Stil de fuziune*
- **commit_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **add_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **delete_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **merge_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **addon_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **allow_translation_propagation** (*string*) – *Permiteți propagarea traducerii*
- **enable_suggestions** (*string*) – *Enable suggestions*
- **suggestion_voting** (*string*) – *Votul prin sugestie*
- **suggestion_autoaccept** (*string*) – *Sugestii de autoacceptare*
- **push_on_commit** (*string*) – *Împingeți pe comitere*
- **commit_pending_age** (*string*) – *Vârsta modificărilor care urmează să fie confirmate*
- **auto_lock_error** (*string*) – *Blocare pe eroare*
- **language_regex** (*string*) – *Filtru lingvistic*
- **variant_regex** (*string*) – *Variante expresie regulată*
- **repository_url** (*string*) – URL to repository status; see `GET /api/components/(string:project)/(string:component)/repository/`
- **translations_url** (*string*) – URL to translations list; see `GET /api/components/(string:project)/(string:component)/translations/`
- **lock_url** (*string*) – URL to lock status; see `GET /api/components/(string:project)/(string:component)/lock/`
- **changes_list_url** (*string*) – URL to changes list; see `GET /api/components/(string:project)/(string:component)/changes/`
- **task_url** (*string*) – URL to a background task (if any); see `GET /api/tasks/(str:uuid)/`

Example JSON data:

```
{
    "branch": "main",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "Weblate",
    "slug": "weblate",
    "project": {
        "name": "Hello",
        "slug": "hello",
        "source_language": {
            "code": "en",
            "direction": "ltr",
            "name": "English",
            "url": "http://example.com/api/languages/en/",
            "web_url": "http://example.com/languages/en/"
        },
        "url": "http://example.com/api/projects/hello/",
```

(continues on next page)

(continuare din pagina precedentă)

```
        "web": "https://weblate.org/",
        "web_url": "http://example.com/projects/hello/"
    },
    "source_language": {
        "code": "en",
        "direction": "ltr",
        "name": "English",
        "url": "http://example.com/api/languages/en/",
        "web_url": "http://example.com/languages/en/"
    },
    "repo": "file:///home/nijel/work/weblate-hello",
    "template": "",
    "new_base": "",
    "url": "http://example.com/api/components/hello/weblate/",
    "vcs": "git",
    "web_url": "http://example.com/projects/hello/weblate/"
}
```

PATCH /api/components/(string: *project*)/(string: *component*)/ Edit a component by a PATCH request.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug
- **source_language** (*string*) – Project source language code (optional)

Request JSON Object

- **name** (*string*) – name of component
- **slug** (*string*) – slug of component
- **repo** (*string*) – VCS repository URL

CURL example:

```
curl \
    --data-binary '{"name": "new name"}' \
    -H "Content-Type: application/json" \
    -H "Authorization: Token TOKEN" \
    PATCH http://example.com/api/projects/hello/components/
```

JSON request example:

```
PATCH /api/projects/hello/components/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{
    "name": "new name"
}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
```

(continues on next page)

(continuare din pagina precedentă)

```
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{
    "branch": "main",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "new name",
    "slug": "weblate",
    "project": {
        "name": "Hello",
        "slug": "hello",
        "source_language": {
            "code": "en",
            "direction": "ltr",
            "name": "English",
            "url": "http://example.com/api/languages/en/",
            "web_url": "http://example.com/languages/en/"
        },
        "url": "http://example.com/api/projects/hello/",
        "web": "https://weblate.org/",
        "web_url": "http://example.com/projects/hello/"
    },
    "repo": "file:///home/nijel/work/weblate-hello",
    "template": "",
    "new_base": "",
    "url": "http://example.com/api/components/hello/weblate/",
    "vcs": "git",
    "web_url": "http://example.com/projects/hello/weblate/"
}
```

PUT /api/components/(string: *project*)/(string: *component*)/ Edit a component by a PUT request.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Request JSON Object

- **branch** (*string*) – VCS repository branch
- **file_format** (*string*) – file format of translations
- **filemask** (*string*) – mask of translation files in the repository
- **name** (*string*) – name of component
- **slug** (*string*) – slug of component
- **repo** (*string*) – VCS repository URL
- **template** (*string*) – base file for monolingual translations
- **new_base** (*string*) – base file for adding new translations
- **vcs** (*string*) – version control system

DELETE /api/components/(string: *project*)/(string: *component*)/ Nou în versiunea 3.9.

Deletes a component.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

GET /api/components/(string: *project*)/string: *component*/changes/ Returns a list of component changes. This is essentially a component scoped *GET /api/changes/* accepting same params.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Response JSON Object

- **results** (*array*) – array of component objects; see *GET /api/changes/(int:id)/*

GET /api/components/(string: *project*)/string: *component*/screenshots/ Returns a list of component screenshots.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Response JSON Object

- **results** (*array*) – array of component screenshots; see *GET /api/screenshots/(int:id)/*

GET /api/components/(string: *project*)/string: *component*/lock/ Returns component lock status.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Response JSON Object

- **locked** (*boolean*) – whether component is locked for updates

Example JSON data:

```
{
    "locked": false
}
```

POST /api/components/(string: *project*)/string: *component*/lock/ Sets component lock status.

Response is same as *GET /api/components/(string:project)/(string:component)/lock/*.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Request JSON Object

- **lock** – Boolean whether to lock or not.

CURL example:

```
curl \
    -d lock=true \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/components/hello/weblate/repository/
```

JSON request example:

```
POST /api/components/hello/weblate/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"lock": true}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{"locked":true}
```

GET /api/components/(string: *project*)/(string: *component*)/repository/ Returns information about VCS repository status.

The response is same as for *GET /api/projects/(string:project)/repository/*.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Response JSON Object

- **needs_commit** (*boolean*) – whether there are any pending changes to commit
- **needs_merge** (*boolean*) – whether there are any upstream changes to merge
- **needs_push** (*boolean*) – whether there are any local changes to push
- **remote_commit** (*string*) – Remote commit information
- **status** (*string*) – VCS repository status as reported by VCS
- **merge_failure** – Text describing merge failure or null if there is none

POST /api/components/(string: *project*)/(string: *component*)/repository/ Performs the given operation on a VCS repository.

See *POST /api/projects/(string:project)/repository/* for documentation.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Request JSON Object

- **operation** (*string*) – Operation to perform: one of `push`, `pull`, `commit`, `reset`, `cleanup`

Response JSON Object

- **result** (*boolean*) – result of the operation

CURL example:

```
curl \
    -d operation=pull \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/components/hello/weblate/repository/
```

JSON request example:

```
POST /api/components/hello/weblate/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"operation":"pull"}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{"result":true}
```

GET /api/components/(string: *project*)/(string: *component*)/monolingual_base/ Downloads base file for monolingual translations.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

GET /api/components/(string: *project*)/(string: *component*)/new_template/ Downloads template file for new translations.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

GET /api/components/(string: *project*)/(string: *component*)/translations/ Returns a list of translation objects in the given component.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Response JSON Object

- **results** (*array*) – array of translation objects; see *GET /api/translations/(string:project)/(string:component)/(string:language)/*

POST /api/components/(string: *project*)/(string: *component*)/translations/ Creates new translation in the given component.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Request JSON Object

- **language_code** (*string*) – translation language code; see *GET /api/languages/(string:language)/*

Response JSON Object

- **result** (*object*) – new translation object created

CURL example:

```
curl \
    -d language_code=cs \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/projects/hello/components/
```

JSON request example:

```
POST /api/projects/hello/components/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"language_code": "cs"}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{
    "failing_checks": 0,
    "failing_checks_percent": 0,
    "failing_checks_words": 0,
    "filename": "po/cs.po",
    "fuzzy": 0,
    "fuzzy_percent": 0.0,
    "fuzzy_words": 0,
    "have_comment": 0,
    "have_suggestion": 0,
    "is_template": false,
    "is_source": false,
    "language": {
        "code": "cs",
        "direction": "ltr",
        "name": "Czech",
        "url": "http://example.com/api/languages/cs/",
        "web_url": "http://example.com/languages/cs/"
    },
    "language_code": "cs",
    "id": 125,
```

(continues on next page)

(continuare din pagina precedentă)

```
    "last_author": null,
    "last_change": null,
    "share_url": "http://example.com/engage/hello/cs/",
    "total": 4,
    "total_words": 15,
    "translate_url": "http://example.com/translate/hello/weblate/cs/",
    "translated": 0,
    "translated_percent": 0.0,
    "translated_words": 0,
    "url": "http://example.com/api/translations/hello/weblate/cs/",
    "web_url": "http://example.com/projects/hello/weblate/cs/"
}
```

GET /api/components/(string: *project*)/string: *component*/statistics/ Returns paginated statistics for all translations within component.

Nou în versiunea 2.7.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Response JSON Object

- **results** (*array*) – array of translation statistics objects; see *GET /api/translations/(string:project)/(string:component)/(string:language)/statistics/*

GET /api/components/(string: *project*)/string: *component*/links/ Returns projects linked with a component.

Nou în versiunea 4.5.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Response JSON Object

- **projects** (*array*) – associated projects; see *GET /api/projects/(string:project)/*

POST /api/components/(string: *project*)/string: *component*/links/ Associate project with a component.

Nou în versiunea 4.5.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug

Form Parameters

- **string project_slug** – Project slug

DELETE /api/components/(string: *project*)/string: *component*/links/string: *project_slug*/ Remove association of a project with a component.

Nou în versiunea 4.5.

Parameters

- **project** (*string*) – Lipsă URL a proiectului

- **component** (*string*) – Componenta URL slug
- **project_slug** (*string*) – Slug of the project to remove

1.12.9 Traduceri

GET /api/translations/
Returns a list of translations.

Vezi și:

Translation object attributes are documented at *GET /api/translations/(string:project)/(string:component)/(string:language)/*.

GET /api/translations/(string: *project*)**/string:** *component***/string:** *language***/** Returns information about a translation.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug
- **language** (*string*) – Translation language code

Response JSON Object

- **component** (*object*) – component object; see *GET /api/components/(string:project)/(string:component)/*
- **failing_checks** (*int*) – number of strings failing checks
- **failing_checks_percent** (*float*) – percentage of strings failing checks
- **failing_checks_words** (*int*) – number of words with failing checks
- **filename** (*string*) – translation filename
- **fuzzy** (*int*) – number of fuzzy (marked for edit) strings
- **fuzzy_percent** (*float*) – percentage of fuzzy (marked for edit) strings
- **fuzzy_words** (*int*) – number of words in fuzzy (marked for edit) strings
- **have_comment** (*int*) – number of strings with comment
- **have_suggestion** (*int*) – number of strings with suggestion
- **is_template** (*boolean*) – whether the translation has a monolingual base
- **language** (*object*) – source language object; see *GET /api/languages/(string:language)/*
- **language_code** (*string*) – language code used in the repository; this can be different from language code in the language object
- **last_author** (*string*) – name of last author
- **last_change** (*timestamp*) – last change timestamp
- **revision** (*string*) – revision hash for the file
- **share_url** (*string*) – URL for sharing leading to engagement page
- **total** (*int*) – total number of strings
- **total_words** (*int*) – total number of words
- **translate_url** (*string*) – URL for translating
- **translated** (*int*) – number of translated strings

- **translated_percent** (*float*) – percentage of translated strings
- **translated_words** (*int*) – number of translated words
- **repository_url** (*string*) – URL to repository status; see *GET /api/translations/(string:project)/(string:component)/(string:language)/repository/*
- **file_url** (*string*) – URL to file object; see *GET /api/translations/(string:project)/(string:component)/(string:language)/file/*
- **changes_list_url** (*string*) – URL to changes list; see *GET /api/translations/(string:project)/(string:component)/(string:language)/changes/*
- **units_list_url** (*string*) – URL to strings list; see *GET /api/translations/(string:project)/(string:component)/(string:language)/units/*

Example JSON data:

```
{
    "component": {
        "branch": "main",
        "file_format": "po",
        "filemask": "po/*.po",
        "git_export": "",
        "license": "",
        "license_url": "",
        "name": "Weblate",
        "new_base": "",
        "project": {
            "name": "Hello",
            "slug": "hello",
            "source_language": {
                "code": "en",
                "direction": "ltr",
                "name": "English",
                "url": "http://example.com/api/languages/en/",
                "web_url": "http://example.com/languages/en/"
            },
            "url": "http://example.com/api/projects/hello/",
            "web": "https://weblate.org/",
            "web_url": "http://example.com/projects/hello/"
        },
        "repo": "file:///home/nijel/work/weblate-hello",
        "slug": "weblate",
        "template": "",
        "url": "http://example.com/api/components/hello/weblate/",
        "vcs": "git",
        "web_url": "http://example.com/projects/hello/weblate/"
    },
    "failing_checks": 3,
    "failing_checks_percent": 75.0,
    "failing_checks_words": 11,
    "filename": "po/cs.po",
    "fuzzy": 0,
    "fuzzy_percent": 0.0,
    "fuzzy_words": 0,
    "have_comment": 0,
    "have_suggestion": 0,
    "is_template": false,
    "language": {
```

(continues on next page)

(continuare din pagina precedentă)

```
        "code": "cs",
        "direction": "ltr",
        "name": "Czech",
        "url": "http://example.com/api/languages/cs/",
        "web_url": "http://example.com/languages/cs/"
    },
    "language_code": "cs",
    "last_author": "Weblate Admin",
    "last_change": "2016-03-07T10:20:05.499",
    "revision": "7ddfafe6daaf57fc8654cc852ea6be212b015792",
    "share_url": "http://example.com/engage/hello/cs/",
    "total": 4,
    "total_words": 15,
    "translate_url": "http://example.com/translate/hello/weblate/cs/",
    "translated": 4,
    "translated_percent": 100.0,
    "translated_words": 15,
    "url": "http://example.com/api/translations/hello/weblate/cs/",
    "web_url": "http://example.com/projects/hello/weblate/cs/"
}
```

DELETE /api/translations/(string: *project*)/(string: *component*/string: *language*/ Nou în versiunea 3.9.

Deletes a translation.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug
- **language** (*string*) – Translation language code

GET /api/translations/(string: *project*)/(string: *component*/string: *language*/changes/ Returns a list of translation changes. This is essentially a translations-scoped *GET /api/changes/* accepting the same parameters.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug
- **language** (*string*) – Translation language code

Response JSON Object

- **results** (*array*) – array of component objects; see *GET /api/changes/(int:id)/*

GET /api/translations/(string: *project*)/(string: *component*/string: *language*/units/ Returns a list of translation units.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug
- **language** (*string*) – Translation language code
- **q** (*string*) – Search query string *Searching* (optional)

Response JSON Object

- **results** (*array*) – array of component objects; see *GET /api/units/(int:id)/*

POST /api/translations/(string: *project*)/
string: *component*/string: *language*/units/ Add new monolingual unit.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug
- **language** (*string*) – Translation language code

Request JSON Object

- **key** (*string*) – Name of translation unit
- **value** (*array*) – The translation unit value

Vezi și:

Gestionați șirurile de caractere, adding-new-strings

POST /api/translations/(string: *project*)/
string: *component*/string: *language*/autotranslate/ Trigger automatic translation.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug
- **language** (*string*) – Translation language code

Request JSON Object

- **mode** (*string*) – Modul de traducere automată
- **filter_type** (*string*) – Automatic translation filter type
- **auto_source** (*string*) – Automatic translation source - `mt` or `others`
- **component** (*string*) – Activați contribuția la memoria de traducere partajată pentru proiect pentru a avea acces la componente suplimentare.
- **engines** (*array*) – Motoare de traducere automată
- **threshold** (*string*) – Prag de punctaj

GET /api/translations/(string: *project*)/
string: *component*/string: *language*/file/ Download current translation file as it is stored in the VCS (without the `format` parameter) or converted to another format (see *Downloading translations*).

Notă: This API endpoint uses different logic for output than rest of API as it operates on whole file rather than on data. Set of accepted `format` parameter differs and without such parameter you get translation file as stored in VCS.

Query Parameters

- **format** – File format to use; if not specified no format conversion happens; supported file formats: `po`, `mo`, `xliff`, `xliff11`, `tbx`, `csv`, `xlsx`, `json`, `aresource`, `strings`

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug
- **language** (*string*) – Translation language code

POST /api/translations/(string: *project*)/
string: *component*/string: *language*/file/ Upload new file with translations.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug
- **language** (*string*) – Translation language code

Form Parameters

- **string conflict** – How to deal with conflicts (`ignore`, `replace-translated` or `replace-approved`)
- **file file** – Uploaded file
- **string email** – E-mail de autor
- **string author** – Numele autorului
- **string method** – Upload method (`translate`, `approve`, `suggest`, `fuzzy`, `replace`, `source`, `add`), see *Import methods*
- **string fuzzy** – Fuzzy (marked for edit) strings processing (*empty*, `process`, `approve`)

CURL example:

```
curl -X POST \
    -F file=@strings.xml \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/translations/hello/android/cs/file/
```

GET /api/translations/(string: *project*)/
string: *component*/string: *language*/repository/ Returns information about VCS repository status.

The response is same as for *GET /api/components/(string:project)/(string:component)/repository/*.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug
- **language** (*string*) – Translation language code

POST /api/translations/(string: *project*)/
string: *component*/string: *language*/repository/ Performs given operation on the VCS repository.

See *POST /api/projects/(string:project)/repository/* for documentation.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug
- **language** (*string*) – Translation language code

Request JSON Object

- **operation** (*string*) – Operation to perform: one of `push`, `pull`, `commit`, `reset`, `cleanup`

Response JSON Object

- **result** (*boolean*) – result of the operation

GET /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/statistics/ Returns detailed translation statistics.

Nou în versiunea 2.7.

Parameters

- **project** (*string*) – Lipsă URL a proiectului
- **component** (*string*) – Componenta URL slug
- **language** (*string*) – Translation language code

Response JSON Object

- **code** (*string*) – language code
- **failing** (*int*) – number of failing checks
- **failing_percent** (*float*) – percentage of failing checks
- **fuzzy** (*int*) – number of fuzzy (marked for edit) strings
- **fuzzy_percent** (*float*) – percentage of fuzzy (marked for edit) strings
- **total_words** (*int*) – total number of words
- **translated_words** (*int*) – number of translated words
- **last_author** (*string*) – name of last author
- **last_change** (*timestamp*) – date of last change
- **name** (*string*) – language name
- **total** (*int*) – total number of strings
- **translated** (*int*) – number of translated strings
- **translated_percent** (*float*) – percentage of translated strings
- **url** (*string*) – URL to access the translation (engagement URL)
- **url_translate** (*string*) – URL to access the translation (real translation URL)

1.12.10 Units

A *unit* is a single piece of a translation which pairs a source string with a corresponding translated string and also contains some related metadata. The term is derived from the Translate Toolkit and XLIFF.

Nou în versiunea 2.10.

GET /api/units/

Returns list of translation units.

Vezi și:

Unit object attributes are documented at `GET /api/units/(int:id)/`.

GET /api/units/(int: *id*)/

Schimbat în versiunea 4.3: The `target` and `source` are now arrays to properly handle plural strings.

Returns information about translation unit.

Parameters

- **id** (*int*) – Unit ID

Response JSON Object

- **translation** (*string*) – URL of a related translation object
- **source** (*array*) – source string

- **previous_source** (*string*) – previous source string used for fuzzy matching
- **target** (*array*) – target string
- **id_hash** (*string*) – unique identifier of the unit
- **content_hash** (*string*) – unique identifier of the source string
- **location** (*string*) – location of the unit in source code
- **context** (*string*) – translation unit context
- **note** (*string*) – translation unit note
- **flags** (*string*) – translation unit flags
- **state** (*int*) – unit state, 0 - not translated, 10 - needs editing, 20 - translated, 30 - approved, 100 - read only
- **fuzzy** (*boolean*) – whether the unit is fuzzy or marked for review
- **translated** (*boolean*) – whether the unit is translated
- **approved** (*boolean*) – whether the translation is approved
- **position** (*int*) – unit position in translation file
- **has_suggestion** (*boolean*) – whether the unit has suggestions
- **has_comment** (*boolean*) – whether the unit has comments
- **has_failing_check** (*boolean*) – whether the unit has failing checks
- **num_words** (*int*) – number of source words
- **priority** (*int*) – translation priority; 100 is default
- **id** (*int*) – unit identifier
- **explanation** (*string*) – String explanation, available on source units, see *Additional info on source strings*
- **extra_flags** (*string*) – Additional string flags, available on source units, see *Personalizarea comportamentului cu ajutorul steguleţelor*
- **web_url** (*string*) – URL where the unit can be edited
- **souce_unit** (*string*) – Source unit link; see `GET /api/units/(int:id)/`

PATCH /api/units/(int: *id*)/

Nou în versiunea 4.3.

Performs partial update on translation unit.

Parameters

- **id** (*int*) – Unit ID

Request JSON Object

- **state** (*int*) – unit state, 0 - not translated, 10 - needs editing, 20 - translated, 30 - approved (need review workflow enabled, see *Dedicated reviewers*)
- **target** (*array*) – target string
- **explanation** (*string*) – String explanation, available on source units, see *Additional info on source strings*
- **extra_flags** (*string*) – Additional string flags, available on source units, see *Personalizarea comportamentului cu ajutorul steguleţelor*

PUT /api/units/(int: *id*)/

Nou în versiunea 4.3.

Performs full update on translation unit.

Parameters

- **id** (*int*) – Unit ID

Request JSON Object

- **state** (*int*) – unit state, 0 - not translated, 10 - needs editing, 20 - translated, 30 - approved (need review workflow enabled, see *Dedicated reviewers*)
- **target** (*array*) – target string
- **explanation** (*string*) – String explanation, available on source units, see *Additional info on source strings*
- **extra_flags** (*string*) – Additional string flags, available on source units, see *Personalizarea comportamentului cu ajutorul steguleţelor*

DELETE /api/units/(int: *id*)/

Nou în versiunea 4.3.

Deletes a translation unit.

Parameters

- **id** (*int*) – Unit ID

1.12.11 Modificări

Nou în versiunea 2.10.

GET /api/changes/

Schimbat în versiunea 4.1: Filtering of changes was introduced in the 4.1 release.

Returns a list of translation changes.

Vezi şi:

Change object attributes are documented at `GET /api/changes/(int:id)/`.

Query Parameters

- **user** (*string*) – Username of user to filters
- **action** (*int*) – Action to filter, can be used several times
- **timestamp_after** (*timestamp*) – ISO 8601 formatted timestamp to list changes after
- **timestamp_before** (*timestamp*) – ISO 8601 formatted timestamp to list changes before

GET /api/changes/(int: *id*)/

Returns information about translation change.

Parameters

- **id** (*int*) – Change ID

Response JSON Object

- **unit** (*string*) – URL of a related unit object
- **translation** (*string*) – URL of a related translation object
- **component** (*string*) – URL of a related component object
- **user** (*string*) – URL of a related user object
- **author** (*string*) – URL of a related author object
- **timestamp** (*timestamp*) – event timestamp

- **action** (*int*) – numeric identification of action
- **action_name** (*string*) – text description of action
- **target** (*string*) – event changed text or detail
- **id** (*int*) – change identifier

1.12.12 Capturi de ecran

Nou în versiunea 2.14.

GET /api/screenshots/
Returns a list of screenshot string information.

Vezi și:

Screenshot object attributes are documented at *GET /api/screenshots/(int:id)/*.

GET /api/screenshots/(int: *id*)/
Returns information about screenshot information.

Parameters

- **id** (*int*) – Screenshot ID

Response JSON Object

- **name** (*string*) – name of a screenshot
- **component** (*string*) – URL of a related component object
- **file_url** (*string*) – URL to download a file; see *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – link to associated source string information; see *GET /api/units/(int:id)/*

GET /api/screenshots/(int: *id*)/file/
Download the screenshot image.

Parameters

- **id** (*int*) – Screenshot ID

POST /api/screenshots/(int: *id*)/file/
Replace screenshot image.

Parameters

- **id** (*int*) – Screenshot ID

Form Parameters

- **file image** – Uploaded file

CURL example:

```
curl -X POST \
    -F image=@image.png \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/screenshots/1/file/
```

POST /api/screenshots/(int: *id*)/units/
Associate source string with screenshot.

Parameters

- **id** (*int*) – Screenshot ID

Form Parameters

- **string unit_id** – Unit ID

Response JSON Object

- **name** (*string*) – name of a screenshot
- **translation** (*string*) – URL of a related translation object
- **file_url** (*string*) – URL to download a file; see *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – link to associated source string information; see *GET /api/units/(int:id)/*

DELETE /api/screenshots/(int: *id*)/units/
int: *unit_id* Remove source string association with screenshot.

Parameters

- **id** (*int*) – Screenshot ID
- **unit_id** – Source string unit ID

POST /api/screenshots/
Creates a new screenshot.

Form Parameters

- **file image** – Uploaded file
- **string name** – Numele capturii de ecran
- **string project_slug** – Project slug
- **string component_slug** – Component slug
- **string language_code** – Codul limbii

Response JSON Object

- **name** (*string*) – name of a screenshot
- **component** (*string*) – URL of a related component object
- **file_url** (*string*) – URL to download a file; see *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – link to associated source string information; see *GET /api/units/(int:id)/*

PATCH /api/screenshots/(int: *id*)/
Edit partial information about screenshot.

Parameters

- **id** (*int*) – Screenshot ID

Response JSON Object

- **name** (*string*) – name of a screenshot
- **component** (*string*) – URL of a related component object
- **file_url** (*string*) – URL to download a file; see *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – link to associated source string information; see *GET /api/units/(int:id)/*

PUT /api/screenshots/(int: *id*)/
Edit full information about screenshot.

Parameters

- **id** (*int*) – Screenshot ID

Response JSON Object

- **name** (*string*) – name of a screenshot
- **component** (*string*) – URL of a related component object
- **file_url** (*string*) – URL to download a file; see *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – link to associated source string information; see *GET /api/units/(int:id)/*

DELETE /api/screenshots/(int: *id*)/

Delete screenshot.

Parameters

- **id** (*int*) – Screenshot ID

1.12.13 Add-ons

Nou în versiunea 4.4.1.

GET /api/addons/

Returns a list of addons.

Vezi și:

Add-on object attributes are documented at *GET /api/addons/(int:id)/*.

GET /api/addons/(int: *id*)/

Returns information about addon information.

Parameters

- **id** (*int*) – Add-on ID

Response JSON Object

- **name** (*string*) – name of an addon
- **component** (*string*) – URL of a related component object
- **configuration** (*object*) – Optional addon configuration

Vezi și:

Add-ons

POST /api/components/(string: *project*)/
string: *component*/addons/ Creates a new addon.

Parameters

- **project_slug** (*string*) – Project slug
- **component_slug** (*string*) – Component slug

Request JSON Object

- **name** (*string*) – name of an addon
- **configuration** (*object*) – Optional addon configuration

PATCH /api/addons/(int: *id*)/

Edit partial information about addon.

Parameters

- **id** (*int*) – Add-on ID

Response JSON Object

- **configuration** (*object*) – Optional addon configuration

PUT /api/addons/(int: *id*)/
Edit full information about addon.

Parameters

- **id** (*int*) – Add-on ID

Response JSON Object

- **configuration** (*object*) – Optional addon configuration

DELETE /api/addons/(int: *id*)/
Delete addon.

Parameters

- **id** (*int*) – Add-on ID

1.12.14 Liste de componente

Nou în versiunea 4.0.

GET /api/component-lists/
Returns a list of component lists.

Vezi și:

Component list object attributes are documented at *GET /api/component-lists/(str:slug)/*.

GET /api/component-lists/(str: *slug*)/
Returns information about component list.

Parameters

- **slug** (*string*) – Component list slug

Response JSON Object

- **name** (*string*) – name of a component list
- **slug** (*string*) – slug of a component list
- **show_dashboard** (*boolean*) – whether to show it on a dashboard
- **components** (*array*) – link to associated components; see *GET /api/components/(string:project)/(string:component)/*
- **auto_assign** (*array*) – automatic assignment rules

PUT /api/component-lists/(str: *slug*)/
Changes the component list parameters.

Parameters

- **slug** (*string*) – Component list slug

Request JSON Object

- **name** (*string*) – name of a component list
- **slug** (*string*) – slug of a component list
- **show_dashboard** (*boolean*) – whether to show it on a dashboard

PATCH /api/component-lists/(str: *slug*)/
Changes the component list parameters.

Parameters

- **slug** (*string*) – Component list slug

Request JSON Object

- **name** (*string*) – name of a component list
- **slug** (*string*) – slug of a component list
- **show_dashboard** (*boolean*) – whether to show it on a dashboard

DELETE /api/component-lists/(**str:** *slug*)**/**
Deletes the component list.

Parameters

- **slug** (*string*) – Component list slug

POST /api/component-lists/(**str:** *slug*)**/components/**
Associate component with a component list.

Parameters

- **slug** (*string*) – Component list slug

Form Parameters

- **string component_id** – Component ID

DELETE /api/component-lists/(**str:** *slug*)**/components/**
str: *component_slug* Disassociate a component from the component list.

Parameters

- **slug** (*string*) – Component list slug
- **component_slug** (*string*) – Component slug

1.12.15 Glosar

Schimbat în versiunea 4.5: Glossaries are now stored as regular components, translations and strings, please use respective API instead.

1.12.16 Tasks

Nou în versiunea 4.4.

GET /api/tasks/
Listing of the tasks is currently not available.

GET /api/tasks/(**str:** *uuid*)**/**
Returns information about a task

Parameters

- **uuid** (*string*) – Task UUID

Response JSON Object

- **completed** (*boolean*) – Whether the task has completed
- **progress** (*int*) – Task progress in percent
- **result** (*object*) – Task result or progress details
- **log** (*string*) – Task log

1.12.17 Metrics

GET /api/metrics/

Returns server metrics.

Response JSON Object

- **units** (*int*) – Number of units
- **units_translated** (*int*) – Number of translated units
- **users** (*int*) – Number of users
- **changes** (*int*) – Numărul de modificări
- **projects** (*int*) – Number of projects
- **components"** (*int*) – Number of components
- **translations"** (*int*) – Number of translations
- **languages"** (*int*) – Number of used languages
- **checks"** (*int*) – Number of triggered quality checks
- **configuration_errors"** (*int*) – Number of configuration errors
- **suggestions"** (*int*) – Number of pending suggestions
- **celery_queues** (*object*) – Lengths of Celery queues, see *Background tasks using Celery*
- **name** (*string*) – Configured server name

1.12.18 Cârlige de notificare

Notification hooks allow external applications to notify Weblate that the VCS repository has been updated.

You can use repository endpoints for projects, components and translations to update individual repositories; see `POST /api/projects/(string:project)/repository/` for documentation.

GET /hooks/update/(string: *project*)/string: *component*/

Învechit începând cu versiunea 2.6: Please use `POST /api/components/(string:project)/(string:component)/repository/` instead which works properly with authentication for ACL limited projects.

Triggers update of a component (pulling from VCS and scanning for translation changes).

GET /hooks/update/(string: *project*)/

Învechit începând cu versiunea 2.6: Please use `POST /api/projects/(string:project)/repository/` instead which works properly with authentication for ACL limited projects.

Triggers update of all components in a project (pulling from VCS and scanning for translation changes).

POST /hooks/github/

Special hook for handling GitHub notifications and automatically updating matching components.

Notă: GitHub includes direct support for notifying Weblate: enable Weblate service hook in repository settings and set the URL to the URL of your Weblate installation.

Vezi și:

Automatically receiving changes from GitHub For instruction on setting up GitHub integration

https://docs.github.com/en/github/extending-github/about-webhooks Generic information about GitHub Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/gitlab/

Special hook for handling GitLab notifications and automatically updating matching components.

Vezi și:

Automatically receiving changes from GitLab For instruction on setting up GitLab integration

https://docs.gitlab.com/ee/user/project/integrations/webhooks.html Generic information about GitLab Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/bitbucket/

Special hook for handling Bitbucket notifications and automatically updating matching components.

Vezi și:

Automatically receiving changes from Bitbucket For instruction on setting up Bitbucket integration

https://support.atlassian.com/bitbucket-cloud/docs/manage-webhooks/ Generic information about Bitbucket Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/pagure/

Nou în versiunea 3.3.

Special hook for handling Pagure notifications and automatically updating matching components.

Vezi și:

Automatically receiving changes from Pagure For instruction on setting up Pagure integration

https://docs.pagure.org/pagure/usage/using_webhooks.html Generic information about Pagure Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/azure/

Nou în versiunea 3.8.

Special hook for handling Azure Repos notifications and automatically updating matching components.

Vezi și:

Automatically receiving changes from Azure Repos For instruction on setting up Azure integration

https://docs.microsoft.com/en-us/azure/devops/service-hooks/services/webhooks?view=azure-devops Generic information about Azure Repos Web Hooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/gitea/

Nou în versiunea 3.9.

Special hook for handling Gitea Webhook notifications and automatically updating matching components.

Vezi și:

Automatically receiving changes from Gitea Repos For instruction on setting up Gitea integration

https://docs.gitea.io/en-us/webhooks/ Generic information about Gitea Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/gitee/

Nou în versiunea 3.9.

Special hook for handling Gitee Webhook notifications and automatically updating matching components.

Vezi și:

Automatically receiving changes from Gitee Repos For instruction on setting up Gitee integration

https://gitee.com/help/categories/40 Generic information about Gitee Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

1.12.19 Exports

Weblate provides various exports to allow you to further process the data.

GET /exports/stats/(string: *project*)/string: *component*/

Query Parameters

- **format** (*string*) – Output format: either `json` or `csv`

Invechit începând cu versiunea 2.6: Please use *GET /api/components/(string:project)/(string:component)/statistics/* and *GET /api/translations/(string:project)/(string:component)/(string:language)/statistics/* instead; it allows access to ACL controlled projects as well.

Retrieves statistics for given component in given format.

Example request:

```
GET /exports/stats/weblate/main/ HTTP/1.1
Host: example.com
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

[
    {
        "code": "cs",
        "failing": 0,
        "failing_percent": 0.0,
        "fuzzy": 0,
        "fuzzy_percent": 0.0,
        "last_author": "Michal Čihař",
        "last_change": "2012-03-28T15:07:38+00:00",
        "name": "Czech",
        "total": 436,
        "total_words": 15271,
        "translated": 436,
        "translated_percent": 100.0,
        "translated_words": 3201,
        "url": "http://hosted.weblate.org/engage/weblate/cs/",
        "url_translate": "http://hosted.weblate.org/projects/weblate/main/cs/"
    },
    {
        "code": "nl",
        "failing": 21,
```

(continues on next page)

(continuare din pagina precedentă)


```
        "failing_percent": 4.8,
        "fuzzy": 11,
        "fuzzy_percent": 2.5,
        "last_author": null,
        "last_change": null,
        "name": "Dutch",
        "total": 436,
        "total_words": 15271,
        "translated": 319,
        "translated_percent": 73.2,
        "translated_words": 3201,
        "url": "http://hosted.weblate.org/engage/weblate/nl/",
        "url_translate": "http://hosted.weblate.org/projects/weblate/main/nl/"
    },
    {
        "code": "el",
        "failing": 11,
        "failing_percent": 2.5,
        "fuzzy": 21,
        "fuzzy_percent": 4.8,
        "last_author": null,
        "last_change": null,
        "name": "Greek",
        "total": 436,
        "total_words": 15271,
        "translated": 312,
        "translated_percent": 71.6,
        "translated_words": 3201,
        "url": "http://hosted.weblate.org/engage/weblate/el/",
        "url_translate": "http://hosted.weblate.org/projects/weblate/main/el/"
    }
]
```


1.12.20 Fluxuri RSS

Changes in translations are exported in RSS feeds.

GET /exports/rss/(**string:** *project*)**/**
string: *component***/string:** *language***/** Retrieves RSS feed with recent changes for a translation.

GET /exports/rss/(**string:** *project*)**/**
string: *component***/** Retrieves RSS feed with recent changes for a component.

GET /exports/rss/(**string:** *project*)**/**
Retrieves RSS feed with recent changes for a project.

GET /exports/rss/language/(**string:** *language*)**/**
Retrieves RSS feed with recent changes for a language.

GET /exports/rss/
Retrieves RSS feed with recent changes for Weblate instance.

Vezi și:

RSS on wikipedia

1.13 Weblate Client

Nou în versiunea 2.7: There has been full wlc utility support ever since Weblate 2.7. If you are using an older version some incompatibilities with the API might occur.

1.13.1 Installation

The Weblate Client is shipped separately and includes the Python module. To use the commands below, you need to install `wlc`:

```
pip3 install wlc
```

1.13.2 Docker usage

The Weblate Client is also available as a Docker image.

The image is published on Docker Hub: https://hub.docker.com/r/weblate/wlc

Installing:

```
docker pull weblate/wlc
```

The Docker container uses Weblate's default settings and connects to the API deployed in localhost. The API URL and API_KEY can be configured through the arguments accepted by Weblate.

The command to launch the container uses the following syntax:

```
docker run --rm weblate/wlc [WLC_ARGS]
```

Exemplu:

```
docker run --rm weblate/wlc --url https://hosted.weblate.org/api/ list-projects
```

You might want to pass your *Configuration files* to the Docker container, the easiest approach is to add your current directory as `/home/weblate` volume:

```
docker run --volume $PWD:/home/weblate --rm weblate/wlc show
```

1.13.3 Getting started

The wlc configuration is stored in `~/.config/weblate` (see *Configuration files* for other locations), please create it to match your environment:

```
[weblate]
url = https://hosted.weblate.org/api/

[keys]
https://hosted.weblate.org/api/ = APIKEY
```

You can then invoke commands on the default server:

```
wlc ls
wlc commit sandbox/hello-world
```

Vezi și:

Configuration files

1.13.4 Synopsis

```
wlc [arguments] <command> [options]
```

Commands actually indicate which operation should be performed.

1.13.5 Descriere

Weblate Client is a Python library and command-line utility to manage Weblate remotely using *Weblate's REST API*. The command-line utility can be invoked as **wlc** and is built-in on `wlc`.

Arguments

The program accepts the following arguments which define output format or which Weblate instance to use. These must be entered before any command.

--format `{csv,json,text,html}`
Specify the output format.

--url `URL`
Specify the API URL. Overrides any value found in the configuration file, see *Configuration files*. The URL should end with `/api/`, for example `https://hosted.weblate.org/api/`.

--key `KEY`
Specify the API user key to use. Overrides any value found in the configuration file, see *Configuration files*. You can find your key in your profile on Weblate.

--config `PATH`
Overrides the configuration file path, see *Configuration files*.

--config-section `SECTION`
Overrides configuration file section in use, see *Configuration files*.

Commands

The following commands are available:

version
Prints the current version.

list-languages
Lists used languages in Weblate.

list-projects
Lists projects in Weblate.

list-components
Lists components in Weblate.

list-translations
Lists translations in Weblate.

show
Shows Weblate object (translation, component or project).

ls
Lists Weblate object (translation, component or project).

commit
Commits changes made in a Weblate object (translation, component or project).

pull
Pulls remote repository changes into Weblate object (translation, component or project).

push
Pushes Weblate object changes into remote repository (translation, component or project).

reset
Nou în versiunea 0.7: Supported since wlc 0.7.

Resets changes in Weblate object to match remote repository (translation, component or project).

cleanup
Nou în versiunea 0.9: Supported since wlc 0.9.

Removes any untracked changes in a Weblate object to match the remote repository (translation, component or project).

repo
Displays repository status for a given Weblate object (translation, component or project).

statistics
Displays detailed statistics for a given Weblate object (translation, component or project).

lock-status
Nou în versiunea 0.5: Supported since wlc 0.5.

Displays lock status.

lock
Nou în versiunea 0.5: Supported since wlc 0.5.

Locks component from further translation in Weblate.

unlock
Nou în versiunea 0.5: Supported since wlc 0.5.

Unlocks translation of Weblate component.

changes
Nou în versiunea 0.7: Supported since wlc 0.7 and Weblate 2.10.

Displays changes for a given object.

download
Nou în versiunea 0.7: Supported since wlc 0.7.

Downloads a translation file.

--convert
Converts file format, if unspecified no conversion happens on the server and the file is downloaded as is to the repository.

--output
Specifies file to save output in, if left unspecified it is printed to stdout.

upload
Nou în versiunea 0.9: Supported since wlc 0.9.

Uploads a translation file.

--overwrite
Overwrite existing translations upon uploading.

--input
File from which content is read, if left unspecified it is read from stdin.

Sugestie: You can get more detailed information on invoking individual commands by passing `--help`, for example: `wlc ls --help`.

1.13.6 Configuration files

.weblate, .weblate.ini, weblate.ini Schimbat în versiunea 1.6: The files with *.ini* extension are accepted as well.

Per project configuration file

C:\Users\NAME\AppData\weblate.ini Nou în versiunea 1.6.

User configuration file on Windows.

~/.config/weblate User configuration file

/etc/xdg/weblate System wide configuration file

The program follows the XDG specification, so you can adjust placement of config files by environment variables `XDG_CONFIG_HOME` or `XDG_CONFIG_DIRS`. On Windows `APPDATA` directory is preferred location for the configuration file.

Following settings can be configured in the `[weblate]` section (you can customize this by *--config-section*):

key
API KEY to access Weblate.

url
API server URL, defaults to `http://127.0.0.1:8000/api/`.

translation
Path to the default translation - component or project.

The configuration file is an INI file, for example:

```
[weblate]
url = https://hosted.weblate.org/api/
key = APIKEY
translation = weblate/application
```

Additionally API keys can be stored in the `[keys]` section:

```
[keys]
https://hosted.weblate.org/api/ = APIKEY
```

This allows you to store keys in your personal settings, while using the `.weblate` configuration in the VCS repository so that wlc knows which server it should talk to.

1.13.7 Examples

Print current program version:

```
$ wlc version
version: 0.1
```

List all projects:

```
$ wlc list-projects
name: Hello
slug: hello
url: http://example.com/api/projects/hello/
web: https://weblate.org/
web_url: http://example.com/projects/hello/
```

You can also designate what project wlc should work on:

```
$ cat .weblate
[weblate]
url = https://hosted.weblate.org/api/
translation = weblate/application

$ wlc show
branch: main
file_format: po
source_language: en
filemask: weblate/locale/*/LC_MESSAGES/django.po
git_export: https://hosted.weblate.org/git/weblate/application/
license: GPL-3.0+
license_url: https://spdx.org/licenses/GPL-3.0+
name: Application
new_base: weblate/locale/django.pot
project: weblate
repo: git://github.com/WeblateOrg/weblate.git
slug: application
template:
url: https://hosted.weblate.org/api/components/weblate/application/
vcs: git
web_url: https://hosted.weblate.org/projects/weblate/application/
```

With this setup it is easy to commit pending changes in the current project:

```
$ wlc commit
```

1.14 Weblate's Python API

1.14.1 Installation

The Python API is shipped separately, you need to install the *Weblate Client*: (wlc) to have it.

```
pip install wlc
```

1.14.2 wlc

WeblateException

exception wlc.**WeblateException**

Base class for all exceptions.

Weblate

class wlc.**Weblate**(*key=''*, *url=None*, *config=None*)

Parametrii

- **key** (*str*) – User key
- **url** (*str*) – API server URL, if not specified default is used
- **config** (wlc.config.WeblateConfig) – Configuration object, overrides any other parameters.

Access class to the API, define API key and optionally API URL.

get (*path*)

Parametrii **path** (*str*) – Request path

Tipul întors object

Performs a single API GET call.

post (*path*, ***kwargs*)

Parametrii **path** (*str*) – Request path

Tipul întors object

Performs a single API GET call.

1.14.3 wlc.config

WeblateConfig

class wlc.config.**WeblateConfig** (*section='wlc'*)

Parametrii **section** (*str*) – Configuration section to use

Configuration file parser following XDG specification.

load (*path=None*)

Parametrii **path** (*str*) – Path from which to load configuration.

Loads configuration from a file, if none is specified, it loads from the *wlc* configuration file (`~/.config/wlc`) placed in your XDG configuration path (`/etc/xdg/wlc`).

1.14.4 wlc.main

wlc.main.**main** (*settings=None*, *stdout=None*, *args=None*)

Parametrii

- **settings** (*list*) – Settings to override as list of tuples
- **stdout** (*object*) – stdout file object for printing output, uses `sys.stdout` as default
- **args** (*list*) – Command-line arguments to process, uses `sys.args` as default

Main entry point for command-line interface.

@wlc.main.**register_command** (*command*)

Decorator to register *Command* class in main parser used by *main()*.

Command

class `wlc.main.`**`Command`**(*args*, *config*, *stdout=None*)

Main class for invoking commands.

CAPITOLUL 2

Administrator docs

2.1 Configuration instructions

2.1.1 Installing Weblate

Installing using Docker

With dockerized Weblate deployment you can get your personal Weblate instance up and running in seconds. All of Weblate's dependencies are already included. PostgreSQL is set up as the default database.

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

Notă: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

Installation

The following examples assume you have a working Docker environment, with `docker-compose` installed. Please check the Docker documentation for instructions.

1. Clone the weblate-docker repo:

```
git clone https://github.com/WeblateOrg/docker-compose.git weblate-docker
cd weblate-docker
```

2. Create a `docker-compose.override.yml` file with your settings. See *Docker environment variables* for full list of environment variables.

```
version: '3'
services:
  weblate:
    ports:
      - 80:8080
    environment:
      WEBLATE_EMAIL_HOST: smtp.example.com
      WEBLATE_EMAIL_HOST_USER: user
      WEBLATE_EMAIL_HOST_PASSWORD: pass
      WEBLATE_SERVER_EMAIL: weblate@example.com
      WEBLATE_DEFAULT_FROM_EMAIL: weblate@example.com
      WEBLATE_SITE_DOMAIN: weblate.example.com
      WEBLATE_ADMIN_PASSWORD: password for the admin user
      WEBLATE_ADMIN_EMAIL: weblate.admin@example.com
```

Notă: If *`WEBLATE_ADMIN_PASSWORD`* is not set, the admin user is created with a random password shown on first startup.

The provided example makes Weblate listen on port 80, edit the port mapping in the `docker-compose.override.yml` file to change it.

3. Start Weblate containers:

```
docker-compose up
```

Enjoy your Weblate deployment, it's accessible on port 80 of the `weblate` container.

Schimbat în versiunea 2.15-2: The setup has changed recently, priorly there was separate web server container, since 2.15-2 the web server is embedded in the Weblate container.

Schimbat în versiunea 3.7.1-6: In July 2019 (starting with the 3.7.1-6 tag), the containers are not running as a root user. This has changed the exposed port from 80 to 8080.

Vezi și:

Invoking management commands

Choosing Docker hub tag

You can use following tags on Docker hub, see https://hub.docker.com/r/weblate/weblate/tags/ for full list of available ones.

Tag name	Descriere	Use case
latest	Weblate stable release, matches latest tagged release	Rolling updates in a production environment
<VERSION>-<PATCH>	Weblate stable release	Well defined deploy in a production environment
edge	Weblate stable release with development changes in the Docker container (for example updated dependencies)	Rolling updates in a staging environment
edge-<DATE>-<SHA>	Weblate stable release with development changes in the Docker container (for example updated dependencies)	Well defined deploy in a staging environment
bleeding	Development version Weblate from Git	Rollling updates to test upcoming Weblate features
bleeding-<DATE>-<SHA>	Development version Weblate from Git	Well defined deploy to test upcoming Weblate features

Every image is tested by our CI before it gets published, so even the *bleeding* version should be quite safe to use.

Docker container with HTTPS support

Please see *Installation* for generic deployment instructions, this section only mentions differences compared to it.

Using own SSL certificates

Nou în versiunea 3.8-3.

In case you have own SSL certificate you want to use, simply place the files into the Weblate data volume (see *Docker container volumes*):

- `ssl/fullchain.pem` containing the certificate including any needed CA certificates
- `ssl/privkey.pem` containing the private key

Both of these files must be owned by the same user as the one starting the docker container and have file mask set to `600` (readable and writable only by the owning user).

Additionally, Weblate container will now accept SSL connections on port 4443, you will want to include the port forwarding for HTTPS in docker compose override:

```
version: '3'
services:
  weblate:
    ports:
      - 80:8080
      - 443:4443
```

If you already host other sites on the same server, it is likely ports `80` and `443` are used by a reverse proxy, such as NGINX. To pass the HTTPS connection from NGINX to the docker container, you can use the following configuration:

```
server {
    listen 443;
    listen [::]:443;
```

(continues on next page)

(continuare din pagina precedentă)

```
    server_name <SITE_URL>;
    ssl_certificate /etc/letsencrypt/live/<SITE>/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/<SITE>/privkey.pem;

    location / {
            proxy_set_header HOST $host;
            proxy_set_header X-Forwarded-Proto https;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Host $server_name;
            proxy_pass https://127.0.0.1:<EXPOSED_DOCKER_PORT>;
    }
}
```

Replace `<SITE_URL>`, `<SITE>` and `<EXPOSED_DOCKER_PORT>` with actual values from your environment.

Automatic SSL certificates using Let's Encrypt

In case you want to use Let's Encrypt automatically generated SSL certificates on public installation, you need to add a reverse HTTPS proxy an additional Docker container, https-portal will be used for that. This is made use of in the `docker-compose-https.yml` file. Then create a `docker-compose-https.override.yml` file with your settings:

```
version: '3'
services:
  weblate:
    environment:
      WEBLATE_EMAIL_HOST: smtp.example.com
      WEBLATE_EMAIL_HOST_USER: user
      WEBLATE_EMAIL_HOST_PASSWORD: pass
      WEBLATE_SITE_DOMAIN: weblate.example.com
      WEBLATE_ADMIN_PASSWORD: password for admin user
  https-portal:
    environment:
      DOMAINS: 'weblate.example.com -> http://weblate:8080'
```

Whenever invoking **`docker-compose`** you need to pass both files to it, and then do:

```
docker-compose -f docker-compose-https.yml -f docker-compose-https.override.yml␣
↪build
docker-compose -f docker-compose-https.yml -f docker-compose-https.override.yml up
```

Upgrading the Docker container

Usually it is good idea to only update the Weblate container and keep the PostgreSQL container at the version you have, as upgrading PostgreSQL is quite painful and in most cases does not bring many benefits.

You can do this by sticking with the existing docker-compose and just pull the latest images and then restart:

```
docker-compose stop
docker-compose pull
docker-compose up
```

The Weblate database should be automatically migrated on first startup, and there should be no need for additional manual actions.

Notă: Upgrades across 3.0 are not supported by Weblate. If you are on 2.x series and want to upgrade to 3.x, first upgrade to the latest 3.0.1-x (at time of writing this it is the `3.0.1-7`) image, which will do the migration and then continue upgrading to newer versions.

You might also want to update the `docker-compose` repository, though it's not needed in most case. Please beware of PostgreSQL version changes in this case as it's not straightforward to upgrade the database, see GitHub issue for more info.

Admin sign in

After container setup, you can sign in as *admin* user with password provided in *WEBLATE_ADMIN_PASSWORD*, or a random password generated on first start if that was not set.

To reset *admin* password, restart the container with *WEBLATE_ADMIN_PASSWORD* set to new password.

Vezi și:

WEBLATE_ADMIN_PASSWORD, *WEBLATE_ADMIN_NAME*, *WEBLATE_ADMIN_EMAIL*

Number of processes and memory consumption

The number of worker processes for both uWSGI and Celery is determined automatically based on number of CPUs. This works well for most cloud virtual machines as these typically have few CPUs and good amount of memory.

In case you have a lot of CPU cores and hit out of memory issues, try reducing number of workers:

```
environment:
  WEBLATE_WORKERS: 2
```

You can also fine-tune individual worker categories:

```
environment:
  UWSGI_WORKERS: 4
  CELERY_MAIN_OPTIONS: --concurrency 2
  CELERY_NOTIFY_OPTIONS: --concurrency 1
  CELERY_TRANSLATE_OPTIONS: --concurrency 1
```

Vezi și:

WEBLATE_WORKERS, *CELERY_MAIN_OPTIONS*, *CELERY_NOTIFY_OPTIONS*, *CELERY_MEMORY_OPTIONS*, *CELERY_TRANSLATE_OPTIONS*, *CELERY_BACKUP_OPTIONS*, *CELERY_BEAT_OPTIONS*, *UWSGI_WORKERS*

Scaling horizontally

Nou în versiunea 4.6.

Atenționare: This feature is a technology preview.

You can run multiple Weblate containers to scale the service horizontally. The `/app/data` volume has to be shared by all containers, it is recommended to use cluster filesystem such as GlusterFS for this. The `/app/cache` volume should be separate for each container.

Each Weblate container has defined role using *WEBLATE_SERVICE* environment variable. Please follow carefully the documentation as some of the services should be running just once in the cluster and the ordering of the services matters as well.

You can find example setup in the `docker-compose` repo as docker-compose-split.yml.

Docker environment variables

Many of Weblate's *Configurație* can be set in the Docker container using environment variables:

Generic settings

WEBLATE_DEBUG

Configures Django debug mode using `DEBUG`.

Example:

```
environment:
  WEBLATE_DEBUG: 1
```

Vezi și:

Disable debug mode

WEBLATE_LOGLEVEL

Configures the logging verbosity.

WEBLATE_SITE_TITLE

Changes the site-title shown in the header of all pages.

WEBLATE_SITE_DOMAIN

Configures the site domain. This parameter is required.

Vezi și:

Set correct site domain, `SITE_DOMAIN`

WEBLATE_ADMIN_NAME

WEBLATE_ADMIN_EMAIL

Configures the site-admin's name and e-mail. It is used for both `ADMINS` setting and creating *admin* user (see `WEBLATE_ADMIN_PASSWORD` for more info on that).

Example:

```
environment:
  WEBLATE_ADMIN_NAME: Weblate admin
  WEBLATE_ADMIN_EMAIL: noreply@example.com
```

Vezi și:

Admin sign in, *Properly configure admins*, `ADMINS`

WEBLATE_ADMIN_PASSWORD

Sets the password for the *admin* user.

- If not set and *admin* user does not exist, it is created with a random password shown on first container startup.
- If not set and *admin* user exists, no action is performed.
- If set the *admin* user is adjusted on every container startup to match `WEBLATE_ADMIN_PASSWORD`, `WEBLATE_ADMIN_NAME` and `WEBLATE_ADMIN_EMAIL`.

> **Atenționare:** It might be a security risk to store password in the configuration file. Consider using this variable only for initial setup (or let Weblate generate random password on initial startup) or for password recovery.

Vezi și:

Admin sign in, *WEBLATE_ADMIN_PASSWORD*, *WEBLATE_ADMIN_PASSWORD_FILE*, *WEBLATE_ADMIN_NAME*, *WEBLATE_ADMIN_EMAIL*

WEBLATE_ADMIN_PASSWORD_FILE
Sets the path to a file containing the password for the *admin* user.

Vezi și:

WEBLATE_ADMIN_PASSWORD

WEBLATE_SERVER_EMAIL

WEBLATE_DEFAULT_FROM_EMAIL
Configures the address for outgoing e-mails.

Vezi și:

Configure e-mail sending

WEBLATE_CONTACT_FORM
Configures contact form behavior, see *CONTACT_FORM*.

WEBLATE_ALLOWED_HOSTS
Configures allowed HTTP hostnames using *ALLOWED_HOSTS*.

Defaults to `*` which allows all hostnames.

Example:

```
environment:
  WEBLATE_ALLOWED_HOSTS: weblate.example.com,example.com
```

Vezi și:

ALLOWED_HOSTS, *Allowed hosts setup*, *Set correct site domain*

WEBLATE_REGISTRATION_OPEN
Configures whether registrations are open by toggling *REGISTRATION_OPEN*.

Example:

```
environment:
  WEBLATE_REGISTRATION_OPEN: 0
```

WEBLATE_REGISTRATION_ALLOW_BACKENDS
Configure which authentication methods can be used to create new account via *REGISTRATION_ALLOW_BACKENDS*.

Example:

```
environment:
  WEBLATE_REGISTRATION_OPEN: 0
  WEBLATE_REGISTRATION_ALLOW_BACKENDS: azuread-oauth2,azuread-tenant-
↪oauth2
```

WEBLATE_TIME_ZONE
Configures the used time zone in Weblate, see `TIME_ZONE`.

Notă: To change the time zone of the Docker container itself, use the `TZ` environment variable.

Example:

```
environment:
  WEBLATE_TIME_ZONE: Europe/Prague
```

WEBLATE_ENABLE_HTTPS

Makes Weblate assume it is operated behind a reverse HTTPS proxy, it makes Weblate use HTTPS in e-mail and API links or set secure flags on cookies.

Sugestie: Please see *ENABLE_HTTPS* documentation for possible caveats.

Notă: This does not make the Weblate container accept HTTPS connections, you need to configure that as well, see *Docker container with HTTPS support* for examples.

Example:

```
environment:
  WEBLATE_ENABLE_HTTPS: 1
```

Vezi și:

ENABLE_HTTPS *Set correct site domain*, *WEBLATE_SECURE_PROXY_SSL_HEADER*

WEBLATE_IP_PROXY_HEADER

Lets Weblate fetch the IP address from any given HTTP header. Use this when using a reverse proxy in front of the Weblate container.

Enables *IP_BEHIND_REVERSE_PROXY* and sets *IP_PROXY_HEADER*.

Notă: The format must conform to Django's expectations. Django transforms raw HTTP header names as follows:

- converts all characters to uppercase
- replaces any hyphens with underscores
- prepends `HTTP_` prefix

So `X-Forwarded-For` would be mapped to `HTTP_X_FORWARDED_FOR`.

Example:

```
environment:
  WEBLATE_IP_PROXY_HEADER: HTTP_X_FORWARDED_FOR
```

WEBLATE_SECURE_PROXY_SSL_HEADER

A tuple representing a HTTP header/value combination that signifies a request is secure. This is needed when Weblate is running behind a reverse proxy doing SSL termination which does not pass standard HTTPS headers.

Example:

```
environment:
  WEBLATE_SECURE_PROXY_SSL_HEADER: HTTP_X_FORWARDED_PROTO,https
```

Vezi și:

`SECURE_PROXY_SSL_HEADER`

WEBLATE_REQUIRE_LOGIN

Enables *REQUIRE_LOGIN* to enforce authentication on whole Weblate.

Example:

```
environment:
  WEBLATE_REQUIRE_LOGIN: 1
```

WEBLATE_LOGIN_REQUIRED_URLS_EXCEPTIONS

WEBLATE_ADD_LOGIN_REQUIRED_URLS_EXCEPTIONS

WEBLATE_REMOVE_LOGIN_REQUIRED_URLS_EXCEPTIONS

Adds URL exceptions for authentication required for the whole Weblate installation using *LOGIN_REQUIRED_URLS_EXCEPTIONS*.

You can either replace whole settings, or modify default value using `ADD` and `REMOVE` variables.

WEBLATE_GOOGLE_ANALYTICS_ID

Configures ID for Google Analytics by changing *GOOGLE_ANALYTICS_ID*.

WEBLATE_GITHUB_USERNAME

Configures GitHub username for GitHub pull-requests by changing *GITHUB_USERNAME*.

Vezi și:

GitHub

WEBLATE_GITHUB_TOKEN

Nou în versiunea 4.3.

Configures GitHub personal access token for GitHub pull-requests via API by changing *GITHUB_TOKEN*.

Vezi și:

GitHub

WEBLATE_GITLAB_USERNAME

Configures GitLab username for GitLab merge-requests by changing *GITLAB_USERNAME*

Vezi și:

GitLab

WEBLATE_GITLAB_TOKEN

Configures GitLab personal access token for GitLab merge-requests via API by changing *GITLAB_TOKEN*

Vezi și:

GitLab

WEBLATE_PAGURE_USERNAME

Configures Pagure username for Pagure merge-requests by changing *PAGURE_USERNAME*

Vezi și:

Pagure

WEBLATE_PAGURE_TOKEN

Configures Pagure personal access token for Pagure merge-requests via API by changing *PAGURE_TOKEN*

Vezi și:

Pagure

WEBLATE_SIMPLIFY_LANGUAGES

Configures the language simplification policy, see *SIMPLIFY_LANGUAGES*.

WEBLATE_DEFAULT_ACCESS_CONTROL

Configures the default *Controlul accesului* for new projects, see *DEFAULT_ACCESS_CONTROL*.

WEBLATE_DEFAULT_RESTRICTED_COMPONENT

Configures the default value for *Restricted access* for new components, see *DEFAULT_RESTRICTED_COMPONENT*.

WEBLATE_DEFAULT_TRANSLATION_PROPAGATION

Configures the default value for *Permiteți propagarea traducerii* for new components, see *DEFAULT_TRANSLATION_PROPAGATION*.

WEBLATE_DEFAULT_COMMITER_EMAIL

Configures *DEFAULT_COMMITER_EMAIL*.

WEBLATE_DEFAULT_COMMITER_NAME
Configures *DEFAULT_COMMITER_NAME*.

WEBLATE_DEFAULT_SHARED_TM
Configures *DEFAULT_SHARED_TM*.

WEBLATE_AKISMET_API_KEY
Configures the Akismet API key, see *AKISMET_API_KEY*.

WEBLATE_GPG_IDENTITY
Configures GPG signing of commits, see *WEBLATE_GPG_IDENTITY*.

Vezi și:

Signing Git commits with GnuPG

WEBLATE_URL_PREFIX
Configures URL prefix where Weblate is running, see *URL_PREFIX*.

WEBLATE_SILENCED_SYSTEM_CHECKS
Configures checks which you do not want to be displayed, see `SILENCED_SYSTEM_CHECKS`.

WEBLATE_CSP_SCRIPT_SRC

WEBLATE_CSP_IMG_SRC

WEBLATE_CSP_CONNECT_SRC

WEBLATE_CSP_STYLE_SRC

WEBLATE_CSP_FONT_SRC
Allows to customize `Content-Security-Policy` HTTP header.

Vezi și:

Content security policy, *CSP_SCRIPT_SRC*, *CSP_IMG_SRC*, *CSP_CONNECT_SRC*, *CSP_STYLE_SRC*, *CSP_FONT_SRC*

WEBLATE_LICENSE_FILTER
Configures *LICENSE_FILTER*.

WEBLATE_LICENSE_REQUIRED
Configures *LICENSE_REQUIRED*

WEBLATE_WEBSITE_REQUIRED
Configures *WEBSITE_REQUIRED*

WEBLATE_HIDE_VERSION
Configures *HIDE_VERSION*.

WEBLATE_BASIC_LANGUAGES
Configures *BASIC_LANGUAGES*.

WEBLATE_DEFAULT_AUTO_WATCH
Configures *DEFAULT_AUTO_WATCH*.

WEBLATE_RATELIMIT_ATTEMPTS

WEBLATE_RATELIMIT_LOCKOUT

WEBLATE_RATELIMIT_WINDOW
Nou în versiunea 4.6.

Configures rate limiter.

Sugestie: You can set configuration for any rate limiter scopes. To do that add `WEBLATE_` prefix to any of setting described in *Rate limiting*.

Vezi și:

Rate limiting, `RATELIMIT_ATTEMPTS`, `RATELIMIT_WINDOW`, `RATELIMIT_LOCKOUT`

WEBLATE_ENABLE_AVATARS
Nou în versiunea 4.6.1.

Configures `ENABLE_AVATARS`.

Machine translation settings

Sugestie: Configuring API key for a service automatically configures it in `MT_SERVICES`.

WEBLATE_MT_APERTIUM_APY
Enables *Apertium* machine translation and sets `MT_APERTIUM_APY`

WEBLATE_MT_AWS_REGION

WEBLATE_MT_AWS_ACCESS_KEY_ID

WEBLATE_MT_AWS_SECRET_ACCESS_KEY
Configures *AWS* machine translation.

```
environment:
  WEBLATE_MT_AWS_REGION: us-east-1
  WEBLATE_MT_AWS_ACCESS_KEY_ID: AKIAIOSFODNN7EXAMPLE
  WEBLATE_MT_AWS_SECRET_ACCESS_KEY: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

WEBLATE_MT_DEEPL_KEY
Enables *DeepL* machine translation and sets `MT_DEEPL_KEY`

WEBLATE_MT_DEEPL_API_URL
Configures *DeepL* API version to use, see `MT_DEEPL_API_URL`.

WEBLATE_MT_LIBRETRANSLATE_KEY
Enables *LibreTranslate* machine translation and sets `MT_LIBRETRANSLATE_KEY`

WEBLATE_MT_LIBRETRANSLATE_API_URL
Configures *LibreTranslate* API instance to use, see `MT_LIBRETRANSLATE_API_URL`.

WEBLATE_MT_GOOGLE_KEY
Enables *Google Translate* and sets `MT_GOOGLE_KEY`

WEBLATE_MT_MICROSOFT_COGNITIVE_KEY
Enables *Microsoft Cognitive Services Translator* and sets `MT_MICROSOFT_COGNITIVE_KEY`

WEBLATE_MT_MICROSOFT_ENDPOINT_URL
Sets `MT_MICROSOFT_ENDPOINT_URL`, please note this is supposed to contain domain name only.

WEBLATE_MT_MICROSOFT_REGION
Sets `MT_MICROSOFT_REGION`

WEBLATE_MT_MICROSOFT_BASE_URL
Sets `MT_MICROSOFT_BASE_URL`

WEBLATE_MT_MODERNMT_KEY
Enables *ModernMT* and sets `MT_MODERNMT_KEY`.

WEBLATE_MT_MYMEMORY_ENABLED

Enables *MyMemory* machine translation and sets `MT_MYMEMORY_EMAIL` to `WEBLATE_ADMIN_EMAIL`.

Example:

```
environment:
  WEBLATE_MT_MYMEMORY_ENABLED: 1
```

WEBLATE_MT_GLOSBE_ENABLED

Enables *Glosbe* machine translation.

```
environment:
  WEBLATE_MT_GLOSBE_ENABLED: 1
```

WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED

Enables *Microsoft Terminology Service* machine translation.

```
environment:
  WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED: 1
```

WEBLATE_MT_SAP_BASE_URL

WEBLATE_MT_SAP_SANDBOX_APIKEY

WEBLATE_MT_SAP_USERNAME

WEBLATE_MT_SAP_PASSWORD

WEBLATE_MT_SAP_USE_MT

Configures *SAP Translation Hub* machine translation.

```
environment:
    WEBLATE_MT_SAP_BASE_URL: "https://example.hana.ondemand.com/translationhub/
↪api/v1/"
    WEBLATE_MT_SAP_USERNAME: "user"
    WEBLATE_MT_SAP_PASSWORD: "password"
    WEBLATE_MT_SAP_USE_MT: 1
```

Authentication settings

LDAP

WEBLATE_AUTH_LDAP_SERVER_URI

WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE

WEBLATE_AUTH_LDAP_USER_ATTR_MAP

WEBLATE_AUTH_LDAP_BIND_DN

WEBLATE_AUTH_LDAP_BIND_PASSWORD

WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS

WEBLATE_AUTH_LDAP_USER_SEARCH

WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER

WEBLATE_AUTH_LDAP_USER_SEARCH_UNION

WEBLATE_AUTH_LDAP_USER_SEARCH_UNION_DELIMITER

LDAP authentication configuration.

Example for direct bind:

```
environment:
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE: uid=%(user)s,ou=People,dc=example,dc=net
  # map weblate 'full_name' to ldap 'name' and weblate 'email' attribute to
↪'mail' ldap attribute.
```

(continues on next page)

(continuare din pagina precedentă)

```
  # another example that can be used with OpenLDAP: 'full_name:cn,email:mail'
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
```

Example for search and bind:

```
environment:
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_BIND_DN: CN=ldap,CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_BIND_PASSWORD: password
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
  WEBLATE_AUTH_LDAP_USER_SEARCH: CN=Users,DC=example,DC=com
```

Example for union search and bind:

```
environment:
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_BIND_DN: CN=ldap,CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_BIND_PASSWORD: password
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
  WEBLATE_AUTH_LDAP_USER_SEARCH_UNION: ou=users,dc=example,
↪dc=com|ou=otherusers,dc=example,dc=com
```

Example with search and bind against Active Directory:

```
environment:
  WEBLATE_AUTH_LDAP_BIND_DN: CN=ldap,CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_BIND_PASSWORD: password
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS: 0
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
  WEBLATE_AUTH_LDAP_USER_SEARCH: CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER: (sAMAccountName=%(user)s)
```

Vezi și:

Autentificare LDAP

GitHub

WEBLATE_SOCIAL_AUTH_GITHUB_KEY

WEBLATE_SOCIAL_AUTH_GITHUB_SECRET

Enables *Autentificarea GitHub*.

Bitbucket

WEBLATE_SOCIAL_AUTH_BITBUCKET_KEY

WEBLATE_SOCIAL_AUTH_BITBUCKET_SECRET

Enables *Autentificarea Bitbucket*.

Facebook

WEBLATE_SOCIAL_AUTH_FACEBOOK_KEY

WEBLATE_SOCIAL_AUTH_FACEBOOK_SECRET
Enables *Facebook OAuth 2*.

Google

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_KEY

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_DOMAINS

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_EMAILS
Enables *Google OAuth 2*.

GitLab

WEBLATE_SOCIAL_AUTH_GITLAB_KEY

WEBLATE_SOCIAL_AUTH_GITLAB_SECRET

WEBLATE_SOCIAL_AUTH_GITLAB_API_URL
Enables *GitLab OAuth 2*.

Azure Active Directory

WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_KEY

WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET
Enables Azure Active Directory authentication, see *Microsoft Azure directoare active*.

Azure Active Directory with Tenant support

WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY

WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECRET

WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENANT_ID
Enables Azure Active Directory authentication with Tenant support, see *Microsoft Azure directoare active*.

Keycloak

WEBLATE_SOCIAL_AUTH_KEYCLOAK_KEY

WEBLATE_SOCIAL_AUTH_KEYCLOAK_SECRET

WEBLATE_SOCIAL_AUTH_KEYCLOAK_PUBLIC_KEY

WEBLATE_SOCIAL_AUTH_KEYCLOAK_ALGORITHM

WEBLATE_SOCIAL_AUTH_KEYCLOAK_AUTHORIZATION_URL

WEBLATE_SOCIAL_AUTH_KEYCLOAK_ACCESS_TOKEN_URL
Enables Keycloak authentication, see documentation.

Linux vendors

You can enable authentication using Linux vendors authentication services by setting following variables to any value.

WEBLATE_SOCIAL_AUTH_FEDORA

WEBLATE_SOCIAL_AUTH_OPENSUSE

WEBLATE_SOCIAL_AUTH_UBUNTU

Slack

WEBLATE_SOCIAL_AUTH_SLACK_KEY

SOCIAL_AUTH_SLACK_SECRET
Enables Slack authentication, see *Slack*.

SAML

Self-signed SAML keys are automatically generated on first container startup. In case you want to use own keys, place the certificate and private key in `/app/data/ssl/saml.crt` and `/app/data/ssl/saml.key`.

WEBLATE_SAML_IDP_ENTITY_ID

WEBLATE_SAML_IDP_URL

WEBLATE_SAML_IDP_X509CERT
SAML Identity Provider settings, see *Autentificare SAML*.

Other authentication settings

WEBLATE_NO_EMAIL_AUTH
Disables e-mail authentication when set to any value.

PostgreSQL database setup

The database is created by `docker-compose.yml`, so these settings affect both Weblate and PostgreSQL containers.

Vezi și:

Database setup for Weblate

POSTGRES_PASSWORD
PostgreSQL password.

POSTGRES_PASSWORD_FILE
Path to the file containing the PostgreSQL password. Use as an alternative to POSTGRES_PASSWORD.

POSTGRES_USER
PostgreSQL username.

POSTGRES_DATABASE
PostgreSQL database name.

POSTGRES_HOST
PostgreSQL server hostname or IP address. Defaults to `database`.

POSTGRES_PORT
PostgreSQL server port. Defaults to none (uses the default value).

POSTGRES_SSL_MODE
Configure how PostgreSQL handles SSL in connection to the server, for possible choices see SSL Mode Descriptions

POSTGRES_ALTER_ROLE
Configures name of role to alter during migrations, see *Configuring Weblate to use PostgreSQL*.

POSTGRES_CONN_MAX_AGE
Nou în versiunea 4.8.1.

The lifetime of a database connection, as an integer of seconds. Use 0 to close database connections at the end of each request (this is the default behavior).

Enabling connection persistence will typically, cause more open connection to the database. Please adjust your database configuration prior enabling.

> **Atenționare:** This is currently experimental, not widely tested and not supported by the Weblate team.

Vezi și:

`CONN_MAX_AGE`, Persistent connections

Database backup settings

Vezi și:

Date descărcate pentru copii de rezervă

WEBLATE_DATABASE_BACKUP
Configures the daily database dump using *`DATABASE_BACKUP`*. Defaults to `plain`.

Caching server setup

Using Redis is strongly recommended by Weblate and you have to provide a Redis instance when running Weblate in Docker.

Vezi și:

Enable caching

REDIS_HOST
The Redis server hostname or IP address. Defaults to `cache`.

REDIS_PORT
The Redis server port. Defaults to `6379`.

REDIS_DB
The Redis database number, defaults to `1`.

REDIS_PASSWORD
The Redis server password, not used by default.

REDIS_TLS
Enables using SSL for Redis connection.

REDIS_VERIFY_SSL
Can be used to disable SSL certificate verification for Redis connection.

Email server setup

To make outgoing e-mail work, you need to provide a mail server.

Example TLS configuration:

```
environment:
    WEBLATE_EMAIL_HOST: smtp.example.com
    WEBLATE_EMAIL_HOST_USER: user
    WEBLATE_EMAIL_HOST_PASSWORD: pass
```

Example SSL configuration:

```
environment:
    WEBLATE_EMAIL_HOST: smtp.example.com
    WEBLATE_EMAIL_PORT: 465
    WEBLATE_EMAIL_HOST_USER: user
    WEBLATE_EMAIL_HOST_PASSWORD: pass
    WEBLATE_EMAIL_USE_TLS: 0
    WEBLATE_EMAIL_USE_SSL: 1
```

Vezi și:

Configuring outgoing e-mail

WEBLATE_EMAIL_HOST

Mail server hostname or IP address.

Vezi și:

WEBLATE_EMAIL_PORT, *WEBLATE_EMAIL_USE_SSL*, *WEBLATE_EMAIL_USE_TLS*, EMAIL_HOST

WEBLATE_EMAIL_PORT

Mail server port, defaults to 25.

Vezi și:

EMAIL_PORT

WEBLATE_EMAIL_HOST_USER

E-mail authentication user.

Vezi și:

EMAIL_HOST_USER

WEBLATE_EMAIL_HOST_PASSWORD

E-mail authentication password.

Vezi și:

EMAIL_HOST_PASSWORD

WEBLATE_EMAIL_HOST_PASSWORD_FILE

Path to the file containing the e-mail authentication password.

Vezi și:

WEBLATE_EMAIL_HOST_PASSWORD

WEBLATE_EMAIL_USE_SSL

Whether to use an implicit TLS (secure) connection when talking to the SMTP server. In most e-mail documentation, this type of TLS connection is referred to as SSL. It is generally used on port 465. If you are experiencing problems, see the explicit TLS setting *WEBLATE_EMAIL_USE_TLS*.

Vezi și:

WEBLATE_EMAIL_PORT, *WEBLATE_EMAIL_USE_TLS*, EMAIL_USE_SSL

WEBLATE_EMAIL_USE_TLS
Whether to use a TLS (secure) connection when talking to the SMTP server. This is used for explicit TLS connections, generally on port 587 or 25. If you are experiencing connections that hang, see the implicit TLS setting *WEBLATE_EMAIL_USE_SSL*.

Vezi și:

WEBLATE_EMAIL_PORT, *WEBLATE_EMAIL_USE_SSL*, EMAIL_USE_TLS

WEBLATE_EMAIL_BACKEND
Configures Django back-end to use for sending e-mails.

Vezi și:

Configure e-mail sending, EMAIL_BACKEND

Site integration

WEBLATE_GET_HELP_URL
Configures *GET_HELP_URL*.

WEBLATE_STATUS_URL
Configures *STATUS_URL*.

WEBLATE_LEGAL_URL
Configures *LEGAL_URL*.

WEBLATE_PRIVACY_URL
Configures *PRIVACY_URL*.

Error reporting

It is recommended to collect errors from the installation systematically, see *Collecting error reports*.

To enable support for Rollbar, set the following:

ROLLBAR_KEY
Your Rollbar post server access token.

ROLLBAR_ENVIRONMENT
Your Rollbar environment, defaults to `production`.

To enable support for Sentry, set following:

SENTRY_DSN
Your Sentry DSN.

SENTRY_ENVIRONMENT
Your Sentry Environment (optional).

Localization CDN

WEBLATE_LOCALIZE_CDN_URL

WEBLATE_LOCALIZE_CDN_PATH
Nou în versiunea 4.2.1.

Configuration for *CDN de localizare JavaScript*.

The *WEBLATE_LOCALIZE_CDN_PATH* is path within the container. It should be stored on the persistent volume and not in the transient storage.

One of possibilities is storing that inside the Weblate data dir:

```
environment:
  WEBLATE_LOCALIZE_CDN_URL: https://cdn.example.com/
  WEBLATE_LOCALIZE_CDN_PATH: /app/data/l10n-cdn
```

Notă: You are responsible for setting up serving of the files generated by Weblate, it only does stores the files in configured location.

Vezi și:

weblate-cdn, *LOCALIZE_CDN_URL*, *LOCALIZE_CDN_PATH*

Changing enabled apps, checks, addons or autofixes

Nou în versiunea 3.8-5.

The built-in configuration of enabled checks, addons or autofixes can be adjusted by the following variables:

WEBLATE_ADD_APPS

WEBLATE_REMOVE_APPS

WEBLATE_ADD_CHECK

WEBLATE_REMOVE_CHECK

WEBLATE_ADD_AUTOFIX

WEBLATE_REMOVE_AUTOFIX

WEBLATE_ADD_ADDONS

WEBLATE_REMOVE_ADDONS

Example:

```
environment:
  WEBLATE_REMOVE_AUTOFIX: weblate.trans.autofixes.whitespace.
↪SameBookendingWhitespace
  WEBLATE_ADD_ADDONS: customize.addons.MyAddon,customize.addons.OtherAddon
```

Vezi și:

CHECK_LIST, *AUTOFIX_LIST*, *WEBLATE_ADDONS*, INSTALLED_APPS

Container settings

WEBLATE_WORKERS

Nou în versiunea 4.6.1.

Base number of worker processes running in the container. When not set it is determined automatically on container startup based on number of CPU cores available.

It is used to determine *CELERY_MAIN_OPTIONS*, *CELERY_NOTIFY_OPTIONS*, *CELERY_MEMORY_OPTIONS*, *CELERY_TRANSLATE_OPTIONS*, *CELERY_BACKUP_OPTIONS*, *CELERY_BEAT_OPTIONS*, and *UWSGI_WORKERS*. You can use these settings to fine-tune.

CELERY_MAIN_OPTIONS

CELERY_NOTIFY_OPTIONS

CELERY_MEMORY_OPTIONS

CELERY_TRANSLATE_OPTIONS

CELERY_BACKUP_OPTIONS

CELERY_BEAT_OPTIONS

These variables allow you to adjust Celery worker options. It can be useful to adjust concurrency (`--concurrency 16`) or use different pool implementation (`--pool=gevent`).

By default, the number of concurrent workers is based on *WEBLATE_WORKERS*.

Example:

```
environment:
  CELERY_MAIN_OPTIONS: --concurrency 16
```

Vezi și:

Celery worker options, *Background tasks using Celery*

UWSGI_WORKERS

Configure how many uWSGI workers should be executed.

It defaults to *WEBLATE_WORKERS*.

Example:

```
environment:
  UWSGI_WORKERS: 32
```

WEBLATE_SERVICE

Defines which services should be executed inside the container. Use this for *Scaling horizontally*.

Following services are defined:

celery-beat Celery task scheduler, only one instance should be running. This container is also responsible for the database structure migrations and it should be started prior others.

celery-backup Celery worker for backups, only one instance should be running.

celery-celery Generic Celery worker.

celery-memory Translation memory Celery worker.

celery-notify Notifications Celery worker.

celery-translate Automatic translation Celery worker.

web Web server.

Docker container volumes

There are two volumes (data and cache) exported by the Weblate container. The other service containers (PostgreSQL or Redis) have their data volumes as well, but those are not covered by this document.

The data volume is used to store Weblate persistent data such as cloned repositories or to customize Weblate installation.

The placement of the Docker volume on host system depends on your Docker configuration, but usually it is stored in `/var/lib/docker/volumes/weblate-docker_weblate-data/_data/` (the path consist of name of your docker-compose directory, container, and volume names). In the container it is mounted as `/app/data`.

The cache volume is mounted as `/app/cache` and is used to store static files. Its content is recreated on container startup and the volume can be mounted using ephemeral filesystem such as *tmpfs*.

When creating the volumes manually, the directories should be owned by UID 1000 as that is user used inside the container.

Vezi și:

Docker volumes documentation

Further configuration customization

You can further customize Weblate installation in the data volume, see *Docker container volumes*.

Custom configuration files

You can additionally override the configuration in `/app/data/settings-override.py` (see *Docker container volumes*). This is executed at the end of built-in settings, after all environment settings are loaded, and you can adjust or override them.

Replacing logo and other static files

Nou în versiunea 3.8-5.

The static files coming with Weblate can be overridden by placing into `/app/data/python/customize/static` (see *Docker container volumes*). For example creating `/app/data/python/customize/static/favicon.ico` will replace the favicon.

Sugestie: The files are copied to the corresponding location upon container startup, so a restart of Weblate is needed after changing the content of the volume.

Alternatively you can also include own module (see *Customizing Weblate*) and add it as separate volume to the Docker container, for example:

```
weblate:
  volumes:
    - weblate-data:/app/data
    - ./weblate_customization/weblate_customization:/app/data/python/weblate_
↪customization
  environment:
    WEBLATE_ADD_APPS: weblate_customization
```

Adding own Python modules

Nou în versiunea 3.8-5.

You can place own Python modules in `/app/data/python/` (see *Docker container volumes*) and they can be then loaded by Weblate, most likely by using *Custom configuration files*.

Vezi și:

Customizing Weblate

Installing on Debian and Ubuntu

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

Notă: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

Installation

System requirements

Install the dependencies needed to build the Python modules (see *Software requirements*):

```
apt install \
   libxml2-dev libxslt-dev libfreetype6-dev libjpeg-dev libz-dev libyaml-dev \
   libcairo-dev gir1.2-pango-1.0 libgirepository1.0-dev libacl1-dev libssl-dev \
   build-essential python3-gdbm python3-dev python3-pip python3-virtualenv␣
↪virtualenv git
```

Install wanted optional dependencies depending on features you intend to use (see *Optional dependencies*):

```
apt install tesseract-ocr libtesseract-dev libleptonica-dev
```

Optionally install software for running production server, see *Running server*, *Database setup for Weblate*, *Background tasks using Celery*. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
apt install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with ``mod_wsgi``
apt install apache2 libapache2-mod-wsgi-py3

# Caching backend: Redis
apt install redis-server

# Database server: PostgreSQL
apt install postgresql postgresql-contrib

# SMTP server
apt install exim4
```

Python modules

Sugestie: We're using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:

```
virtualenv --python=python3 ~/weblate-env
```

2. Activate the virtualenv for Weblate:

```
. ~/weblate-env/bin/activate
```

3. Install Weblate including all dependencies:

```
pip install Weblate
```

4. Install database driver:

```
pip install psycopg2-binary
```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check *Optional dependencies*):

```
pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
```

Configuring Weblate

Notă: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **`weblate`** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`.
2. Adjust the values in the new `settings.py` file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see *Adjusting configuration*.
3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check *Database setup for Weblate* for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Running server* and *Serving static files*):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see *Compressing client assets*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Background tasks using Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see *Running server* for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

- You can now access Weblate on `http://localhost:8000/`.
- Login with admin credentials obtained during installation or register with new users.
- You can now run Weblate commands using **`weblate`** command when Weblate virtualenv is active, see *Management commands*.
- You can stop the test server with Ctrl+C.
- Review potential issues with your installation either on `/manage/performance/` URL or using **`weblate check --deploy`**, see *Production setup*.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *Project configuration* for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See *Component configuration* for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see *Supported file formats* for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on SUSE and openSUSE

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

Notă: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

Installation

System requirements

Install the dependencies needed to build the Python modules (see *Software requirements*):

```
zypper install \
   libxslt-devel libxml2-devel freetype-devel libjpeg-devel zlib-devel libyaml-
↪devel \
   cairo-devel typelib-1_0-Pango-1_0 gobject-introspection-devel libacl-devel \
   python3-pip python3-virtualenv python3-devel git
```

Install wanted optional dependencies depending on features you intend to use (see *Optional dependencies*):

```
zypper install tesseract-ocr tesseract-devel leptonica-devel
```

Optionally install software for running production server, see *Running server*, *Database setup for Weblate*, *Background tasks using Celery*. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
zypper install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with ``mod_wsgi``
zypper install apache2 apache2-mod_wsgi

# Caching backend: Redis
zypper install redis-server

# Database server: PostgreSQL
zypper install postgresql postgresql-contrib

# SMTP server
zypper install postfix
```

Python modules

Sugestie: We're using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:

```
. ~/weblate-env/bin/activate
```

3. Install Weblate including all dependencies:

```
pip install Weblate
```

4. Install database driver:

```
pip install psycopg2-binary
```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check *Optional dependencies*):

```
pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
```

Configuring Weblate

Notă: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **`weblate`** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`.
2. Adjust the values in the new `settings.py` file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see *Adjusting configuration*.
3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check *Database setup for Weblate* for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Running server* and *Serving static files*):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see *Compressing client assets*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Background tasks using Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see *Running server* for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

- You can now access Weblate on `http://localhost:8000/`.
- Login with admin credentials obtained during installation or register with new users.
- You can now run Weblate commands using **`weblate`** command when Weblate virtualenv is active, see *Management commands*.
- You can stop the test server with Ctrl+C.
- Review potential issues with your installation either on `/manage/performance/` URL or using **`weblate check --deploy`**, see *Production setup*.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *Project configuration* for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See *Component configuration* for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see *Supported file formats* for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on RedHat, Fedora and CentOS

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

Notă: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

Installation

System requirements

Install the dependencies needed to build the Python modules (see *Software requirements*):

```
dnf install \
   libxslt-devel libxml2-devel freetype-devel libjpeg-devel zlib-devel libyaml-
↪devel \
   cairo-devel pango-devel gobject-introspection-devel libacl-devel \
   python3-pip python3-virtualenv python3-devel git
```

Install wanted optional dependencies depending on features you intend to use (see *Optional dependencies*):

```
dnf install tesseract-langpack-eng tesseract-devel leptonica-devel
```

Optionally install software for running production server, see *Running server*, *Database setup for Weblate*, *Background tasks using Celery*. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
dnf install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with ``mod_wsgi``
dnf install apache2 apache2-mod_wsgi

# Caching backend: Redis
dnf install redis

# Database server: PostgreSQL
dnf install postgresql postgresql-contrib

# SMTP server
dnf install postfix
```

Python modules

Sugestie: We're using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:

   ```
   . ~/weblate-env/bin/activate
   ```

3. Install Weblate including all dependencies:

   ```
   pip install Weblate
   ```

4. Install database driver:

   ```
   pip install psycopg2-binary
   ```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check *Optional dependencies*):

```
pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
```

Configuring Weblate

Notă: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **weblate** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`.
2. Adjust the values in the new `settings.py` file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see *Adjusting configuration*.
3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check *Database setup for Weblate* for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Running server* and *Serving static files*):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see *Compressing client assets*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Background tasks using Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see *Running server* for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

- You can now access Weblate on `http://localhost:8000/`.
- Login with admin credentials obtained during installation or register with new users.
- You can now run Weblate commands using **weblate** command when Weblate virtualenv is active, see *Management commands*.
- You can stop the test server with Ctrl+C.

- Review potential issues with your installation either on `/manage/performance/` URL or using **`weblate check --deploy`**, see *Production setup*.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *Project configuration* for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See *Component configuration* for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see *Supported file formats* for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on macOS

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

Notă: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

Installation

System requirements

Install the dependencies needed to build the Python modules (see *Software requirements*):

```
brew install python pango cairo gobject-introspection libffi glib libyaml
pip3 install virtualenv
```

Make sure pip will be able to find the `libffi` version provided by homebrew — this will be needed during the installation build step.

```
export PKG_CONFIG_PATH="/usr/local/opt/libffi/lib/pkgconfig"
```

Install wanted optional dependencies depending on features you intend to use (see *Optional dependencies*):

```
brew install tesseract
```

Optionally install software for running production server, see *Running server*, *Database setup for Weblate*, *Background tasks using Celery*. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
brew install nginx uwsgi

# Web server option 2: Apache with ``mod_wsgi``
brew install httpd

# Caching backend: Redis
brew install redis

# Database server: PostgreSQL
brew install postgresql
```

Python modules

Sugestie: We're using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:

   ```
   . ~/weblate-env/bin/activate
   ```

3. Install Weblate including all dependencies:

   ```
   pip install Weblate
   ```

4. Install database driver:

   ```
   pip install psycopg2-binary
   ```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check *Optional dependencies*):

   ```
   pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
   ```

Configuring Weblate

Notă: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **`weblate`** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`.

2. Adjust the values in the new `settings.py` file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see *Adjusting configuration*.
3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check *Database setup for Weblate* for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Running server* and *Serving static files*):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see *Compressing client assets*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Background tasks using Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see *Running server* for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

- You can now access Weblate on `http://localhost:8000/`.
- Login with admin credentials obtained during installation or register with new users.
- You can now run Weblate commands using **`weblate`** command when Weblate virtualenv is active, see *Management commands*.
- You can stop the test server with Ctrl+C.
- Review potential issues with your installation either on `/manage/performance/` URL or using **`weblate check --deploy`**, see *Production setup*.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *Project configuration* for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See *Component configuration* for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see *Supported file formats* for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing from sources

1. Please follow the installation instructions for your system first:
   - *Installing on Debian and Ubuntu*
   - *Installing on SUSE and openSUSE*
   - *Installing on RedHat, Fedora and CentOS*
2. Grab the latest Weblate sources using Git (or download a tarball and unpack that):

   ```
   git clone https://github.com/WeblateOrg/weblate.git weblate-src
   ```

   Alternatively you can use released archives. You can download them from our website <https://weblate.org/>. Those downloads are cryptographically signed, please see *Verifying release signatures*.
3. Install current Weblate code into the virtualenv:

   ```
   . ~/weblate-env/bin/activate
   pip install -e weblate-src
   ```
4. Copy `weblate/settings_example.py` to `weblate/settings.py`.
5. Adjust the values in the new `settings.py` file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see *Adjusting configuration*.
6. Create the database used by Weblate, see *Database setup for Weblate*.
7. Build Django tables, static files and initial data (see *Filling up the database* and *Serving static files*):

   ```
   weblate migrate
   weblate collectstatic
   weblate compress
   weblate compilemessages
   ```

   Notă: This step should be repeated whenever you update the repository.

Installing on OpenShift

With the OpenShift Weblate template you can get your personal Weblate instance up and running in seconds. All of Weblate's dependencies are already included. PostgreSQL is set up as the default database and persistent volume claims are used.

You can find the template at <https://github.com/WeblateOrg/openshift/>.

Installation

The following examples assume you have a working OpenShift v3.x environment, with `oc` client tool installed. Please check the OpenShift documentation for instructions.

The `template.yml` is suited for running all components in OpenShift. There is also `template-external-postgresql.yml` which does not start a PostgreSQL server and allows you to configure external PostgreSQL server.

Web Console

Copy the raw content from template.yml and import them into your project, then use the `Create` button in the OpenShift web console to create your application. The web console will prompt you for the values for all of the parameters used by the template.

CLI

To upload the Weblate template to your current project's template library, pass the `template.yml` file with the following command:

```
$ oc create -f https://raw.githubusercontent.com/WeblateOrg/openshift/main/
↪template.yml \
   -n <PROJECT>
```

The template is now available for selection using the web console or the CLI.

Parameters

The parameters that you can override are listed in the parameters section of the template. You can list them with the CLI by using the following command and specifying the file to be used:

```
$ oc process --parameters -f https://raw.githubusercontent.com/WeblateOrg/
↪openshift/main/template.yml

# If the template is already uploaded
$ oc process --parameters -n <PROJECT> weblate
```

Provisioning

You can also use the CLI to process templates and use the configuration that is generated to create objects immediately.

```
$ oc process -f https://raw.githubusercontent.com/WeblateOrg/openshift/main/
↪template.yml \
    -p APPLICATION_NAME=weblate \
    -p WEBLATE_VERSION=4.3.1-1 \
    -p WEBLATE_SITE_DOMAIN=weblate.app-openshift.example.com \
    -p POSTGRESQL_IMAGE=docker-registry.default.svc:5000/openshift/postgresql:9.6 \
    -p REDIS_IMAGE=docker-registry.default.svc:5000/openshift/redis:3.2 \
    | oc create -f
```

The Weblate instance should be available after successful migration and deployment at the specified `WEBLATE_SITE_DOMAIN` parameter.

After container setup, you can sign in as *admin* user with password provided in `WEBLATE_ADMIN_PASSWORD`, or a random password generated on first start if that was not set.

To reset *admin* password, restart the container with `WEBLATE_ADMIN_PASSWORD` set to new password in the respective `Secret`.

Eliminate

```
$ oc delete all -l app=<APPLICATION_NAME>
$ oc delete configmap -l app= <APPLICATION_NAME>
$ oc delete secret -l app=<APPLICATION_NAME>
# ATTTENTION! The following command is only optional and will permanently delete␣
↪all of your data.
$ oc delete pvc -l app=<APPLICATION_NAME>

$ oc delete all -l app=weblate \
    && oc delete secret -l app=weblate \
    && oc delete configmap -l app=weblate \
    && oc delete pvc -l app=weblate
```

Configurație

By processing the template a respective `ConfigMap` will be created and which can be used to customize the Weblate image. The `ConfigMap` is directly mounted as environment variables and triggers a new deployment every time it is changed. For further configuration options, see *Docker environment variables* for full list of environment variables.

Installing on Kubernetes

Notă: This guide is looking for contributors experienced with Kubernetes to cover the setup in more details.

With the Kubernetes Helm chart you can get your personal Weblate instance up and running in seconds. All of Weblate's dependencies are already included. PostgreSQL is set up as the default database and persistent volume claims are used.

You can find the chart at <https://github.com/WeblateOrg/helm/> and it can be displayed at <https://artifacthub.io/packages/helm/weblate/weblate>.

Installation

```
helm repo add weblate https://helm.weblate.org
helm install my-release weblate/weblate
```

Configurație

For further configuration options, see *Docker environment variables* for full list of environment variables.

Depending on your setup and experience, choose an appropriate installation method for you:

- *Installing using Docker*, recommended for production setups.
- Virtualenv installation, recommended for production setups:
  - *Installing on Debian and Ubuntu*
  - *Installing on SUSE and openSUSE*
  - *Installing on RedHat, Fedora and CentOS*
  - *Installing on macOS*
- *Installing from sources*, recommended for development.
- *Installing on OpenShift*

- *Installing on Kubernetes*

2.1.2 Software requirements

Operating system

Weblate is known to work on Linux, FreeBSD and macOS. Other Unix like systems will most likely work too.

Weblate is not supported on Windows. But it may still work and patches are happily accepted.

Other services

Weblate is using other services for its operation. You will need at least following services running:

- PostgreSQL database server, see *Database setup for Weblate*.
- Redis server for cache and tasks queue, see *Background tasks using Celery*.
- SMTP server for outgoing e-mail, see *Configuring outgoing e-mail*.

Python dependencies

Weblate is written in Python and supports Python 3.6 or newer. You can install dependencies using pip or from your distribution packages, full list is available in `requirements.txt`.

Most notable dependencies:

Django https://www.djangoproject.com/

Celery https://docs.celeryproject.org/

Translate Toolkit https://toolkit.translatehouse.org/

translation-finder https://github.com/WeblateOrg/translation-finder

Python Social Auth https://python-social-auth.readthedocs.io/

Django REST Framework https://www.django-rest-framework.org/

Optional dependencies

Following modules are necessary for some Weblate features. You can find all of them in `requirements-optional.txt`.

`Mercurial` (optional for Mercurial repositories support) https://www.mercurial-scm.org/

`phply` (optional for PHP support) https://github.com/viraptor/phply

`tesserocr` (optional for screenshots OCR) https://github.com/sirfz/tesserocr

`akismet` (optional for suggestion spam protection) https://github.com/ubernostrum/akismet

`ruamel.yaml` (optional for *YAML files*) https://pypi.org/project/ruamel.yaml/

`Zeep` (optional for *Microsoft Terminology Service*) https://docs.python-zeep.org/

`aeidon` (optional for *Subtitle files*) https://pypi.org/project/aeidon/

Database backend dependencies

Weblate supports PostgreSQL, MySQL and MariaDB, see *Database setup for Weblate* and backends documentation for more details.

Other system requirements

The following dependencies have to be installed on the system:

`Git` https://git-scm.com/

Pango, Cairo and related header files and gir introspection data https://cairographics.org/, https://pango.gnome.org/, see *Pango and Cairo*

`git-review` (optional for Gerrit support) https://pypi.org/project/git-review/

`git-svn` (optional for Subversion support) https://git-scm.com/docs/git-svn

`tesseract` and its data (optional for screenshots OCR) https://github.com/tesseract-ocr/tesseract

`licensee` (optional for detecting license when creating component) https://github.com/licensee/licensee

Build-time dependencies

To build some of the *Python dependencies* you might need to install their dependencies. This depends on how you install them, so please consult individual packages for documentation. You won't need those if using prebuilt `Wheels` while installing using `pip` or when you use distribution packages.

Pango and Cairo

Schimbat în versiunea 3.7.

Weblate uses Pango and Cairo for rendering bitmap widgets (see promotion) and rendering checks (see *Gestionarea fonturilor*). To properly install Python bindings for those you need to install system libraries first - you need both Cairo and Pango, which in turn need GLib. All those should be installed with development files and GObject introspection data.

2.1.3 Verifying release signatures

Weblate release are cryptographically signed by the releasing developer. Currently this is Michal Čihař. Fingerprint of his PGP key is:

```
63CB 1DF1 EF12 CF2A C0EE 5A32 9C27 B313 42B7 511D
```

and you can get more identification information from <https://keybase.io/nijel>.

You should verify that the signature matches the archive you have downloaded. This way you can be sure that you are using the same code that was released. You should also verify the date of the signature to make sure that you downloaded the latest version.

Each archive is accompanied with `.asc` files which contain the PGP signature for it. Once you have both of them in the same folder, you can verify the signature:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Ne 3. března 2019, 16:43:15 CET
gpg:                using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: Can't check signature: public key not found
```

As you can see GPG complains that it does not know the public key. At this point you should do one of the following steps:

- Use *wkd* to download the key:

```
$ gpg --auto-key-locate wkd --locate-keys michal@cihar.com
pub   rsa4096 2009-06-17 [SC]
      63CB1DF1EF12CF2AC0EE5A329C27B31342B7511D
uid           [ultimate] Michal Čihař <michal@cihar.com>
uid           [ultimate] Michal Čihař <nijel@debian.org>
uid           [ultimate] [jpeg image of size 8848]
uid           [ultimate] Michal Čihař (Braiins) <michal.cihar@braiins.cz>
sub   rsa4096 2009-06-17 [E]
sub   rsa4096 2015-09-09 [S]
```

- Download the keyring from Michal's server, then import it with:

```
$ gpg --import wmxth3chu9jfxdxywj1skpmhsj311mzm
```

- Download and import the key from one of the key servers:

```
$ gpg --keyserver hkp://pgp.mit.edu --recv-keys␣
↪87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: key 9C27B31342B7511D: "Michal Čihař <michal@cihar.com>" imported
gpg: Total number processed: 1
gpg:               unchanged: 1
```

This will improve the situation a bit - at this point you can verify that the signature from the given key is correct but you still can not trust the name used in the key:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Ne 3. března 2019, 16:43:15 CET
gpg:                using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: Good signature from "Michal Čihař <michal@cihar.com>" [ultimate]
gpg:                 aka "Michal Čihař <nijel@debian.org>" [ultimate]
gpg:                 aka "[jpeg image of size 8848]" [ultimate]
gpg:                 aka "Michal Čihař (Braiins) <michal.cihar@braiins.cz>"␣
↪[ultimate]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 63CB 1DF1 EF12 CF2A C0EE  5A32 9C27 B313 42B7 511D
```

The problem here is that anybody could issue the key with this name. You need to ensure that the key is actually owned by the mentioned person. The GNU Privacy Handbook covers this topic in the chapter Validating other keys on your public keyring. The most reliable method is to meet the developer in person and exchange key fingerprints, however you can also rely on the web of trust. This way you can trust the key transitively though signatures of others, who have met the developer in person.

Once the key is trusted, the warning will not occur:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Sun Mar  3 16:43:15 2019 CET
gpg:                using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: Good signature from "Michal Čihař <michal@cihar.com>" [ultimate]
gpg:                 aka "Michal Čihař <nijel@debian.org>" [ultimate]
gpg:                 aka "[jpeg image of size 8848]" [ultimate]
gpg:                 aka "Michal Čihař (Braiins) <michal.cihar@braiins.cz>"␣
↪[ultimate]
```

Should the signature be invalid (the archive has been changed), you would get a clear error regardless of the fact that the key is trusted or not:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: Signature made Sun Mar  3 16:43:15 2019 CET
gpg:                using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: BAD signature from "Michal Čihař <michal@cihar.com>" [ultimate]
```

2.1.4 Filesystem permissions

The Weblate process needs to be able to read and write to the directory where it keeps data - `DATA_DIR`. All files within this directory should be owned and writable by the user running all Weblate processes (typically WSGI and Celery, see *Running server* and *Background tasks using Celery*).

The default configuration places them in the same tree as the Weblate sources, however you might prefer to move these to a better location such as: `/var/lib/weblate`.

Weblate tries to create these directories automatically, but it will fail when it does not have permissions to do so.

You should also take care when running *Management commands*, as they should be ran under the same user as Weblate itself is running, otherwise permissions on some files might be wrong.

In the Docker container, all files in the `/app/data` volume have to be owned by weblate user inside the container (UID 1000).

Vezi și:

Serving static files

2.1.5 Database setup for Weblate

It is recommended to run Weblate with a PostgreSQL database server.

Vezi și:

Use a powerful database engine, Databases, *Migrating from other databases to PostgreSQL*

PostgreSQL

PostgreSQL is usually the best choice for Django-based sites. It's the reference database used for implementing Django database layer.

Notă: Weblate uses trigram extension which has to be installed separately in some cases. Look for `postgresql-contrib` or a similarly named package.

Vezi și:

PostgreSQL notes

Creating a database in PostgreSQL

It is usually a good idea to run Weblate in a separate database, and separate user account:

```
# If PostgreSQL was not installed before, set the main password
sudo -u postgres psql postgres -c "\password postgres"

# Create a database user called "weblate"
sudo -u postgres createuser --superuser --pwprompt weblate

# Create the database "weblate" owned by "weblate"
sudo -u postgres createdb -E UTF8 -O weblate weblate
```

Sugestie: If you don't want to make the Weblate user a superuser in PostgreSQL, you can omit that. In that case you will have to perform some of the migration steps manually as a PostgreSQL superuser in schema Weblate will use:

```
CREATE EXTENSION IF NOT EXISTS pg_trgm WITH SCHEMA weblate;
```

Configuring Weblate to use PostgreSQL

The `settings.py` snippet for PostgreSQL:

```
DATABASES = {
    "default": {
        # Database engine
        "ENGINE": "django.db.backends.postgresql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Name of role to alter to set parameters in PostgreSQL,
        # use in case role name is different than user used for authentication.
        # "ALTER_ROLE": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "database.example.com",
        # Set to empty string for default
        "PORT": "",
    }
}
```

The database migration performs ALTER ROLE on database role used by Weblate. In most cases the name of the role matches username. In more complex setups the role name is different than username and you will get error about non-existing role during the database migration (`psycopg2.errors.UndefinedObject: role "weblate@hostname" does not exist`). This is known to happen with Azure Database for PostgreSQL, but it's not limited to this environment. Please set `ALTER_ROLE` to change name of the role Weblate should alter during the database migration.

MySQL and MariaDB

Sugestie: Some Weblate features will perform better with *PostgreSQL*. This includes searching and translation memory, which both utilize full-text features in the database and PostgreSQL implementation is superior.

Weblate can be also used with MySQL or MariaDB, please see MySQL notes and MariaDB notes for caveats using Django with those. Because of the limitations it is recommended to use *PostgreSQL* for new installations.

Weblate requires MySQL at least 5.7.8 or MariaDB at least 10.2.7.

Following configuration is recommended for Weblate:

- Use the `utf8mb4` charset to allow representation of higher Unicode planes (for example emojis).
- Configure the server with `innodb_large_prefix` to allow longer indices on text fields.
- Set the isolation level to `READ COMMITTED`.
- The SQL mode should be set to `STRICT_TRANS_TABLES`.

MySQL 8.x, MariaDB 10.5.x or newer have reasonable default configuration so that no server tweaking should be necessary and all what is needed can be configured on the client side.

Below is an example `/etc/my.cnf.d/server.cnf` for a server with 8 GB of RAM. These settings should be sufficient for most installs. MySQL and MariaDB have tunables that will increase the performance of your server that are considered not necessary unless you are planning on having large numbers of concurrent users accessing the system. See the various vendors documentation on those details.

It is absolutely critical to reduce issues when installing that the setting `innodb_file_per_table` is set properly and MySQL/MariaDB restarted before you start your Weblate install.

```
[mysqld]
character-set-server = utf8mb4
character-set-client = utf8mb4
collation-server = utf8mb4_unicode_ci

datadir=/var/lib/mysql

log-error=/var/log/mariadb/mariadb.log

innodb_large_prefix=1
innodb_file_format=Barracuda
innodb_file_per_table=1
innodb_buffer_pool_size=2G
sql_mode=STRICT_TRANS_TABLES
```

Sugestie: In case you are getting `#1071 - Specified key was too long; max key length is 767 bytes` error, please update your configuration to include the `innodb` settings above and restart your install.

Sugestie: In case you are getting `#2006 - MySQL server has gone away` error, configuring `CONN_MAX_AGE` might help.

Configuring Weblate to use MySQL/MariaDB

The `settings.py` snippet for MySQL and MariaDB:

```
DATABASES = {
    "default": {
        # Database engine
        "ENGINE": "django.db.backends.mysql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "127.0.0.1",
        # Set to empty string for default
        "PORT": "3306",
        # In case you wish to use additional
        # connection options
        "OPTIONS": {},
    }
}
```

You should also create the `weblate` user account in MySQL or MariaDB before you begin the install. Use the commands below to achieve that:

```
GRANT ALL ON weblate.* to 'weblate'@'localhost' IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
```

2.1.6 Other configurations

Configuring outgoing e-mail

Weblate sends out e-mails on various occasions - for account activation and on various notifications configured by users. For this it needs access to an SMTP server.

The mail server setup is configured using these settings: `EMAIL_HOST`, `EMAIL_HOST_PASSWORD`, `EMAIL_USE_TLS`, `EMAIL_USE_SSL`, `EMAIL_HOST_USER` and `EMAIL_PORT`. Their names are quite self-explanatory, but you can find more info in the Django documentation.

Sugestie: In case you get error about not supported authentication (for example `SMTP AUTH extension not supported by server`), it is most likely caused by using insecure connection and server refuses to authenticate this way. Try enabling `EMAIL_USE_TLS` in such case.

Vezi și:

Not receiving e-mails from Weblate, *Configuring outgoing e-mail in Docker container*

Running behind reverse proxy

Several features in Weblate rely on being able to get client IP address. This includes *Rate limiting*, *Spam protection* or *Jurnal de audit*.

In default configuration Weblate parses IP address from `REMOTE_ADDR` which is set by the WSGI handler.

In case you are running a reverse proxy, this field will most likely contain its address. You need to configure Weblate to trust additional HTTP headers and parse the IP address from these. This can not be enabled by default as it would allow IP address spoofing for installations not using a reverse proxy. Enabling *`IP_BEHIND_REVERSE_PROXY`* might be enough for the most usual setups, but you might need to adjust *`IP_PROXY_HEADER`* and *`IP_PROXY_OFFSET`* as well.

Vezi și:

Spam protection, *Rate limiting*, *Jurnal de audit*, *`IP_BEHIND_REVERSE_PROXY`*, *`IP_PROXY_HEADER`*, *`IP_PROXY_OFFSET`*, `SECURE_PROXY_SSL_HEADER`

HTTP proxy

Weblate does execute VCS commands and those accept proxy configuration from environment. The recommended approach is to define proxy settings in `settings.py`:

```
import os

os.environ["http_proxy"] = "http://proxy.example.com:8080"
os.environ["HTTPS_PROXY"] = "http://proxy.example.com:8080"
```

Vezi și:

Proxy Environment Variables

2.1.7 Adjusting configuration

Vezi și:

Sample configuration

Copy `weblate/settings_example.py` to `weblate/settings.py` and adjust it to match your setup. You will probably want to adjust the following options: `ADMINS`

List of site administrators to receive notifications when something goes wrong, for example notifications on failed merges, or Django errors.

Vezi și:

`ADMINS`

`ALLOWED_HOSTS`

You need to set this to list the hosts your site is supposed to serve. For example:

```
ALLOWED_HOSTS = ["demo.weblate.org"]
```

Alternatively you can include wildcard:

```
ALLOWED_HOSTS = ["*"]
```

Vezi și:

`ALLOWED_HOSTS`, *`WEBLATE_ALLOWED_HOSTS`*, *Allowed hosts setup*

`SESSION_ENGINE`

Configure how your sessions will be stored. In case you keep the default database backend engine, you should schedule: **`weblate clearsessions`** to remove stale session data from the database.

If you are using Redis as cache (see *Enable caching*) it is recommended to use it for sessions as well:

```
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
```

Vezi și:

Configuring the session engine, `SESSION_ENGINE`

`DATABASES`

Connectivity to database server, please check Django's documentation for more details.

Vezi și:

Database setup for Weblate, `DATABASES`, Databases

`DEBUG`

Disable this for any production server. With debug mode enabled, Django will show backtraces in case of error to users, when you disable it, errors will be sent per e-mail to `ADMINS` (see above).

Debug mode also slows down Weblate, as Django stores much more info internally in this case.

Vezi și:

`DEBUG`

`DEFAULT_FROM_EMAIL`

E-mail sender address for outgoing e-mail, for example registration e-mails.

Vezi și:

`DEFAULT_FROM_EMAIL`

`SECRET_KEY`

Key used by Django to sign some info in cookies, see *Django secret key* for more info.

Vezi și:

`SECRET_KEY`

`SERVER_EMAIL`

E-mail used as sender address for sending e-mails to the administrator, for example notifications on failed merges.

Vezi și:

`SERVER_EMAIL`

2.1.8 Filling up the database

After your configuration is ready, you can run `weblate migrate` to create the database structure. Now you should be able to create translation projects using the admin interface.

In case you want to run an installation non interactively, you can use `weblate migrate --noinput`, and then create an admin user using `createadmin` command.

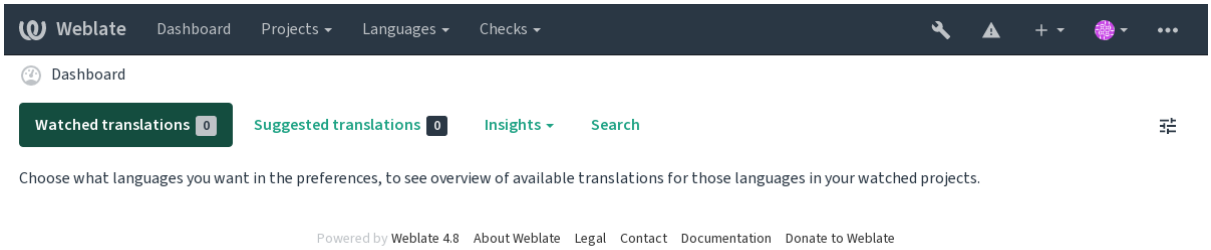
Once you are done, you should also check the *Performance report* in the admin interface, which will give you hints of potential non optimal configuration on your site.

Vezi și:

Configurație, *List of privileges*

2.1.9 Production setup

For a production setup you should carry out adjustments described in the following sections. The most critical settings will trigger a warning, which is indicated by an exclamation mark in the top bar if signed in as a superuser:



It is also recommended to inspect checks triggered by Django (though you might not need to fix all of them):

```
weblate check --deploy
```

You can also review the very same checklist from the *Interfața de gestionare*.

Vezi și:

Deployment checklist

Disable debug mode

Disable Django's debug mode (*DEBUG*) by:

```
DEBUG = False
```

With debug mode on, Django stores all executed queries and shows users backtraces of errors, which is not desired in a production setup.

Vezi și:

Adjusting configuration

Properly configure admins

Set the correct admin addresses to the *ADMINS* setting to defining who will receive e-mails in case something goes wrong on the server, for example:

```
ADMINS = (("Your Name", "your_email@example.com"),)
```

Vezi și:

Adjusting configuration

Set correct site domain

Adjust site name and domain in the admin interface, otherwise links in RSS or registration e-mails will not work. This is configured using *SITE_DOMAIN* which should contain site domain name.

Schimbat în versiunea 4.2: Prior to the 4.2 release the Django sites framework was used instead, please see The "sites" framework.

Vezi și:

Allowed hosts setup, *Correctly configure HTTPS* *SITE_DOMAIN*, *WEBLATE_SITE_DOMAIN*, *ENABLE_HTTPS*

Correctly configure HTTPS

It is strongly recommended to run Weblate using the encrypted HTTPS protocol. After enabling it, you should set *ENABLE_HTTPS* in the settings:

```
ENABLE_HTTPS = True
```

Sugestie: You might want to set up HSTS as well, see SSL/HTTPS for more details.

Vezi și:

ENABLE_HTTPS, *Allowed hosts setup*, *Set correct site domain*

Set properly SECURE_HSTS_SECONDS

If your site is served over SSL, you have to consider setting a value for `SECURE_HSTS_SECONDS` in the `settings.py` to enable HTTP Strict Transport Security. By default it's set to 0 as shown below.

```
SECURE_HSTS_SECONDS = 0
```

If set to a non-zero integer value, the `django.middleware.security.SecurityMiddleware` sets the HTTP Strict Transport Security header on all responses that do not already have it.

> **Atenționare:** Setting this incorrectly can irreversibly (for some time) break your site. Read the HTTP Strict Transport Security documentation first.

Use a powerful database engine

- Please use PostgreSQL for a production environment, see *Database setup for Weblate* for more info.
- Use adjacent location for running the database server, otherwise the networking performance or reliability might ruin your Weblate experience.
- Check the database server performance or tweak its configuration, for example using PGTune.

Vezi și:

Database setup for Weblate, *Migrating from other databases to PostgreSQL*, *Adjusting configuration*, Databases

Enable caching

If possible, use Redis from Django by adjusting the `CACHES` configuration variable, for example:

```
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/0",
        # If redis is running on same host as Weblate, you might
        # want to use unix sockets instead:
        # 'LOCATION': 'unix:///var/run/redis/redis.sock?db=0',
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
            "PARSER_CLASS": "redis.connection.HiredisParser",
        },
    }
}
```

Sugestie: In case you change Redis settings for the cache, you might need to adjust them for Celery as well, see *Background tasks using Celery*.

Vezi și:

Avatar caching, Django's cache framework

Avatar caching

In addition to caching of Django, Weblate performs caching of avatars. It is recommended to use a separate, file-backed cache for this purpose:

```
CACHES = {
    "default": {
        # Default caching backend setup, see above
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "unix:///var/run/redis/redis.sock?db=0",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
            "PARSER_CLASS": "redis.connection.HiredisParser",
        },
    },
    "avatar": {
        "BACKEND": "django.core.cache.backends.filebased.FileBasedCache",
        "LOCATION": os.path.join(DATA_DIR, "avatar-cache"),
        "TIMEOUT": 604800,
        "OPTIONS": {
            "MAX_ENTRIES": 1000,
        },
    },
}
```

Vezi și:

ENABLE_AVATARS, *AVATAR_URL_PREFIX*, *Avatars*, *Enable caching*, Django's cache framework

Configure e-mail sending

Weblate needs to send out e-mails on several occasions, and these e-mails should have a correct sender address, please configure *SERVER_EMAIL* and *DEFAULT_FROM_EMAIL* to match your environment, for example:

```
SERVER_EMAIL = "admin@example.org"
DEFAULT_FROM_EMAIL = "weblate@example.org"
```

Notă: To disable sending e-mails by Weblate set `EMAIL_BACKEND` to `django.core.mail.backends.dummy.EmailBackend`.

This will disable *all* e-mail delivery including registration or password reset e-mails.

Vezi și:

Adjusting configuration, *Configuring outgoing e-mail*, `EMAIL_BACKEND`, `DEFAULT_FROM_EMAIL`, `SERVER_EMAIL`

Allowed hosts setup

Django requires *ALLOWED_HOSTS* to hold a list of domain names your site is allowed to serve, leaving it empty will block any requests.

In case this is not configured to match your HTTP server, you will get errors like `Invalid HTTP_HOST header: '1.1.1.1'. You may need to add '1.1.1.1' to ALLOWED_HOSTS.`

Sugestie: On Docker container, this is available as *WEBLATE_ALLOWED_HOSTS*.

Vezi și:

ALLOWED_HOSTS, *WEBLATE_ALLOWED_HOSTS*, *Set correct site domain*

Django secret key

The *SECRET_KEY* setting is used by Django to sign cookies, and you should really generate your own value rather than using the one from the example setup.

You can generate a new key using `weblate/examples/generate-secret-key` shipped with Weblate.

Vezi și:

SECRET_KEY

Home directory

Schimbat în versiunea 2.1: This is no longer required, Weblate now stores all its data in *DATA_DIR*.

The home directory for the user running Weblate should exist and be writable by this user. This is especially needed if you want to use SSH to access private repositories, but Git might need to access this directory as well (depending on the Git version you use).

You can change the directory used by Weblate in `settings.py`, for example to set it to `configuration` directory under the Weblate tree:

```
os.environ["HOME"] = os.path.join(BASE_DIR, "configuration")
```

Notă: On Linux, and other UNIX like systems, the path to user's home directory is defined in `/etc/passwd`. Many distributions default to a non-writable directory for users used for serving web content (such as `apache`, `www-data` or `wwwrun`), so you either have to run Weblate under a different user, or change this setting.

Vezi și:

Accessing repositories

Template loading

It is recommended to use a cached template loader for Django. It caches parsed templates and avoids the need to do parsing with every single request. You can configure it using the following snippet (the `loaders` setting is important here):

```
TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [
            os.path.join(BASE_DIR, "templates"),
        ],
        "OPTIONS": {
            "context_processors": [
                "django.contrib.auth.context_processors.auth",
                "django.template.context_processors.debug",
                "django.template.context_processors.i18n",
                "django.template.context_processors.request",
                "django.template.context_processors.csrf",
                "django.contrib.messages.context_processors.messages",
                "weblate.trans.context_processors.weblate_context",
            ],
            "loaders": [
```

(continues on next page)

(continuare din pagina precedentă)

```
                (
                    "django.template.loaders.cached.Loader",
                    [
                        "django.template.loaders.filesystem.Loader",
                        "django.template.loaders.app_directories.Loader",
                    ],
                ),
            ],
        },
    },
]
```

Vezi și:

`django.template.loaders.cached.Loader`

Running maintenance tasks

For optimal performance, it is good idea to run some maintenance tasks in the background. This is now automatically done by *Background tasks using Celery* and covers following tasks:

- Configuration health check (hourly).
- Committing pending changes (hourly), see *Lazy commits* and `commit_pending`.
- Updating component alerts (daily).
- Update remote branches (nightly), see `AUTO_UPDATE`.
- Translation memory backup to JSON (daily), see `dump_memory`.
- Fulltext and database maintenance tasks (daily and weekly tasks), see `cleanuptrans`.

Schimbat în versiunea 3.2: Since version 3.2, the default way of executing these tasks is using Celery and Weblate already comes with proper configuration, see *Background tasks using Celery*.

System locales and encoding

The system locales should be configured to UTF-8 capable ones. On most Linux distributions this is the default setting. In case it is not the case on your system, please change locales to UTF-8 variant.

For example by editing `/etc/default/locale` and setting there `LANG="C.UTF-8"`.

In some cases the individual services have separate configuration for locales. This varies between distribution and web servers, so check documentation of your web server packages for that.

Apache on Ubuntu uses `/etc/apache2/envvars`:

```
export LANG='en_US.UTF-8'
export LC_ALL='en_US.UTF-8'
```

Apache on CentOS uses `/etc/sysconfig/httpd` (or `/opt/rh/httpd24/root/etc/sysconfig/httpd`):

```
LANG='en_US.UTF-8'
```

Using custom certificate authority

Weblate does verify SSL certificates during HTTP requests. In case you are using custom certificate authority which is not trusted in default bundles, you will have to add its certificate as trusted.

The preferred approach is to do this at system level, please check your distro documentation for more details (for example on debian this can be done by placing the CA certificate into `/usr/local/share/ca-certificates/` and running **`update-ca-certificates`**).

Once this is done, system tools will trust the certificate and this includes Git.

For Python code, you will need to configure requests to use system CA bundle instead of the one shipped with it. This can be achieved by placing following snippet to `settings.py` (the path is Debian specific):

```
import os

os.environ["REQUESTS_CA_BUNDLE"] = "/etc/ssl/certs/ca-certificates.crt"
```

Compressing client assets

Weblate comes with a bunch of JavaScript and CSS files. For performance reasons it is good to compress them before sending to a client. In default configuration this is done on the fly at cost of little overhead. On big installations, it is recommended to enable offline compression mode. This needs to be done in the configuration and the compression has to be triggered on every Weblate upgrade.

The configuration switch is simple by enabling `django.conf.settings.COMPRESS_OFFLINE` and configuring `django.conf.settings.COMPRESS_OFFLINE_CONTEXT` (the latter is already included in the example configuration):

```
COMPRESS_OFFLINE = True
```

On each deploy you need to compress the files to match current version:

```
weblate compress
```

Sugestie: The official Docker image has this feature already enabled.

Vezi şi:

Common Deployment Scenarios, *Serving static files*

2.1.10 Running server

Sugestie: In case you are not experienced with services described below, you might want to try *Installing using Docker*.

You will need several services to run Weblate, the recommended setup consists of:

- Database server (see *Database setup for Weblate*)
- Cache server (see *Enable caching*)
- Frontend web server for static files and SSL termination (see *Serving static files*)
- WSGI server for dynamic content (see *Sample configuration for NGINX and uWSGI*)
- Celery for executing background tasks (see *Background tasks using Celery*)

Notă: There are some dependencies between the services, for example cache and database should be running when starting up Celery or uwsgi processes.

In most cases, you will run all services on single (virtual) server, but in case your installation is heavy loaded, you can split up the services. The only limitation on this is that Celery and Wsgi servers need access to `DATA_DIR`.

Notă: The WSGI process has to be executed under the same user the Celery process, otherwise files in the `DATA_DIR` will be stored with mixed ownership, leading to runtime issues.

See also *Filesystem permissions* and *Background tasks using Celery*.

Running web server

Running Weblate is not different from running any other Django based program. Django is usually executed as uWSGI or fcgi (see examples for different webservers below).

For testing purposes, you can use the built-in web server in Django:

```
weblate runserver
```

Atenționare: DO NOT USE THIS SERVER IN A PRODUCTION SETTING. It has not gone through security audits or performance tests. See also Django documentation on `runserver`.

Sugestie: The Django built-in server serves static files only with `DEBUG` enabled as it is intended for development only. For production use, please see wsgi setups in *Sample configuration for NGINX and uWSGI*, *Sample configuration for Apache*, *Sample configuration for Apache and Gunicorn*, and *Serving static files*.

Serving static files

Schimbat în versiunea 2.4: Prior to version 2.4, Weblate didn't properly use the Django static files framework and the setup was more complex.

Django needs to collect its static files in a single directory. To do so, execute `weblate collectstatic --noinput`. This will copy the static files into a directory specified by the `STATIC_ROOT` setting (this defaults to a `static` directory inside `DATA_DIR`).

It is recommended to serve static files directly from your web server, you should use that for the following paths:

`/static/` Serves static files for Weblate and the admin interface (from defined by `STATIC_ROOT`).

`/media/` Used for user media uploads (e.g. screenshots).

`/favicon.ico` Should be rewritten to rewrite a rule to serve `/static/favicon.ico`.

Vezi și:

Sample configuration for NGINX and uWSGI, *Sample configuration for Apache*, *Sample configuration for Apache and Gunicorn*, *Compressing client assets*, Deploying Django, Deploying static files

Content security policy

The default Weblate configuration enables `weblate.middleware.SecurityMiddleware` middleware which sets security related HTTP headers like `Content-Security-Policy` or `X-XSS-Protection`. These are by default set up to work with Weblate and its configuration, but this might need customization for your environment.

Vezi și:

CSP_SCRIPT_SRC, *CSP_IMG_SRC*, *CSP_CONNECT_SRC*, *CSP_STYLE_SRC*, *CSP_FONT_SRC*

Sample configuration for NGINX and uWSGI

To run production webserver, use the wsgi wrapper installed with Weblate (in virtual env case it is installed as `~/weblate-env/lib/python3.7/site-packages/weblate/wsgi.py`). Don't forget to set the Python search path to your virtualenv as well (for example using `virtualenv = /home/user/weblate-env` in uWSGI).

The following configuration runs Weblate as uWSGI under the NGINX webserver.

Configuration for NGINX (also available as `weblate/examples/weblate.nginx.conf`):

```
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
server {
    listen 80;
    server_name weblate;
    # Not used
    root /var/www/html;

    location ~ ^/favicon.ico$ {
        # DATA_DIR/static/favicon.ico
        alias /home/weblate/data/static/favicon.ico;
        expires 30d;
    }

    location /static/ {
        # DATA_DIR/static/
        alias /home/weblate/data/static/;
        expires 30d;
    }

    location /media/ {
        # DATA_DIR/media/
        alias /home/weblate/data/media/;
        expires 30d;
    }

    location / {
        include uwsgi_params;
        # Needed for long running operations in admin interface
        uwsgi_read_timeout 3600;
        # Adjust based to uwsgi configuration:
        uwsgi_pass unix:///run/uwsgi/app/weblate/socket;
        # uwsgi_pass 127.0.0.1:8080;
    }
}
```

Configuration for uWSGI (also available as `weblate/examples/weblate.uwsgi.ini`):

```
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
[uwsgi]
plugins       = python3
master        = true
protocol      = uwsgi
socket        = 127.0.0.1:8080
wsgi-file     = /home/weblate/weblate-env/lib/python3.9/site-packages/weblate/wsgi.
↪py

# Add path to Weblate checkout if you did not install
# Weblate by pip
# python-path   = /path/to/weblate

# In case you're using virtualenv uncomment this:
virtualenv = /home/weblate/weblate-env

# Needed for OAuth/OpenID
buffer-size   = 8192

# Reload when consuming too much of memory
reload-on-rss = 250

# Increase number of workers for heavily loaded sites
workers       = 8

# Enable threads for Sentry error submission
enable-threads = true

# Child processes do not need file descriptors
close-on-exec = true

# Avoid default 0000 umask
umask = 0022

# Run as weblate user
uid = weblate
gid = weblate

# Enable harakiri mode (kill requests after some time)
# harakiri = 3600
# harakiri-verbose = true

# Enable uWSGI stats server
# stats = :1717
# stats-http = true

# Do not log some errors caused by client disconnects
ignore-sigpipe = true
ignore-write-errors = true
disable-write-exception = true
```

Vezi și:

How to use Django with uWSGI

Sample configuration for Apache

It is recommended to use prefork MPM when using WSGI with Weblate.

The following configuration runs Weblate as WSGI, you need to have enabled `mod_wsgi` (available as `weblate/examples/apache.conf`):

```
#
# VirtualHost for Weblate
#
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
#
<VirtualHost *:80>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>

    # DATA_DIR/media/
    Alias /media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    # Path to your Weblate virtualenv
    WSGIDaemonProcess weblate python-home=/home/weblate/weblate-env user=weblate
    WSGIProcessGroup weblate
    WSGIApplicationGroup %{GLOBAL}

    WSGIScriptAlias / /home/weblate/weblate-env/lib/python3.7/site-packages/
↪weblate/wsgi.py process-group=weblate request-timeout=600
    WSGIPassAuthorization On

    <Directory /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/>
        <Files wsgi.py>
        Require all granted
        </Files>
    </Directory>

</VirtualHost>
```

Notă: Weblate requires Python 3, so please make sure you are running Python 3 variant of the modwsgi. Usually it is available as a separate package, for example `libapache2-mod-wsgi-py3`.

Vezi și:

System locales and encoding, How to use Django with Apache and mod_wsgi

Sample configuration for Apache and Gunicorn

The following configuration runs Weblate in Gunicorn and Apache 2.4 (available as `weblate/examples/apache.gunicorn.conf`):

```
#
# VirtualHost for Weblate using gunicorn on localhost:8000
#
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
#
<VirtualHost *:443>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>

    # DATA_DIR/media/
    Alias /media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/https_cert.cert
    SSLCertificateKeyFile /etc/apache2/ssl/https_key.pem
    SSLProxyEngine On

    ProxyPass /favicon.ico !
    ProxyPass /static/ !
    ProxyPass /media/ !

    ProxyPass / http://localhost:8000/
    ProxyPassReverse / http://localhost:8000/
    ProxyPreserveHost On
</VirtualHost>
```

Vezi și:

How to use Django with Gunicorn

Running Weblate under path

Nou în versiunea 1.3.

It is recommended to use prefork MPM when using WSGI with Weblate.

A sample Apache configuration to serve Weblate under `/weblate`. Again using `mod_wsgi` (also available as `weblate/examples/apache-path.conf`):

```
#
# VirtualHost for Weblate, running under /weblate path
#
```

(continues on next page)

(continuare din pagina precedentă)

```
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
#
<VirtualHost *:80>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /weblate/favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /weblate/static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>

    # DATA_DIR/media/
    Alias /weblate/media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    # Path to your Weblate virtualenv
    WSGIDaemonProcess weblate python-home=/home/weblate/weblate-env user=weblate
    WSGIProcessGroup weblate
    WSGIApplicationGroup %{GLOBAL}

    WSGIScriptAlias /weblate /home/weblate/weblate-env/lib/python3.7/site-packages/
↪weblate/wsgi.py process-group=weblate request-timeout=600
    WSGIPassAuthorization On

    <Directory /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/>
        <Files wsgi.py>
        Require all granted
        </Files>
    </Directory>

</VirtualHost>
```

Additionally, you will have to adjust `weblate/settings.py`:

```
URL_PREFIX = "/weblate"
```

2.1.11 Background tasks using Celery

Nou în versiunea 3.2.

Weblate uses Celery to process background tasks. A typical setup using Redis as a backend looks like this:

```
CELERY_TASK_ALWAYS_EAGER = False
CELERY_BROKER_URL = "redis://localhost:6379"
CELERY_RESULT_BACKEND = CELERY_BROKER_URL
```

Vezi și:

Redis broker configuration in Celery

For development, you might want to use eager configuration, which does process all tasks in place, but this will have performance impact on Weblate:

```
CELERY_TASK_ALWAYS_EAGER = True
CELERY_BROKER_URL = "memory://"
CELERY_TASK_EAGER_PROPAGATES = True
```

You should also start the Celery worker to process the tasks and start scheduled tasks, this can be done directly on the command line (which is mostly useful when debugging or developing):

```
./weblate/examples/celery start
./weblate/examples/celery stop
```

Notă: The Celery process has to be executed under the same user as the WSGI process, otherwise files in the `DATA_DIR` will be stored with mixed ownership, leading to runtime issues.

See also *Filesystem permissions* and *Running server*.

Running Celery as system service

Most likely you will want to run Celery as a daemon and that is covered by Daemonization. For the most common Linux setup using systemd, you can use the example files shipped in the `examples` folder listed below.

Systemd unit to be placed as `/etc/systemd/system/celery-weblate.service`:

```
[Unit]
Description=Celery Service (Weblate)
After=network.target

[Service]
Type=forking
User=weblate
Group=weblate
EnvironmentFile=/etc/default/celery-weblate
WorkingDirectory=/home/weblate
RuntimeDirectory=celery
RuntimeDirectoryPreserve=restart
LogsDirectory=celery
ExecStart=/bin/sh -c '${CELERY_BIN} multi start ${CELERYD_NODES} \
  -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \
  --logfile=${CELERYD_LOG_FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_OPTS}'
ExecStop=/bin/sh -c '${CELERY_BIN} multi stopwait ${CELERYD_NODES} \
  --pidfile=${CELERYD_PID_FILE}'
ExecReload=/bin/sh -c '${CELERY_BIN} multi restart ${CELERYD_NODES} \
  -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \
  --logfile=${CELERYD_LOG_FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_OPTS}'

[Install]
WantedBy=multi-user.target
```

Environment configuration to be placed as `/etc/default/celery-weblate`:

```
# Name of nodes to start
CELERYD_NODES="celery notify memory backup translate"

# Absolute or relative path to the 'celery' command:
CELERY_BIN="/home/weblate/weblate-env/bin/celery"

# App instance to use
# comment out this line if you don't use an app
CELERY_APP="weblate.utils"
```

(continues on next page)

(continuare din pagina precedentă)

```
# Extra command-line arguments to the worker,
# increase concurency if you get weblate.E019
CELERYD_OPTS="--beat:celery --queues:celery=celery --prefetch-multiplier:celery=4 \
    --queues:notify=notify --prefetch-multiplier:notify=10 \
    --queues:memory=memory --prefetch-multiplier:memory=10 \
    --queues:translate=translate --prefetch-multiplier:translate=4 \
    --concurrency:backup=1 --queues:backup=backup  --prefetch-multiplier:backup=2"

# Logging configuration
# - %n will be replaced with the first part of the nodename.
# - %I will be replaced with the current child process index
#   and is important when using the prefork pool to avoid race conditions.
CELERYD_PID_FILE="/run/celery/weblate-%n.pid"
CELERYD_LOG_FILE="/var/log/celery/weblate-%n%I.log"
CELERYD_LOG_LEVEL="INFO"

# Internal Weblate variable to indicate we're running inside Celery
CELERY_WORKER_RUNNING="1"
```

Additional configuration to rotate Celery logs using **`logrotate`** to be placed as `/etc/logrotate.d/celery`:

```
/var/log/celery/*.log {
        weekly
        missingok
        rotate 12
        compress
        notifempty
}
```

Periodic tasks using Celery beat

Weblate comes with built-in setup for scheduled tasks. You can however define additional tasks in `settings.py`, for example see *Lazy commits*.

The tasks are supposed to be executed by Celery beats daemon. In case it is not working properly, it might not be running or its database was corrupted. Check the Celery startup logs in such case to figure out root cause.

Monitoring Celery status

You can use *`celery_queues`* to see current length of Celery task queues. In case the queue will get too long, you will also get configuration error in the admin interface.

> **Atenționare:** The Celery errors are by default only logged into Celery log and are not visible to user. In case you want to have overview on such failures, it is recommended to configure *Collecting error reports*.

Vezi și:

Configuration and defaults, Workers Guide, Daemonization, Monitoring and Management Guide, *`celery_queues`*

2.1.12 Monitoring Weblate

Weblate provides the `/healthz/` URL to be used in simple health checks, for example using Kubernetes. The Docker container has built-in health check using this URL.

For monitoring metrics of Weblate you can use *GET /api/metrics/* API endpoint.

Vezi și:

Weblate plugin for Munin

2.1.13 Collecting error reports

Weblate, as any other software, can fail. In order to collect useful failure states we recommend to use third party services to collect such information. This is especially useful in case of failing Celery tasks, which would otherwise only report error to the logs and you won't get notified on them. Weblate has support for the following services:

Sentry

Weblate has built-in support for Sentry. To use it, it's enough to set *SENTRY_DSN* in the `settings.py`:

```
SENTRY_DSN = "https://id@your.sentry.example.com/"
```

Rollbar

Weblate has built-in support for Rollbar. To use it, it's enough to follow instructions for Rollbar notifier for Python.

In short, you need to adjust `settings.py`:

```
# Add rollbar as last middleware:
MIDDLEWARE = [
    # … other middleware classes …
    "rollbar.contrib.django.middleware.RollbarNotifierMiddleware",
]

# Configure client access
ROLLBAR = {
    "access_token": "POST_SERVER_ITEM_ACCESS_TOKEN",
    "client_token": "POST_CLIENT_ITEM_ACCESS_TOKEN",
    "environment": "development" if DEBUG else "production",
    "branch": "main",
    "root": "/absolute/path/to/code/root",
}
```

Everything else is integrated automatically, you will now collect both server and client side errors.

2.1.14 Migrating Weblate to another server

Migrating Weblate to another server should be pretty easy, however it stores data in few locations which you should migrate carefully. The best approach is to stop Weblate for the migration.

Migrating database

Depending on your database backend, you might have several options to migrate the database. The most straightforward one is to dump the database on one server and import it on the new one. Alternatively you can use replication in case your database supports it.

The best approach is to use database native tools, as they are usually the most effective (e.g. **mysqldump** or **pg_dump**). If you want to migrate between different databases, the only option might be to use Django management to dump and import the database:

```
# Export current data
weblate dumpdata > /tmp/weblate.dump
# Import dump
weblate loaddata /tmp/weblate.dump
```

Migrating VCS repositories

The VCS repositories stored under *DATA_DIR* need to be migrated as well. You can simply copy them or use **rsync** to do the migration more effectively.

Other notes

Don't forget to move other services Weblate might have been using like Redis, Cron jobs or custom authentication backends.

2.2 Weblate deployments

Weblate can be easily installed in your cloud. Please find detailed guide for your platform:

- *Installing using Docker*
- *Installing on OpenShift*
- *Installing on Kubernetes*

2.2.1 Third-party deployments for Weblate

Notă: Following deployments are not developed or supported by Weblate team. Parts of the setup might vary from what is described in this documentation.

Bitnami Weblate stack

Bitnami provides a Weblate stack for many platforms at <https://bitnami.com/stack/weblate>. The setup will be adjusted during installation, see <https://bitnami.com/stack/weblate/README.txt> for more documentation.

Weblate Cloudron Package

Cloudron is a platform for self-hosting web applications. Weblate installed with Cloudron will be automatically kept up-to-date. The package is maintained by the Cloudron team at their Weblate package repo.

Cloudron install

Weblate in YunoHost

The self-hosting project YunoHost provides a package for Weblate. Once you have your YunoHost installation, you may install Weblate as any other application. It will provide you with a fully working stack with backup and restoration, but you may still have to edit your settings file for specific usages.

You may use your administration interface, or this button (it will bring you to your server):

Install with YUNOHOST

It also is possible to use the commandline interface:

```
yunohost app install https://github.com/YunoHost-Apps/weblate_ynh
```

2.3 Upgrading Weblate

2.3.1 Docker image upgrades

The official Docker image (see *Installing using Docker*) has all upgrade steps integrated. There are no manual step besides pulling latest version.

2.3.2 Generic upgrade instructions

Before upgrading, please check the current *Software requirements* as they might have changed. Once all requirements are installed or updated, please adjust your `settings.py` to match changes in the configuration (consult `settings_example.py` for correct values).

Always check *Version specific instructions* before upgrade. In case you are skipping some versions, please follow instructions for all versions you are skipping in the upgrade. Sometimes it's better to upgrade to some intermediate version to ensure a smooth migration. Upgrading across multiple releases should work, but is not as well tested as single version upgrades.

Notă: It is recommended to perform a full database backup prior to upgrade so that you can roll back the database in case upgrade fails, see *Copierea de rezervă și mutarea Weblate*.

1. Stop wsgi and Celery processes. The upgrade can perform incompatible changes in the database, so it is always safer to avoid old processes running while upgrading.
2. Upgrade Weblate code.

   For pip installs it can be achieved by:

   ```
   pip install -U Weblate
   ```

   With Git checkout you need to fetch new source code and update your installation:

```
cd weblate-src
git pull
# Update Weblate inside your virtualenv
. ~/weblate-env/bin/pip install -e .
# Install dependencies directly when not using virtualenv
pip install --upgrade -r requirements.txt
```

3. Upgrade configuration file, refer to `settings_example.py` or *Version specific instructions* for needed steps.
4. Upgrade database structure:

```
weblate migrate --noinput
```

5. Collect updated static files (see *Running server* and *Serving static files*):

```
weblate collectstatic --noinput
```

6. Compress JavaScript and CSS files (optional, see *Compressing client assets*):

```
weblate compress
```

7. If you are running version from Git, you should also regenerate locale files every time you are upgrading. You can do this by invoking:

```
weblate compilemessages
```

8. Verify that your setup is sane (see also *Production setup*):

```
weblate check --deploy
```

9. Restart Celery worker (see *Background tasks using Celery*).

2.3.3 Version specific instructions

Upgrade from 2.x

If you are upgrading from 2.x release, always first upgrade to 3.0.1 and then continue upgrading in the 3.x series. Upgrades skipping this step are not supported and will break.

Vezi și:

Upgrade from 2.20 to 3.0 in Weblate 3.0 documentation

Upgrade from 3.x

If you are upgrading from 3.x release, always first upgrade to 4.0.4 or 4.1.1 and then continue upgrading in the 4.x series. Upgrades skipping this step are not supported and will break.

Vezi și:

Upgrade from 3.11 to 4.0 in Weblate 4.0 documentation

Upgrade from 4.0 to 4.1

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- There are several changes in `settings_example.py`, most notable middleware changes, please adjust your settings accordingly.
- There are new file formats, you might want to include them in case you modified the *WEBLATE_FORMATS*.
- There are new quality checks, you might want to include them in case you modified the *CHECK_LIST*.
- There is change in `DEFAULT_THROTTLE_CLASSES` setting to allow reporting of rate limiting in the API.
- There are some new and updated requirements.
- There is a change in `INSTALLED_APPS`.
- The `MT_DEEPL_API_VERSION` setting has been removed in Version 4.7. The *DeepL* machine translation now uses the new *MT_DEEPL_API_URL* instead. You might need to adjust *MT_DEEPL_API_URL* to match your subscription.

Vezi și:

Generic upgrade instructions

Upgrade from 4.1 to 4.2

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- Upgrade from 3.x releases is not longer supported, please upgrade to 4.0 or 4.1 first.
- There are some new and updated requirements.
- There are several changes in `settings_example.py`, most notable new middleware and changed application ordering.
- The keys for JSON based formats no longer include leading dot. The strings are adjusted during the database migration, but external components might need adjustment in case you rely on keys in exports or API.
- The Celery configuration was changed to no longer use `memory` queue. Please adjust your startup scripts and `CELERY_TASK_ROUTES` setting.
- The Weblate domain is now configured in the settings, see *SITE_DOMAIN* (or *WEBLATE_SITE_DOMAIN*). You will have to configure it before running Weblate.
- The username and email fields on user database now should be case insensitive unique. It was mistakenly not enforced with PostgreSQL.

Vezi și:

Generic upgrade instructions

Upgrade from 4.2 to 4.3

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- There are some changes in quality checks, you might want to include them in case you modified the `CHECK_LIST`.
- The source language attribute was moved from project to a component what is exposed in the API. You will need to update *Weblate Client* in case you are using it.
- The database migration to 4.3 might take long depending on number of strings you are translating (expect around one hour of migration time per 100,000 source strings).
- There is a change in `INSTALLED_APPS`.
- There is a new setting `SESSION_COOKIE_AGE_AUTHENTICATED` which complements `SESSION_COOKIE_AGE`.
- In case you were using **`hub`** or **`lab`** to integrate with GitHub or GitLab, you will need to reconfigure this, see `GITHUB_CREDENTIALS` and `GITLAB_CREDENTIALS`.

Schimbat în versiunea 4.3.1:

- The Celery configuration was changed to add `memory` queue. Please adjust your startup scripts and `CELERY_TASK_ROUTES` setting.

Schimbat în versiunea 4.3.2:

- The `post_update` method of addons now takes extra `skip_push` parameter.

Vezi și:

Generic upgrade instructions

Upgrade from 4.3 to 4.4

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- There is a change in `INSTALLED_APPS`, `weblate.configuration` has to be added there.
- Django 3.1 is now required.
- In case you are using MySQL or MariaDB, the minimal required versions have increased, see *MySQL and MariaDB*.

Schimbat în versiunea 4.4.1:

- *Monolingual gettext* now uses both `msgid` and `msgctxt` when present. This will change identification of translation strings in such files breaking links to Weblate extended data such as screenshots or review states. Please make sure you commit pending changes in such files prior upgrading and it is recommeded to force loading of affected component using `loadpo`.
- Increased minimal required version of translate-toolkit to address several file format issues.

Vezi și:

Generic upgrade instructions

Upgrade from 4.4 to 4.5

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- The migration might take considerable time if you had big glossaries.
- Glossaries are now stored as regular components.
- The glossary API is removed, use regular translation API to access glossaries.
- There is a change in `INSTALLED_APPS` - `weblate.metrics` should be added.

Schimbat în versiunea 4.5.1:

- There is a new dependency on the *pyahocorasick* module.

Vezi și:

Generic upgrade instructions

Upgrade from 4.5 to 4.6

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- There are new file formats, you might want to include them in case you modified the *`WEBLATE_FORMATS`*.
- API for creating components now automatically uses *Weblate internal URLs*, see *`POST /api/projects/(string:project)/components/`*.
- There is a change in dependencies and `PASSWORD_HASHERS` to prefer Argon2 for passwords hashing.

Vezi și:

Generic upgrade instructions

Upgrade from 4.6 to 4.7

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- There are several changes in `settings_example.py`, most notable middleware changes (`MIDDLEWARE`), please adjust your settings accordingly.
- The *DeepL* machine translation now has a generic *`MT_DEEPL_API_URL`* setting to adapt to different subscription models more flexibly. The `MT_DEEPL_API_VERSION` setting is no longer used.
- Django 3.2 is now required.

Vezi și:

Generic upgrade instructions

Upgrade from 4.7 to 4.8

Please follow *Generic upgrade instructions* in order to perform update.

There are no additional upgrade steps needed in this release.

Vezi și:

Generic upgrade instructions

2.3.4 Upgrading from Python 2 to Python 3

Weblate no longer supports Python older than 3.5. In case you are still running on older version, please perform migration to Python 3 first on existing version and upgrade later. See Upgrading from Python 2 to Python 3 in the Weblate 3.11.1 documentation.

2.3.5 Migrating from other databases to PostgreSQL

If you are running Weblate on other dabatase than PostgreSQL, you should consider migrating to PostgreSQL as Weblate performs best with it. The following steps will guide you in migrating your data between the databases. Please remember to stop both web and Celery servers prior to the migration, otherwise you might end up with inconsistent data.

Creating a database in PostgreSQL

It is usually a good idea to run Weblate in a separate database, and separate user account:

```
# If PostgreSQL was not installed before, set the main password
sudo -u postgres psql postgres -c "\password postgres"

# Create a database user called "weblate"
sudo -u postgres createuser -D -P weblate

# Create the database "weblate" owned by "weblate"
sudo -u postgres createdb -E UTF8 -O weblate weblate
```

Migrating using Django JSON dumps

The simplest approach for migration is to utilize Django JSON dumps. This works well for smaller installations. On bigger sites you might want to use pgloader instead, see *Migrating to PostgreSQL using pgloader*.

1. Add PostgreSQL as additional database connection to the `settings.py`:

```
DATABASES = {
    "default": {
        # Database engine
        "ENGINE": "django.db.backends.mysql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "database.example.com",
        # Set to empty string for default
        "PORT": "",
        # Additional database options
```

(continues on next page)

(continuare din pagina precedentă)

```
        "OPTIONS": {
            # In case of using an older MySQL server, which has MyISAM as a␣
↪default storage
            # 'init_command': 'SET storage_engine=INNODB',
            # Uncomment for MySQL older than 5.7:
            # 'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
            # If your server supports it, see the Unicode issues above
            "charset": "utf8mb4",
            # Change connection timeout in case you get MySQL gone away error:
            "connect_timeout": 28800,
        },
    },
    "postgresql": {
        # Database engine
        "ENGINE": "django.db.backends.postgresql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "database.example.com",
        # Set to empty string for default
        "PORT": "",
    },
}
```

2. Run migrations and drop any data inserted into the tables:

```
weblate migrate --database=postgresql
weblate sqlflush --database=postgresql | weblate dbshell --database=postgresql
```

3. Dump legacy database and import to PostgreSQL

```
weblate dumpdata --all --output weblate.json
weblate loaddata weblate.json --database=postgresql
```

4. Adjust `DATABASES` to use just PostgreSQL database as default, remove legacy connection.

Weblate should be now ready to run from the PostgreSQL database.

Migrating to PostgreSQL using pgloader

The pgloader is a generic migration tool to migrate data to PostgreSQL. You can use it to migrate Weblate database.

1. Adjust your `settings.py` to use PostgreSQL as a database.
2. Migrate the schema in the PostgreSQL database:

```
weblate migrate
weblate sqlflush | weblate dbshell
```

3. Run the pgloader to transfer the data. The following script can be used to migrate the database, but you might want to learn more about pgloader to understand what it does and tweak it to match your setup:

```
LOAD DATABASE
     FROM      mysql://weblate:password@localhost/weblate
     INTO postgresql://weblate:password@localhost/weblate

WITH include no drop, truncate, create no tables, create no indexes, no␣
↪foreign keys, disable triggers, reset sequences, data only
```

(continues on next page)

(continuare din pagina precedentă)

```
ALTER SCHEMA 'weblate' RENAME TO 'public'
;
```

2.3.6 Migrating from Pootle

As Weblate was originally written as replacement from Pootle, it is supported to migrate user accounts from Pootle. You can dump the users from Pootle and import them using `importusers`.

2.4 Copierea de rezervă și mutarea Weblate

2.4.1 Backup automatizat folosind BorgBackup

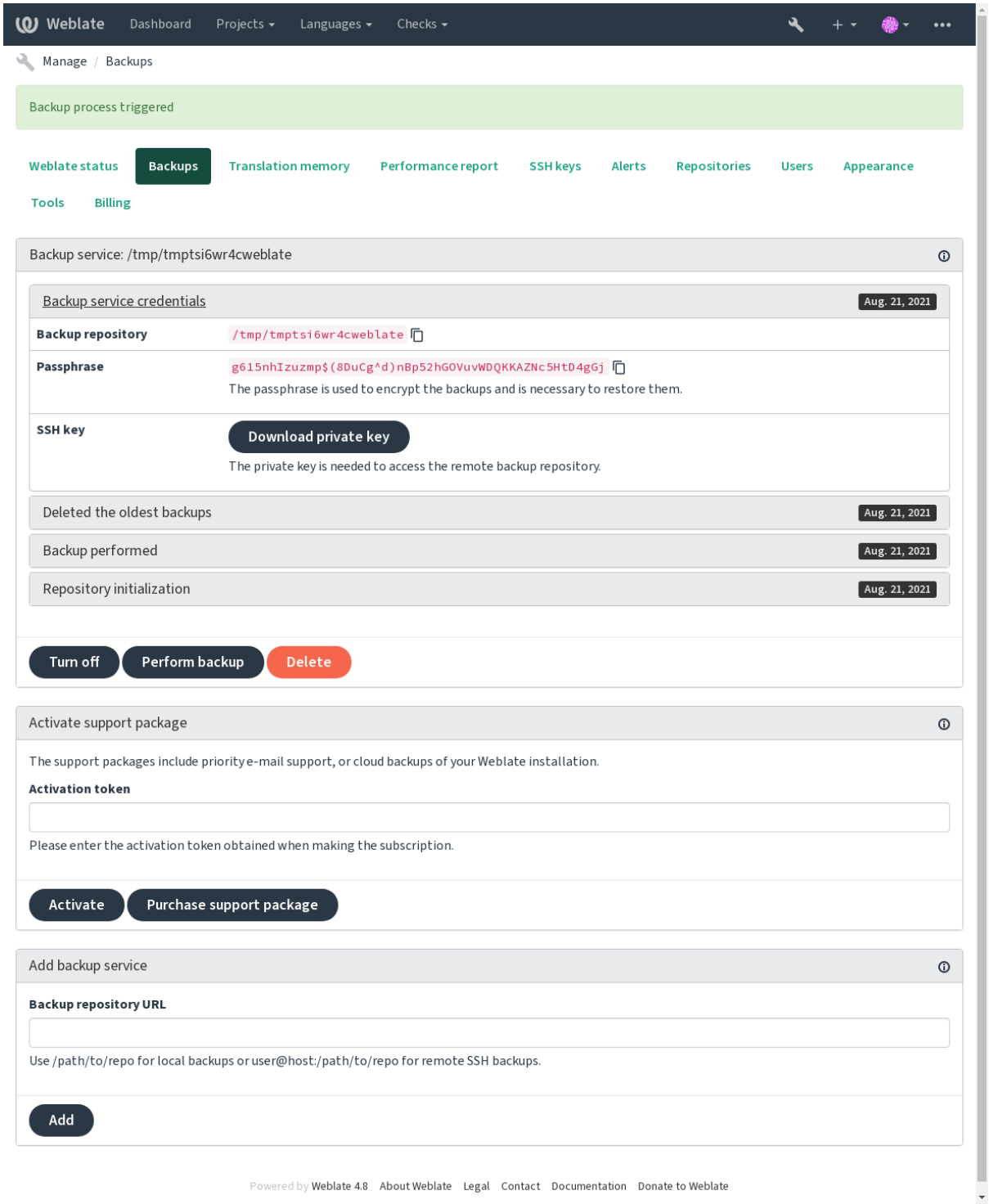
Nou în versiunea 3.9.

Weblate are un suport încorporat pentru crearea de copii de rezervă pentru servicii folosind BorgBackup. Borg creează copii de rezervă criptate, eficiente din punct de vedere al spațiului, care pot fi stocate în siguranță în cloud. Copiile de rezervă pot fi controlate în interfața de gestionare din fila *Backups*.

Schimbat în versiunea 4.4.1: Atât bazele de date PostgreSQL, cât și MySQL/MariaDB sunt incluse în copiile de rezervă automate.

Copiile de rezervă care utilizează Borg sunt incrementale, iar Weblate este configurat să păstreze următoarele copii de rezervă:

- Backup-uri zilnice pentru 14 zile înapoi
- Backup-uri săptămânale pentru 8 săptămâni în urmă
- Backup-uri lunare pentru 6 luni în urmă


Weblate
Dashboard
Projects
Languages
Checks
Manage / Backups
Backup process triggered
Weblate status
Backups
Translation memory
Performance report
SSH keys
Alerts
Repositories
Users
Appearance
Tools
Billing
Backup service: /tmp/tmptsi6wr4cweblate
Backup service credentials
Aug. 21, 2021
Backup repository
/tmp/tmptsi6wr4cweblate
Passphrase
g615nhIzuzmp$(8DuCg^d)nBp52hGOVuvWDQKKAZNc5HtD4gGj
The passphrase is used to encrypt the backups and is necessary to restore them.
SSH key
Download private key
The private key is needed to access the remote backup repository.
Deleted the oldest backups
Aug. 21, 2021
Backup performed
Aug. 21, 2021
Repository initialization
Aug. 21, 2021
Turn off
Perform backup
Delete
Activate support package
The support packages include priority e-mail support, or cloud backups of your Weblate installation.
Activation token
Please enter the activation token obtained when making the subscription.
Activate
Purchase support package
Add backup service
Backup repository URL
Use /path/to/repo for local backups or user@host:/path/to/repo for remote SSH backups.
Add
Powered by Weblate 4.8
About Weblate
Legal
Contact
Documentation
Donate to Weblate

Cheia de criptare Borg

BorgBackup creează copii de rezervă criptate și nu le veți putea restaura fără fraza de acces. Fraza de acces este generată la adăugarea unui nou serviciu de backup și ar trebui să o copiați și să o păstrați într-un loc sigur.

Dacă folosiți *Stocarea de backup provizionată Weblate*, vă rugăm să faceți o copie de rezervă și pentru cheia SSH privată, deoarece aceasta este folosită pentru a accesa copiile de rezervă.

Vezi și:

borg init

2.4.2 Stocarea de backup provizionată Weblate

Cel mai simplu mod de a face o copie de siguranță a instanței Weblate este achiziționarea serviciului backup la weblate.org. Iată cum îl puneți în funcțiune:

1. Achiziționați serviciul *Backup* pe https://weblate.org/support/#backup.
2. Introduceți cheia obținută în interfața de management, a se vedea *Integrating support*.
3. Weblate se conectează la serviciul cloud și obține informații de acces pentru copiile de rezervă.
4. Activați noua configurație de backup din fila *Backups*.
5. Faceți o copie de rezervă a acreditărilor Borg pentru a putea restaura copiile de rezervă, consultați *Cheia de criptare Borg*.

Sugestie: Pasul manual de a porni totul există pentru siguranța dumneavoastră. Fără consimțământul dvs. nu se trimite niciun fel de date către depozitul de backup obținut prin procesul de înregistrare.

2.4.3 Utilizarea stocării de rezervă personalizate

De asemenea, puteți utiliza propriul spațiu de stocare pentru copiile de rezervă. SSH poate fi folosit pentru a stoca copiile de rezervă în destinația la distanță, serverul țintă trebuie să aibă instalat BorgBackup.

Vezi și:

General în documentația Borg

Sistem de fișiere local

Se recomandă să specificați calea absolută pentru copia de rezervă locală, de exemplu */path/to/backup*. Directorul trebuie să poată fi inscripționat de către utilizatorul care rulează Weblate (a se vedea *Filesystem permissions*). Dacă nu există, Weblate încearcă să îl creeze, dar are nevoie de permisiunile corespunzătoare pentru a face acest lucru.

Sugestie: Atunci când executați Weblate în Docker, asigurați-vă că locația de backup este expusă ca volum din containerul Weblate. În caz contrar, copiile de rezervă vor fi aruncate de Docker la repornirea containerului în care se află.

O opțiune este de a plasa copiile de rezervă într-un volum existent, de exemplu `/app/data/borgbackup`. Acesta este un volum existent în container.

Puteți, de asemenea, să adăugați un nou container pentru copiile de rezervă în fișierul Docker Compose, de exemplu, folosind `/borgbackup`:

```
services:
  weblate:
    volumes:
      - /home/weblate/data:/app/data
      - /home/weblate/borgbackup:/borgbackup
```

Directorul în care vor fi stocate copiile de rezervă trebuie să fie deținut de UID 1000, altfel Weblate nu va putea scrie copiile de rezervă acolo.

Remote backups

For creating remote backups, you will have to install BorgBackup onto another server that's accessible for your Weblate deployment via SSH using the Weblate SSH key:

1. Prepare a server where your backups will be stored.
2. Install the SSH server on it (you will get it by default with most Linux distributions).
3. Install BorgBackup on that server; most Linux distributions have packages available (see Installation).
4. Choose an existing user or create a new user that will be used for backing up.
5. Add Weblate SSH key to the user so that Weblate can SSH to the server without a password (see *Weblate SSH key*).
6. Configure the backup location in Weblate as `user@host:/path/to/backups`.

Sugestie: *Stocarea de backup provizionată Weblate* provides you automated remote backups without any effort.

Vezi și:

Weblate SSH key

2.4.4 Restaurarea din BorgBackup

1. Restaurați accesul la depozitul de copii de rezervă și pregătiți fraza de rezervă.
2. Listează toate copiile de rezervă de pe server folosind `borg list REPOSITORY`.
3. Restaurați copia de rezervă dorită în directorul curent folosind `borg extract REPOSITORY::ARCHIVE`.
4. Restaurați baza de date de la descărcarea SQL plasată în directorul `backup` din directorul de date Weblate (see *Date descărcate pentru copii de rezervă*).
5. Copiați configurația Weblate (`backups/settings.py`, see *Date descărcate pentru copii de rezervă*) în locația corectă, a se vedea *Adjusting configuration*.

   When using Docker container, the settings file is already included in the container and you should restore the original environment variables. The `environment.yml` file might help you with this (see *Date descărcate pentru copii de rezervă*).
6. Copiați întregul director de date restaurat în locația configurată prin `DATA_DIR`.

   When using Docker container place the data into the data volume, see *Docker container volumes*.

   Please make sure the files have correct ownership and permissions, see *Filesystem permissions*.

Sesiunea Borg ar putea arăta în felul următor:

```
$ borg list /tmp/xxx
Enter passphrase for key /tmp/xxx:
2019-09-26T14:56:08                  Thu, 2019-09-26 14:56:08␣
↪[de0e0f13643635d5090e9896bdaceb92a023050749ad3f3350e788f1a65576a5]
$ borg extract /tmp/xxx::2019-09-26T14:56:08
Enter passphrase for key /tmp/xxx:
```

Vezi și:

borg list, borg extract

2.4.5 Backup manual

În funcție de ceea ce doriți să salvați, faceți o copie de rezervă a tipului de date pe care Weblate le stochează în fiecare loc respectiv.

Sugestie: Dacă efectuați copii de rezervă manuale, este posibil să doriți să reduceți la tăcere avertismentul Weblate privind lipsa de copii de rezervă adăugând `weblate.I028` la `SILENCED_SYSTEM_CHECKS` în `settings.py` sau *`WEBLATE_SILENCED_SYSTEM_CHECKS`* pentru Docker.

```
SILENCED_SYSTEM_CHECKS.append("weblate.I028")
```

Baza de date

Locația efectivă de stocare depinde de configurația bazei de date.

Sugestie: Baza de date este cea mai importantă modalitate de stocare. Configurați în mod regulat copii de rezervă ale bazei de date. Fără baza de date, toate traducerile dispar.

Backup nativ al bazei de date

Abordarea recomandată este să salvați o copie a bazei de date utilizând instrumente native pentru baze de date, cum ar fi **`pg_dump`** sau **`mysqldump`**. De obicei, se comportă mai bine decât backup-ul Django și restaurează tabelele complete cu toate datele lor.

Puteți restaura această copie de rezervă într-o versiune Weblate mai nouă, aceasta va efectua toate migrările necesare atunci când este rulată în `migrate`. Vă rugăm să consultați *Upgrading Weblate* pentru informații mai detaliate despre cum să faceți upgrade între versiuni.

Copie de rezervă a bazei de date Django

Alternativ, puteți face o copie de siguranță a bazei de date folosind comanda `dumpdata` de la Django. În acest fel, copia de rezervă este agnostică față de baza de date și poate fi utilizată în cazul în care doriți să schimbați backend-ul bazei de date.

Înainte de restaurarea bazei de date, trebuie să rulați exact aceeași versiune Weblate pe care a fost efectuată copia de rezervă. Acest lucru este necesar deoarece structura bazei de date se schimbă între versiuni și ați putea ajunge să corupeți datele într-un fel sau altul. După instalarea aceleiași versiuni, rulați toate migrările bazei de date folosind `migrate`.

Ulterior, unele intrări vor fi deja create în baza de date și le veți avea și în copia de rezervă a bazei de date. Abordarea recomandată este de a șterge manual astfel de intrări folosind shell-ul de management (a se vedea *Invoking management commands*):

```
weblate shell
>>> from weblate.auth.models import User
>>> User.objects.get(username='anonymous').delete()
```

Fișiere

Dacă aveți suficient spațiu de backup, faceți o copie de rezervă a întregului *DATA_DIR*. Acesta este un pariu sigur, chiar dacă include unele fișiere pe care nu le doriți. Următoarele secțiuni descriu în detaliu ce trebuie să salvați și ce puteți sări peste.

Date descărcate pentru copii de rezervă

Schimbat în versiunea 4.7: The environment dump was added as `environment.yml` to help in restoring in the Docker environments.

Stocat în *DATA_DIR* `/backups`.

Weblate descarcă diverse date aici și puteți include aceste fișiere pentru copii de rezervă mai complete. Fișierele sunt actualizate zilnic (necesită un server Celery beats în funcțiune, consultați *Background tasks using Celery*). În prezent, acestea includ:

- Setări Weblate ca `settings.py` (există, de asemenea, o versiune extinsă în `settings-expanded.py`).
- Copie de rezervă a bazei de date PostgreSQL ca `database.sql`.
- Environment dump as `environment.yml`.

În mod implicit, copiile de rezervă ale bazei de date sunt salvate ca text simplu, dar pot fi comprimate sau pot fi omise în întregime folosind *DATABASE_BACKUP*.

To restore the database backup load it using dabase tools, for example:

```
psql --file=database.sql weblate
```

Depozite de control al versiunilor

Stocat în *DATA_DIR* `/vcs`.

The version control repositories contain a copy of your upstream repositories with Weblate changes. If you have *Împingeți pe comitere* enabled for all your translation components, all Weblate changes are included upstream. No need to back up the repositories on the Weblate side as they can be cloned again from the upstream location(s) with no data loss.

Chei SSH și GPG

Stocate în *DATA_DIR* `/ssh` și *DATA_DIR* `/home`.

Dacă utilizați chei SSH sau GPG generate de Weblate, trebuie să faceți o copie de rezervă a acestor locații. În caz contrar, veți pierde cheile private și va trebui să regenerați altele noi.

Fișiere încărcate de utilizator

Stocat în `DATA_DIR` /media.

Ar trebui să faceți o copie de rezervă a tuturor fișierelor încărcate de utilizator (de exemplu, *Visual context for strings*).

Sarcini de celery

Sarcinile Celery pot conține unele informații, dar de obicei nu este necesară pentru o copie de rezervă. Cel mult veți pierde actualizările care nu au fost încă procesate în memoria de traducere. Se recomandă să se efectueze oricum actualizarea textului integral sau a depozitului la restaurare, astfel încât nu există nicio problemă în a le pierde.

Vezi și:

Background tasks using Celery

Linie de comandă pentru backup manual

Folosind un cron job, puteți configura o comandă Bash care să fie executată zilnic, de exemplu:

```
$ XZ_OPT="-9" tar -Jcf ~/backup/weblate-backup-$(date -u +%Y-%m-%d_%H%M%S).xz␣
↪backups vcs ssh home media fonts secret
```

Șirul de caractere dintre ghilimele după *XZ_OPT* vă permite să alegeți opțiunile xz, de exemplu cantitatea de memorie folosită pentru compresie; vedeți https://linux.die.net/man/1/xz

Puteți ajusta lista de dosare și fișiere în funcție de nevoile dumneavoastră. Pentru a evita salvarea memoriei de traducere (în dosarul de copii de rezervă), puteți utiliza:

```
$ XZ_OPT="-9" tar -Jcf ~/backup/weblate-backup-$(date -u +%Y-%m-%d_%H%M%S).xz␣
↪backups/database.sql backups/settings.py vcs ssh home media fonts secret
```

2.4.6 Restaurarea backup-ului manual

1. Restaurați toate datele pe care le-ați salvat.
2. Actualizați toate depozitele folosind `updategit`.

   ```
   weblate updategit --all
   ```

2.4.7 Mutarea unei instalații Weblate

Mutați instalația pe un alt sistem, urmând instrucțiunile de backup și restaurare de mai sus.

Vezi și:

Upgrading from Python 2 to Python 3, *Migrating from other databases to PostgreSQL*

2.5 Autentificare

2.5.1 Înregistrare utilizator

Configurația implicită pentru Weblate este de a utiliza python-social-auth, un formular pe site-ul web pentru a gestiona înregistrarea noilor utilizatori. După confirmarea e-mail-ului, un nou utilizator poate contribui sau se poate autentifica prin utilizarea unuia dintre serviciile terților.

De asemenea, puteți dezactiva înregistrarea noilor utilizatori folosind `REGISTRATION_OPEN`.

Încercările de autentificare sunt supuse *Rate limiting*.

2.5.2 Backend-uri de autentificare

Pentru autentificare se folosește soluția integrată în Django, care include diverse opțiuni sociale. Folosirea acesteia înseamnă că puteți importa baza de date de utilizatori din alte proiecte bazate pe Django (see *Migrating from Pootle*).

În plus, Django poate fi configurat pentru a se autentifica și prin alte mijloace.

Vezi și:

Authentication settings descrie modul de configurare a autentificării în imaginea oficială Docker.

2.5.3 Autentificare socială

Datorită Welcome to Python Social Auth's documentation!, Weblate suportă autentificarea cu ajutorul multor servicii terțe, cum ar fi GitLab, Ubuntu, Fedora, etc.

Vă rugăm să consultați documentația lor pentru instrucțiuni de configurare generică în Django Framework.

Notă: În mod implicit, Weblate se bazează pe serviciile de autentificare ale unor terțe părți pentru a furniza o adresă de e-mail validată. Dacă unele dintre serviciile pe care doriți să le utilizați nu acceptă acest lucru, vă rugăm să impuneți validarea e-mailului pe partea Weblate prin configurarea FORCE_EMAIL_VALIDATION pentru acestea. De exemplu:

```
SOCIAL_AUTH_OPENSUSE_FORCE_EMAIL_VALIDATION = True
```

Vezi și:

Pipeline

Activarea backend-urilor individuale este destul de ușoară, este doar o chestiune de adăugare a unei intrări la `AUTHENTICATION_BACKENDS` și, eventual, adăugarea cheilor necesare pentru o anumită metodă de autentificare. Vă rugăm să rețineți că unele backend-uri nu furnizează în mod implicit e-mailul utilizatorului, trebuie să îl solicitați în mod explicit, altfel Weblate nu va putea să crediteze în mod corespunzător contribuțiile pe care le fac utilizatorii.

Sugestie: Most of the authentication backends require HTTPS. Once HTTPS is enabled in your web server please configure Weblate to report it properly using `ENABLE_HTTPS`, or by `WEBLATE_ENABLE_HTTPS` in the Docker container.

Vezi și:

Python Social Auth backend

Autentificare OpenID

În cazul serviciilor bazate pe OpenID, de obicei este vorba doar de activarea acestora. Următoarea secțiune permite autentificarea OpenID pentru OpenSUSE, Fedora și Ubuntu:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.email.EmailAuth",
    "social_core.backends.suse.OpenSUSEOpenId",
    "social_core.backends.ubuntu.UbuntuOpenId",
    "social_core.backends.fedora.FedoraOpenId",
    "weblate.accounts.auth.WeblateUserBackend",
)
```

Vezi și:

OpenID

Autentificarea GitHub

Trebuie să înregistrați o aplicație OAuth pe GitHub și apoi să-i spuneți lui Weblate toate secretele sale:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.github.GithubOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Social auth backends setup
SOCIAL_AUTH_GITHUB_KEY = "GitHub Client ID"
SOCIAL_AUTH_GITHUB_SECRET = "GitHub Client Secret"
SOCIAL_AUTH_GITHUB_SCOPE = ["user:email"]
```

GitHub ar trebui să fie configurat pentru a avea un URL de callback ca `https://example.com/accounts/complete/github/`.

Notă: URL-ul de callback furnizat de Weblate în timpul autentificării include domeniul configurat. În cazul în care primiți erori privind neconcordanța URL-ului, poate doriți să remediați acest lucru, consultați *Set correct site domain*.

Vezi și:

GitHub

Autentificarea Bitbucket

Trebuie să înregistrați o aplicație pe Bitbucket și apoi să-i spuneți lui Weblate toate secretele sale:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.bitbucket.BitbucketOAuth",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Social auth backends setup
SOCIAL_AUTH_BITBUCKET_KEY = "Bitbucket Client ID"
SOCIAL_AUTH_BITBUCKET_SECRET = "Bitbucket Client Secret"
SOCIAL_AUTH_BITBUCKET_VERIFIED_EMAILS_ONLY = True
```

Notă: URL-ul de callback furnizat de Weblate în timpul autentificării include domeniul configurat. În cazul în care primiți erori privind neconcordanța URL-ului, poate doriți să remediați acest lucru, consultați *Set correct site domain*.

Vezi și:

Bitbucket

Google OAuth 2

Pentru a utiliza Google OAuth 2, trebuie să înregistrați o aplicație pe <https://console.developers.google.com/> și să activați API-ul Google+.

URL-ul de redirecționare este `https://WEBLATE SERVER/accounts/complete/google-oauth2/`

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.google.GoogleOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Social auth backends setup
SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = "Client ID"
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = "Client secret"
```

Notă: URL-ul de callback furnizat de Weblate în timpul autentificării include domeniul configurat. În cazul în care primiți erori privind neconcordanța URL-ului, poate doriți să remediați acest lucru, consultați *Set correct site domain*.

Vezi și:

Google

Facebook OAuth 2

Ca de obicei în cazul serviciilor OAuth 2, trebuie să vă înregistrați aplicația cu Facebook. După ce ați făcut acest lucru, puteți configura Weblate pentru a o utiliza:

URL-ul de redirecționare este `https://WEBLATE SERVER/accounts/complete/facebook/`

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.facebook.FacebookOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Social auth backends setup
SOCIAL_AUTH_FACEBOOK_KEY = "key"
SOCIAL_AUTH_FACEBOOK_SECRET = "secret"
SOCIAL_AUTH_FACEBOOK_SCOPE = ["email", "public_profile"]
```

Notă: URL-ul de callback furnizat de Weblate în timpul autentificării include domeniul configurat. În cazul în care primiți erori privind neconcordanța URL-ului, poate doriți să remediați acest lucru, consultați *Set correct site domain*.

Vezi și:

Facebook

GitLab OAuth 2

Pentru a utiliza GitLab OAuth 2, trebuie să înregistrați o aplicație pe <https://gitlab.com/profile/applications>.

URL-ul de redirecționare este `https://WEBLATE SERVER/accounts/complete/gitlab/` și asigurați-vă că ați marcat domeniul *read_user*.

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.gitlab.GitLabOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Social auth backends setup
SOCIAL_AUTH_GITLAB_KEY = "Application ID"
SOCIAL_AUTH_GITLAB_SECRET = "Secret"
SOCIAL_AUTH_GITLAB_SCOPE = ["read_user"]

# If you are using your own GitLab
# SOCIAL_AUTH_GITLAB_API_URL = 'https://gitlab.example.com/'
```

Notă: URL-ul de callback furnizat de Weblate în timpul autentificării include domeniul configurat. În cazul în care primiți erori privind neconcordanța URL-ului, poate doriți să remediați acest lucru, consultați *Set correct site domain*.

Vezi și:

GitLab

Microsoft Azure directoare active

Weblate poate fi configurat pentru a utiliza chiriași comuni sau specifici pentru autentificare.

URL-ul de redirecționare este `https://WEBLATE SERVER/accounts/complete/azuread-oauth2/` pentru autentificarea comună și `https://WEBLATE SERVER/accounts/complete/azuread-tenant-oauth2/` pentru autentificarea specifică chiriașilor.

```
# Azure AD common

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.azuread.AzureADOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# OAuth2 keys
SOCIAL_AUTH_AZUREAD_OAUTH2_KEY = ""
SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET = ""
```

```
# Azure AD Tenant

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.azuread_tenant.AzureADTenantOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# OAuth2 keys
```

(continues on next page)

(continuare din pagina precedentă)

```
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY = ""
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECRET = ""
# Tenant ID
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENANT_ID = ""
```

Notă: URL-ul de callback furnizat de Weblate în timpul autentificării include domeniul configurat. În cazul în care primiți erori privind neconcordanța URL-ului, poate doriți să remediați acest lucru, consultați *Set correct site domain*.

Vezi și:

Microsoft Azure Active Directory

Slack

Pentru a utiliza Slack OAuth 2, trebuie să înregistrați o aplicație pe <https://api.slack.com/apps>.

URL-ul de redirecționare este `https://WEBLATE SERVER/accounts/complete/slack/`.

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.slack.SlackOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Social auth backends setup
SOCIAL_AUTH_SLACK_KEY = ""
SOCIAL_AUTH_SLACK_SECRET = ""
```

Notă: URL-ul de callback furnizat de Weblate în timpul autentificării include domeniul configurat. În cazul în care primiți erori privind neconcordanța URL-ului, poate doriți să remediați acest lucru, consultați *Set correct site domain*.

Vezi și:

Slack

Overriding authentication method names and icons

You can override the authentication method display name and icon using using settings as `SOCIAL_AUTH_<NAME>_IMAGE` and `SOCIAL_AUTH_<NAME>_TITLE`. For example overriding naming for Auth0 would look like:

```
SOCIAL_AUTH_AUTH0_IMAGE = "custom.svg"
SOCIAL_AUTH_AUTH0_TITLE = "Custom auth"
```

Dezactivarea autentificării prin parolă

Autentificarea prin e-mail și parolă poate fi dezactivată prin eliminarea `social_core.backends.email.EmailAuth` din `AUTHENTICATION_BACKENDS`. Păstrați întotdeauna `weblate.accounts.auth.WeblateUserBackend` acolo, este necesar pentru funcționalitatea de bază a Weblate.

Sfat: Puteți utiliza în continuare autentificarea prin parolă pentru interfața de administrare, pentru utilizatorii pe care îi creați manual acolo. Trebuie doar să navigați la `/admin/`.

De exemplu, autentificarea folosind doar furnizorul openSUSE Open ID poate fi realizată folosind următoarele:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.suse.OpenSUSEOpenId",
    "weblate.accounts.auth.WeblateUserBackend",
)
```

2.5.4 Autentificarea prin parolă

Fișierul implicit `settings.py` vine cu un set rezonabil de `AUTH_PASSWORD_VALIDATORS`:

- Parolele nu pot fi prea asemănătoare cu alte informații personale.
- Parolele trebuie să conțină cel puțin 10 caractere.
- Parolele nu pot fi o parolă folosită în mod obișnuit.
- Parolele nu pot fi în întregime numerice.
- Parolele nu pot fi formate dintr-un singur caracter sau numai din spații albe.
- Parolele nu se pot potrivi cu o parolă pe care ați folosit-o în trecut.

Puteți personaliza această setare pentru a se potrivi cu politica dumneavoastră privind parolele.

În plus, puteți instala și django-zxcvbn-password care oferă estimări destul de realiste ale dificultății parolelor și permite respingerea parolelor sub un anumit prag.

2.5.5 Autentificare SAML

Nou în versiunea 4.1.1.

Vă rugăm să urmați instrucțiunile Python Social Auth pentru configurare. Diferențe notabile:

- Weblate acceptă un singur IDP care trebuie să fie numit `weblate` în `SOCIAL_AUTH_SAML_ENABLED_IDPS`.
- URL-ul metadatelor SAML XML este `/accounts/metadata/saml/`.
- Următoarele setări sunt completate automat: `SOCIAL_AUTH_SAML_SP_ENTITY_ID`, `SOCIAL_AUTH_SAML_TECHNICAL_CONTACT`, `SOCIAL_AUTH_SAML_SUPPORT_CONTACT`, `SOCIAL_AUTH_SAML_SUPPORT_CONTACT`

Exemplu de configurare:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.email.EmailAuth",
    "social_core.backends.saml.SAMLAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)
```

(continues on next page)

(continuare din pagina precedentă)

```
# Social auth backends setup
SOCIAL_AUTH_SAML_SP_ENTITY_ID = f"https://{SITE_DOMAIN}/accounts/metadata/saml/"
SOCIAL_AUTH_SAML_SP_PUBLIC_CERT = "-----BEGIN CERTIFICATE-----"
SOCIAL_AUTH_SAML_SP_PRIVATE_KEY = "-----BEGIN PRIVATE KEY-----"
SOCIAL_AUTH_SAML_ENABLED_IDPS = {
    "weblate": {
        "entity_id": "https://idp.testshib.org/idp/shibboleth",
        "url": "https://idp.testshib.org/idp/profile/SAML2/Redirect/SSO",
        "x509cert": "MIIEDjCCAvagAwIBAgIBADA ... 8Bbnl+ev0peYzxFyF5sQA==",
        "attr_name": "full_name",
        "attr_username": "username",
        "attr_email": "email",
    }
}
SOCIAL_AUTH_SAML_ORG_INFO = {
    "en-US": {
        "name": "example",
        "displayname": "Example Inc.",
        "url": "http://example.com"
    }
}
SOCIAL_AUTH_SAML_TECHNICAL_CONTACT = {
    "givenName": "Tech Gal",
    "emailAddress": "technical@example.com"
}
SOCIAL_AUTH_SAML_SUPPORT_CONTACT = {
    "givenName": "Support Guy",
    "emailAddress": "support@example.com"
}
```

The default configuration extracts user details from following attributes, configure your IDP to provide them:

Attribute	SAML URI reference
Nume și prenume	`urn:oid:2.5.4.3`
First name	`urn:oid:2.5.4.42`
Last name	`urn:oid:2.5.4.4`
Email	`urn:oid:0.9.2342.19200300.100.1.3`
Nume utilizator	`urn:oid:0.9.2342.19200300.100.1.1`

Sugestie: The example above and the Docker image define an IDP labelled `weblate`. You might need to configure this string as *Relay* in your IDP.

Vezi și:

Configuring SAML in Docker, SAML

2.5.6 Autentificare LDAP

Autentificarea LDAP poate fi realizată cel mai bine folosind pachetul *django-auth-ldap*. Îl puteți instala prin mijloace obișnuite:

```
# Using PyPI
pip install django-auth-ldap>=1.3.0

# Using apt-get
apt-get install python-django-auth-ldap
```

Sugestie: This package is included in the Docker container, see *Installing using Docker*.

Notă: Există unele incompatibilități în modulul Python LDAP 3.1.0, care vă pot împiedica să utilizați această versiune. Dacă primiți eroarea AttributeError: «module» object has no attribute «_trace_level», ar putea fi de ajutor dacă faceți downgrade la python-ldap 3.0.0.

După ce ați instalat pachetul, îl puteți conecta la autentificarea Django:

```
# Add LDAP backed, keep Django one if you want to be able to sign in
# even without LDAP for admin account
AUTHENTICATION_BACKENDS = (
    "django_auth_ldap.backend.LDAPBackend",
    "weblate.accounts.auth.WeblateUserBackend",
)

# LDAP server address
AUTH_LDAP_SERVER_URI = "ldaps://ldap.example.net"

# DN to use for authentication
AUTH_LDAP_USER_DN_TEMPLATE = "cn=%(user)s,o=Example"
# Depending on your LDAP server, you might use a different DN
# like:
# AUTH_LDAP_USER_DN_TEMPLATE = 'ou=users,dc=example,dc=com'

# List of attributes to import from LDAP upon sign in
# Weblate stores full name of the user in the full_name attribute
AUTH_LDAP_USER_ATTR_MAP = {
    "full_name": "name",
    # Use the following if your LDAP server does not have full name
    # Weblate will merge them later
    # 'first_name': 'givenName',
    # 'last_name': 'sn',
    # Email is required for Weblate (used in VCS commits)
    "email": "mail",
}

# Hide the registration form
REGISTRATION_OPEN = False
```

Notă: Ar trebui să eliminați `'social_core.backends.email.EmailAuth'` din setarea `AUTHENTICATION_BACKENDS`, altfel utilizatorii își vor putea seta parola în Weblate și se vor putea autentifica folosind-o. Păstrarea `'weblate.accounts.auth.WeblateUserBackend'` este încă necesară pentru a face permisiuni și a facilita utilizatorii anonimi. De asemenea, vă va permite să vă autentificați folosind un cont de administrator local, dacă l-ați creat (e.g. by using `createadmin`).

Utilizarea parolei bind

Dacă nu puteți utiliza direct bind pentru autentificare, va trebui să utilizați căutarea și să furnizați un utilizator care să se lege pentru căutare. De exemplu:

```
import ldap
from django_auth_ldap.config import LDAPSearch

AUTH_LDAP_BIND_DN = ""
AUTH_LDAP_BIND_PASSWORD = ""
AUTH_LDAP_USER_SEARCH = LDAPSearch(
    "ou=users,dc=example,dc=com", ldap.SCOPE_SUBTREE, "(uid=%(user)s)"
)
```

Active Directory integration

```
import ldap
from django_auth_ldap.config import LDAPSearch, NestedActiveDirectoryGroupType

AUTH_LDAP_BIND_DN = "CN=ldap,CN=Users,DC=example,DC=com"
AUTH_LDAP_BIND_PASSWORD = "password"

# User and group search objects and types
AUTH_LDAP_USER_SEARCH = LDAPSearch(
    "CN=Users,DC=example,DC=com", ldap.SCOPE_SUBTREE, "(sAMAccountName=%(user)s)"
)

# Make selected group a superuser in Weblate
AUTH_LDAP_USER_FLAGS_BY_GROUP = {
    # is_superuser means user has all permissions
    "is_superuser": "CN=weblate_AdminUsers,OU=Groups,DC=example,DC=com",
}

# Map groups from AD to Weblate
AUTH_LDAP_GROUP_SEARCH = LDAPSearch(
    "OU=Groups,DC=example,DC=com", ldap.SCOPE_SUBTREE, "(objectClass=group)"
)
AUTH_LDAP_GROUP_TYPE = NestedActiveDirectoryGroupType()
AUTH_LDAP_FIND_GROUP_PERMS = True

# Optionally enable group mirroring from LDAP to Weblate
# AUTH_LDAP_MIRROR_GROUPS = True
```

Vezi și:

Django Authentication Using LDAP, Authentication

2.5.7 Autentificare CAS

Autentificarea CAS poate fi realizată cu ajutorul unui pachet precum *django-cas-ng*.

Primul pas constă în dezvăluirea câmpului de e-mail al utilizatorului prin intermediul CAS. Acest lucru trebuie configurat chiar pe serverul CAS și necesită să rulați cel puțin CAS v2, deoarece CAS v1 nu acceptă deloc atribute.

Al doilea pas este actualizarea Weblate pentru a utiliza serverul CAS și atributele.

Pentru a instala *django-cas-ng*:

```
pip install django-cas-ng
```

După ce ați instalat pachetul, îl puteți conecta la sistemul de autentificare Django modificând fișierul `settings.py`:

```
# Add CAS backed, keep the Django one if you want to be able to sign in
# even without LDAP for the admin account
AUTHENTICATION_BACKENDS = (
    "django_cas_ng.backends.CASBackend",
    "weblate.accounts.auth.WeblateUserBackend",
)

# CAS server address
CAS_SERVER_URL = "https://cas.example.net/cas/"

# Add django_cas_ng somewhere in the list of INSTALLED_APPS
INSTALLED_APPS = (..., "django_cas_ng")
```

În cele din urmă, se poate utiliza un semnal pentru a corela câmpul de e-mail cu obiectul utilizator. Pentru ca acest lucru să funcționeze, trebuie să importați semnalul din pachetul *django-cas-ng* și să vă conectați codul cu acest semnal. Făcând acest lucru în fișierul de setări poate cauza probleme, de aceea este sugerat să îl puneți:

- În metoda `django.apps.AppConfig.ready()` din configurația aplicației dvs
- În fișierul `urls.py` al proiectului (atunci când nu există modele)

```
from django_cas_ng.signals import cas_user_authenticated
from django.dispatch import receiver


@receiver(cas_user_authenticated)
def update_user_email_address(sender, user=None, attributes=None, **kwargs):
    # If your CAS server does not always include the email attribute
    # you can wrap the next two lines of code in a try/catch block.
    user.email = attributes["email"]
    user.save()
```

Vezi și:

Django CAS NG

2.5.8 Configurarea autentificării Django de la o terță parte

În general, orice plugin de autentificare Django ar trebui să funcționeze cu Weblate. Trebuie doar să urmați instrucțiunile pentru plugin, dar nu uitați să păstrați instalat backend-ul de utilizator Weblate.

Vezi și:

Autentificare LDAP, *Autentificare CAS*

De obicei, instalarea va consta în adăugarea unui backend de autentificare la `AUTHENTICATION_BACKENDS` și instalarea unei aplicații de autentificare (dacă există) în `INSTALLED_APPS`:

```
AUTHENTICATION_BACKENDS = (
    # Add authentication backend here
    "weblate.accounts.auth.WeblateUserBackend",
)

INSTALLED_APPS += (
    # Install authentication app here
)
```

2.6 Controlul accesului

Weblate este dotat cu un sistem de privilegii fin pentru a atribui permisiuni utilizatorilor pentru întreaga instanță sau pentru un domeniu limitat.

Schimbat în versiunea 3.0: Before Weblate 3.0, the privilege system was based on Django privilege system only, but is specifically built for Weblate now. If using anything older, please consult the documentation for the specific version you are using.

2.6.1 Cotrol acces simplu

Dacă nu administrați întreaga instalație Weblate și aveți acces doar pentru a gestiona anumite proiecte (ca în Hosted Weblate), opțiunile de gestionare a controlului accesului sunt limitate la următoarele setări. Dacă nu aveți nevoie de o configurare complexă, acestea sunt suficiente pentru dumneavoastră.

Control acces proiect

Notă: Această funcție nu este disponibilă pentru proiectele care rulează planul Libre pe Hosted Weblate.

Puteți limita accesul utilizatorilor la proiecte individuale selectând o altă setare *Controlul accesului*. Opțiunile disponibile sunt:

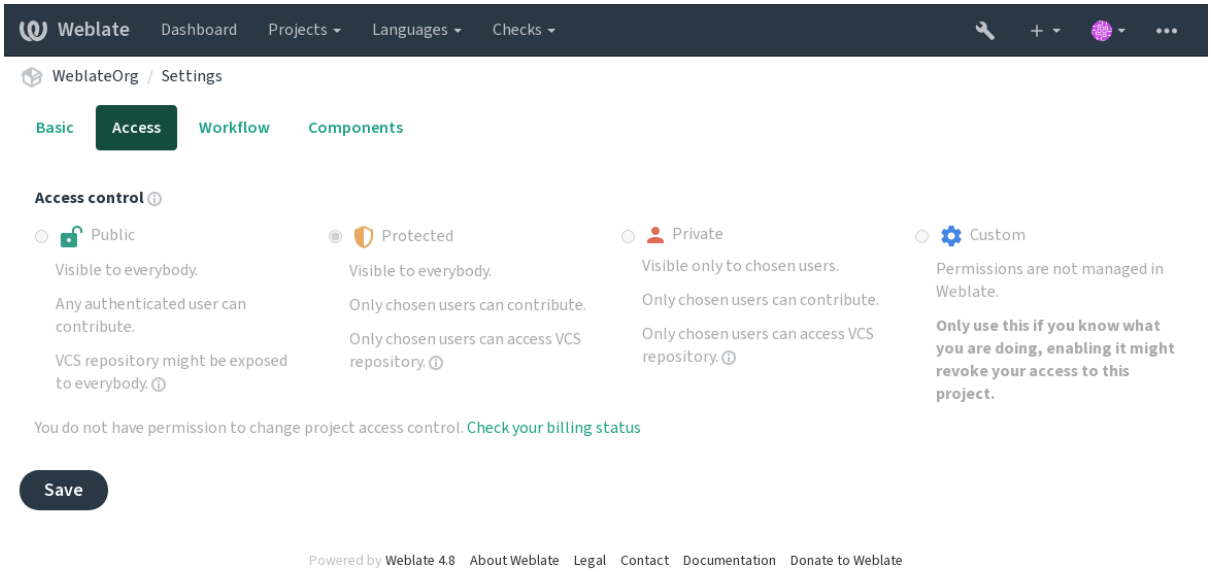
public Vizibil în mod public, traductibil pentru toți utilizatorii conectați.

Protejat Vizibil în mod public, dar traductibil doar pentru utilizatorii selectați.

Privat Vizibil și traductibil numai pentru utilizatorii selectați.

Personalizat *User management* vor fi dezactivate; în mod implicit, tuturor utilizatorilor le este interzis să efectueze orice acțiune în proiect. Va trebui să configurați toate permisiunile folosind *Control acces personalizat*.

Acces control poate fi modificat în fila *Acces* din configurația (*Manage* ↓ *Settings*) a fiecărui proiect respectiv.



The default value can be changed by `DEFAULT_ACCESS_CONTROL`.

Notă: Chiar și în cazul proiectelor *Private*, unele informații despre proiectul dumneavoastră vor fi expuse: statisticile și rezumatul limbii pentru întreaga instanță vor include cifrele pentru toate proiectele, în ciuda setărilor de control al

accesului. Numele proiectului dvs. și alte informații nu pot fi dezvăluite prin intermediul acestui sistem.

Notă: Setul real de permisiuni disponibile în mod implicit pentru utilizatori în proiectele *Public*, *Protected* și *Private* poate fi redefinit de administratorul instanței Weblate folosind *custom settings*.

Atenționare: Prin activarea controlului de acces *Custom*, Weblate va elimina toate *special groups* pe care le-a creat pentru un proiect selectat. Dacă faceți acest lucru fără a avea permisiunea de administrator pentru întreaga instanță Weblate, vă veți pierde instantaneu accesul la gestionarea proiectului.

Vezi și:

Controlul accesului

Managing per-project access control

Utilizatorii cu privilegiul *Manage project access* (see *List of privileges*) pot gestiona utilizatori în proiecte cu control de acces non-*Custom*. Aceștia pot atribui utilizatori la unul dintre următoarele grupuri.

Pentru proiecte *Public*, *Protected* and *Private*:

Administrare Include toate permisiunile disponibile pentru proiect.

Revizuire (numai dacă *review workflow* este activat) Poate aproba traducerile în timpul revizuirii.

Numai pentru proiectele *Protected* și *Private*:

Tradu Poate traduce proiectul și poate încărca traducerile făcute offline.

Sources Poate edita șirurile de surse (dacă este permis în *project settings*) și informații despre șirurile de surse.

Limbi Can manage translated languages (add or remove translations).

Glosar Can manage glossary (add or remove entries, also upload).

Memorie Can manage translation memory.

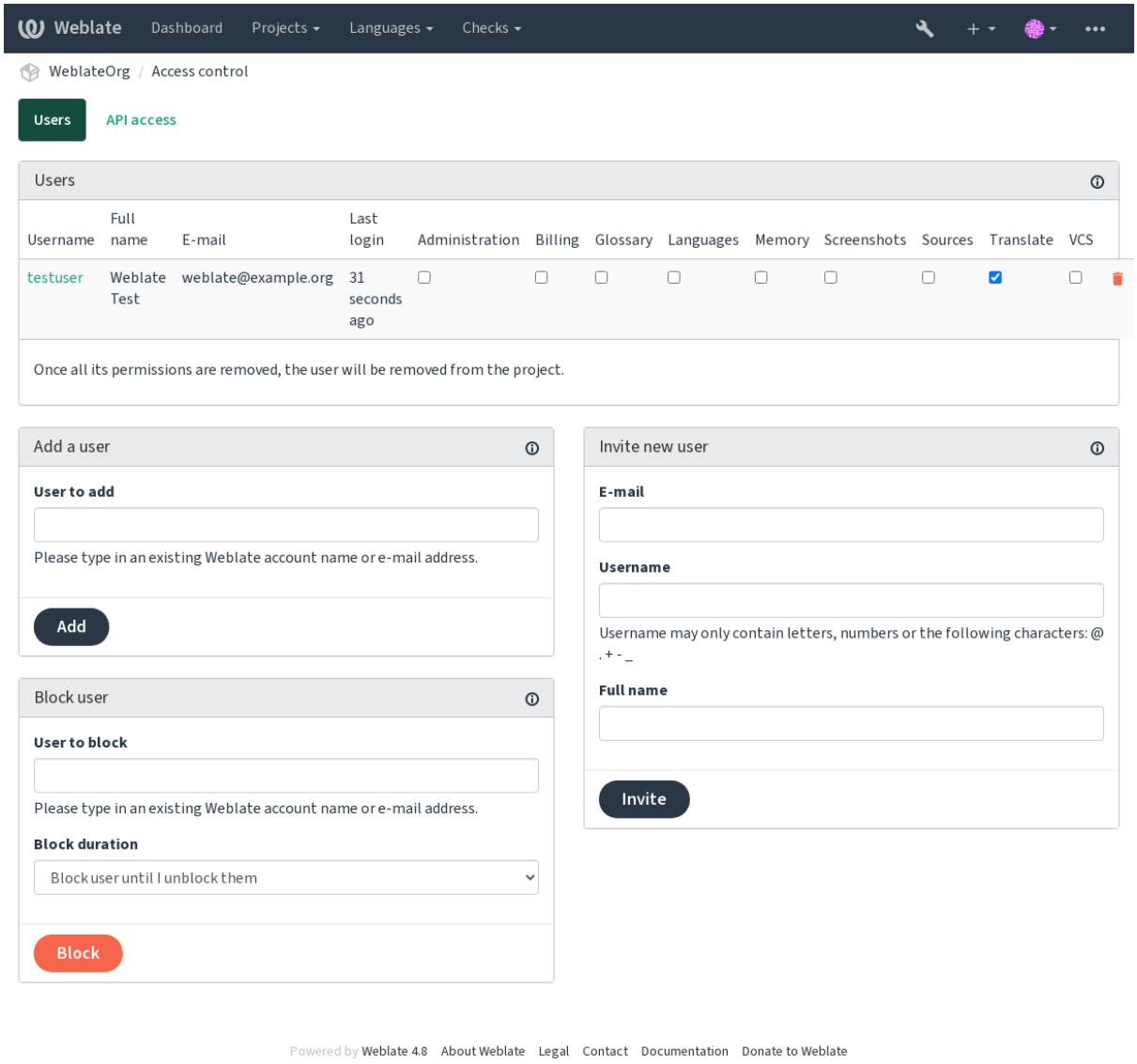
Capturi de ecran Poate gestiona capturi de ecran (adăugați sau eliminați-le și asociați-le la șiruri de surse).

VCS Poate gestiona VCS și poate accesa depozitul exportat.

Facturare Poate accesa informațiile și setările de facturare (see *Facturare*).

Din păcate, deocamdată nu este posibil să modificați acest set predefinit de grupuri. De asemenea, în acest fel nu este posibil să se acorde doar câteva permisiuni suplimentare tuturor utilizatorilor.

Notă: Pentru controlul accesului non-*Custom* se definește de fapt o instanță a fiecărui grup descris mai sus pentru fiecare proiect. Numele real al acestor grupuri va fi `Project@Group`, afișat și în interfața de administrare Django în acest mod. Deși acestea nu pot fi editate din interfața de utilizator Weblate.



Aceste caracteristici sunt disponibile pe pagina *Access control*, care poate fi accesată din meniul proiectului *Manage ↓ Users*.

New user invitation

De asemenea, pe lângă adăugarea unui utilizator existent la proiect, este posibilă și invitarea altora noi. Orice utilizator nou va fi creat imediat, dar contul va rămâne inactiv până la conectarea cu ajutorul unui link din invitația trimisă prin e-mail. Nu este necesar să aveți privilegii la nivelul întregului site pentru a face acest lucru, permisiunea de gestionare a accesului pe domeniul de aplicare al proiectului (e.g. a membership in the *Administration* group).

Sugestie: În cazul în care utilizatorul invitat a ratat valabilitatea invitației, acesta își poate seta parola folosind adresa de e-mail invitată în formularul de resetare a parolei, deoarece contul este deja creat.

Nou în versiunea 3.11: Este posibil să se retrimită e-mailul pentru invitațiile utilizatorilor (invalidând orice invitație trimisă anterior).

Același tip de invitații sunt disponibile la nivelul întregului site din *management interface* din fila *Users*.

Blocking users

Nou în versiunea 4.7.

In case some users behave badly in your project, you have an option to block them from contributing. The blocked user still will be able to see the project if he has permissions for that, but he won't be able to contribute.

Gestionarea permisiunilor per proiect

Puteți seta proiectele dvs. la *Protected* sau *Private* și *manage users* per-proiect în interfața de utilizator Weblate.

În mod implicit, acest lucru împiedică Weblate să acorde accesul oferit de *Users* și *Viewers* *default groups* din cauza configurației proprii a acestor grupuri. Acest lucru nu vă împiedică să acordați permisiuni pentru aceste proiecte la nivelul întregului site prin modificarea grupurilor implicite, prin crearea unuia nou sau prin crearea unor setări personalizate suplimentare pentru componente individuale, așa cum este descris în *Control acces personalizat* de mai jos.

Unul dintre principalele avantaje ale gestionării permisiunilor prin intermediul interfeței de utilizator Weblate este că puteți delega acest lucru altor utilizatori fără a le acorda privilegiul de superutilizator. Pentru a face acest lucru, adăugați-i la grupul *Administration* al proiectului.

2.6.2 Control acces personalizat

Notă: Această funcție nu este disponibilă pentru proiectele care rulează planul Libre pe Hosted Weblate.

The permission system is based on groups and roles, where roles define a set of permissions, and groups link them to users and translations, see *Users, roles, groups, and permissions* for more details.

Cele mai puternice caracteristici ale sistemului de control al accesului la Weblate sunt deocamdată disponibile doar prin intermediul interfeței de administrare *Django*. O puteți utiliza pentru a gestiona permisiunile oricărui proiect. Nu trebuie neapărat să o comutați la *Custom controlul accesului* pentru a o utiliza. Cu toate acestea, trebuie să aveți privilegii de superutilizator pentru a o utiliza.

Dacă nu vă interesează detaliile implementării și doriți doar să creați o configurație suficient de simplă bazată pe valorile implicite sau dacă nu aveți acces la întreaga instalație Weblate la nivelul întregului site (ca în Hosted Weblate), vă rugăm să consultați secțiunea *Cotrol acces simplu*.

Configurații comune

Această secțiune conține o prezentare generală a unor configurații comune care ar putea să vă intereseze.

Site-wide permission management

Pentru a gestiona simultan permisiunile pentru o întreagă instanță, adăugați utilizatorii la grupurile corespunzătoare *default groups*:

- *Users* (this is done by default by the *automatic group assignment*).
- *Reviewers* (if you are using *review workflow* with dedicated reviewers).
- *Managers* (dacă doriți să delegați majoritatea operațiunilor de gestionare altcuiva).

Ar trebui să păstrați toate proiectele configurate ca *Public* (see *Control acces proiect*)), în caz contrar permisiunile la nivel de site oferite de apartenența la grupurile *Users* și *Reviewers* nu vor avea niciun efect.

De asemenea, puteți acorda anumite permisiuni suplimentare la alegerea dumneavoastră grupurilor implicite. De exemplu, este posibil să doriți să acordați permisiunea de a gestiona capturi de ecran tuturor „Utilizatorilor".

De asemenea, puteți defini noi grupuri personalizate. Dacă doriți să continuați să gestionați permisiunile la nivelul întregului site pentru aceste grupuri, alegeți o valoare corespunzătoare pentru *Project selection* (e.g. *All projects* or *All public projects*).

Custom permissions for languages, components or projects

Puteți crea propriile grupuri dedicate pentru a gestiona permisiunile pentru obiecte distincte, cum ar fi limbile, componentele și proiectele. Deși aceste grupuri pot acorda doar privilegii suplimentare, nu puteți revoca nicio permisiune acordată de grupurile la nivel de site sau pe proiect prin adăugarea unui alt grup personalizat.

Example:

If you want (for whatever reason) to allow translation to a specific language (lets say *Czech*) only to a closed set of reliable translators while keeping translations to other languages public, you will have to:

1. Remove the permission to translate *Czech* from all the users. In the default configuration this can be done by altering the *Users* *default group*.

Tabelul 1: Group *Users*

Selectarea limbii	*As defined*
Limbi	All but *Czech*

2. Add a dedicated group for *Czech* translators.

Tabelul 2: Group *Czech translators*

Roluri	*Power users*
Selecția proiectului	*All public projects*
Selectarea limbii	*As defined*
Limbi	*Czech*

3. Add users you wish to give the permissions to into this group.

As you can see, permissions management this way is powerful, but can be quite a tedious job. You can't delegate it to another user, unless granting superuser permissions.

Users, roles, groups, and permissions

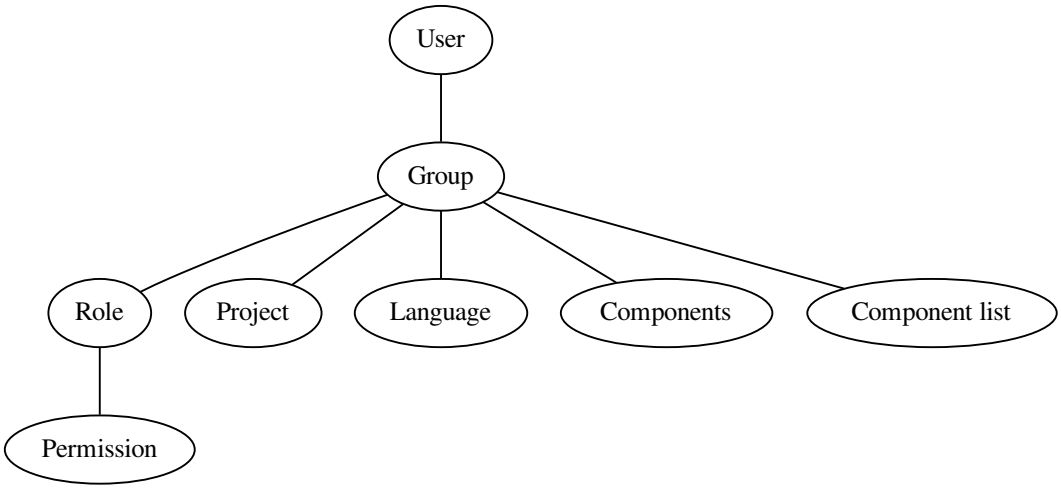
The authentication models consist of several objects:

Permission Individual permission defined by Weblate. Permissions cannot be assigned to users. This can only be done through assignment of roles.

Role A role defines a set of permissions. This allows reuse of these sets in several places, making the administration easier.

User User can belong to several groups.

Group Group connect roles, users, and authentication objects (projects, languages, and component lists).



Notă: A group can have no roles assigned to it, in that case access to browse the project by anyone is assumed (see below).

Access for browse to a project

A user has to be a member of a group linked to the project, or any component inside that project. Having membership is enough, no specific permissions are needed to browse the project (this is used in the default *Viewers* group, see *Lista de grupuri*).

Access for browse to a component

A user can access unrestricted components once able to access the components' project (and will have all the permissions the user was granted for the project). With *Restricted access* turned on, access to the component requires explicit permissions for the component (or a component list the component is in).

Scope of groups

The scope of the permission assigned by the roles in the groups are applied by the following rules:

- If the group specifies any *Component list*, all the permissions given to members of that group are granted for all the components in the component lists attached to the group, and an access with no additional permissions is granted for all the projects these components are in. *Components* and *Projects* are ignored.
- If the group specifies any *Components*, all the permissions given to the members of that group are granted for all the components attached to the group, and an access with no additional permissions is granted for all the projects these components are in. *Projects* are ignored.
- Otherwise, if the group specifies any *Projects*, either by directly listing them or by having *Projects selection* set to a value like *All public projects*, all those permissions are applied to all the projects, which effectively grants the same permissions to access all projects *unrestricted components*.
- The restrictions imposed by a group's *Languages* are applied separately, when it's verified if a user has an access to perform certain actions. Namely, it's applied only to actions directly related to the translation process itself like reviewing, saving translations, adding suggestions, etc.

Sugestie: Use *Language selection* or *Project selection* to automate inclusion of all languages or projects.

Example:

Let's say there is a project `foo` with the components: `foo/bar` and `foo/baz` and the following group:

Tabelul 3: Group *Spanish Admin-Reviewers*

Roluri	*Review Strings*, *Manage repository*
Componente	foo/bar
Limbi	*Spanish*

Members of that group will have following permissions (assuming the default role settings):

- General (browsing) access to the whole project `foo` including both components in it: `foo/bar` and `foo/baz`.
- Review strings in `foo/bar` Spanish translation (not elsewhere).
- Manage VCS for the whole `foo/bar` repository e.g. commit pending changes made by translators for all languages.

Atribuirea automată a grupului

On the bottom of the *Group* editing page in the *Django admin interface*, you can specify *Automatic group assignments*, which is a list of regular expressions used to automatically assign newly created users to a group based on their e-mail addresses. This assignment only happens upon account creation.

The most common use-case for the feature is to assign all new users to some default group. In order to do so, you will probably want to keep the default value (`^.*$`) in the regular expression field. Another use-case for this option might be to give some additional privileges to employees of your company by default. Assuming all of them use corporate e-mail addresses on your domain, this can be accomplished with an expression like `^.*@mycompany.com`.

Notă: Automatic group assignment to *Users* and *Viewers* is always recreated when upgrading from one Weblate version to another. If you want to turn it off, set the regular expression to `^$` (which won't match anything).

Notă: As for now, there is no way to bulk-add already existing users to some group via the user interface. For that, you may resort to using the *REST API*.

Default groups and roles

After installation, a default set of groups is created (see *Lista de grupuri*).

These roles and groups are created upon installation. The built-in roles are always kept up to date by the database migration when upgrading. You can't actually change them, please define a new role if you want to define your own set of permissions.

List of privileges

Billing (see *Facturare*) View billing info [*Administration*, *Billing*]

Modificări Download changes [*Administration*]

Comentarii Postați un comentariu [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

Delete comment [*Administration*]

Componenta Edit component settings [*Administration*]

Lock component, preventing translations [*Administration*]

Glosar Adăugați o intrare în glosar [*Administration*, *Manage glossary*, *Power user*]

Editarea intrării în glosar [*Administration*, *Manage glossary*, *Power user*]

Ștergeți intrarea în glosar [*Administration*, *Manage glossary*, *Power user*]

Încărcați intrările din glosar [*Administration*, *Manage glossary*, *Power user*]

Sugestii automate Use automatic suggestions [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

Memorie de traducere Edit translation memory [*Administration*, *Manage translation memory*]

Delete translation memory [*Administration*, *Manage translation memory*]

Proiecte Edit project settings [*Administration*]

Manage project access [*Administration*]

Rapoarte Descărcați rapoarte [*Administration*]

Capturi de ecran Adăugați captură de ecran [*Administration*, *Manage screenshots*]

Editare captură de ecran [*Administration*, *Manage screenshots*]

Ștergere captură de ecran [*Administration*, *Manage screenshots*]

Șiruri de surse Edit additional string info [*Administration*, *Edit source*]

Șiruri de caractere Add new string [*Administration*]

Remove a string [*Administration*]

Dismiss failing check [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

Editarea șirurilor de caractere [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

Șiruri de revizuire [*Administration*, *Review strings*]

Edit string when suggestions are enforced [*Administration*, *Review strings*]

Editarea șirurilor de surse [*Administration*, *Edit source*, *Power user*]

Sugestii Acceptă sugestia [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

Adăugare sugestie [*Administration*, *Edit source*, *Add suggestion*, *Power user*, *Review strings*, *Translate*]

Delete suggestion [*Administration*, *Power user*]

Vote on suggestion [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

Traduceri Add language for translation [*Administration*, *Power user*, *Manage languages*]

Efectuarea traducerii automate [*Administration*, *Manage languages*]

Delete existing translation [*Administration*, *Manage languages*]

Add several languages for translation [*Administration*, *Manage languages*]

Încărcături Define author of uploaded translation [*Administration*]

Suprascrieți șirurile existente cu încărcare [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

Upload translations [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

VCS Access the internal repository [*Administration*, *Access repository*, *Power user*, *Manage repository*]

Trimiterea modificărilor în depozitul intern [*Administration*, *Manage repository*]

Push change from the internal repository [*Administration*, *Manage repository*]

Resetarea modificărilor din depozitul intern [*Administration*, *Manage repository*]

View upstream repository location [*Administration*, *Access repository*, *Power user*, *Manage repository*]

Update the internal repository [*Administration*, *Manage repository*]

Privilegii la nivelul întregului site Utilizați interfața de gestionare

Adăugați proiecte noi

Adăugați definiții de limbă

Gestionați definițiile de limbă

Gestionați grupurile

Gestionați utilizatorii

Gestionați rolurile

Gestionați anunțurile

Gestionați memoria de traducere

Gestionați listele de componente

Notă: Nu se acordă privilegii la nivelul întregului site pentru niciun rol implicit. Acestea sunt puternice și destul de apropiate de statutul de superutilizator. Cele mai multe dintre ele afectează toate proiectele din instalarea Weblate.

Lista de grupuri

Următoarele grupuri sunt create la instalare (or after executing `setupgroups`) și puteți să le modificați. Cu toate acestea, migrația le va recrea dacă le ștergeți sau le redenumiți.

Guests Definește permisiunile pentru utilizatorii neautentificați.

Acest grup conține numai utilizatori anonimi (see `ANONYMOUS_USER_NAME`).

Puteți elimina rolurile din acest grup pentru a limita permisiunile pentru utilizatorii neautentificați.

Roluri implicite: *Adaugați sugestii*, *Accesați depozitul*

Viewers Acest rol asigură vizibilitatea proiectelor publice pentru toți utilizatorii. În mod implicit, toți utilizatorii sunt membri ai acestui grup.

În mod implicit, *automatic group assignment* face ca toate conturile noi să devină membri ai acestui grup atunci când se alătură.

Roluri implicite: niciunul

Users Grup implicit pentru toți utilizatorii.

În mod implicit, *automatic group assignment* face ca toate conturile noi să devină membri ai acestui grup atunci când se alătură.

Roluri implicite: *Power user*

Reviewers Grup pentru recenzori (see *Translation workflows*).

Roluri implicite: *Review strings*

Managers Grup pentru administratori.

Roluri implicite: *Administration*

Atenționare: Nu eliminați niciodată grupurile și utilizatorii Weblate predefinite, deoarece acest lucru poate duce la probleme neașteptate! Dacă nu aveți nevoie de ei, puteți elimina în schimb toate privilegiile lor.

2.6.3 Additional access restrictions

Dacă doriți să folosiți instalația Weblate într-un mod mai puțin public, adică să permiteți noilor utilizatori doar pe bază de invitație, acest lucru se poate face prin configurarea Weblate astfel încât doar utilizatorii cunoscuți să aibă acces la el. Pentru a face acest lucru, puteți seta `REGISTRATION_OPEN` la `False` pentru a împiedica înregistrarea oricăror utilizatori noi și setați `REQUIRE_LOGIN` la `/.*` pentru a solicita autentificarea pentru a accesa toate paginile site-ului. Aceasta este practic modalitatea de a bloca instalarea Weblate.

Sugestie: Puteți utiliza built-in *invitations* pentru a adăuga noi utilizatori.

2.7 Proiecte de traducere

2.7.1 Translation organization

Weblate organizes translatable VCS content of project/components into a tree-like structure.

- The bottom level object is *Project configuration*, which should hold all translations belonging together (for example translation of an application in several versions and/or accompanying documentation).
- On the level above, *Component configuration*, which is actually the component to translate, you define the VCS repository to use, and the mask of files to translate.
- Above *Component configuration* there are individual translations, handled automatically by Weblate as translation files (which match *File mask* defined in *Component configuration*) appear in the VCS repository.

Weblate supports a wide range of translation formats (both bilingual and monolingual ones) supported by Translate Toolkit, see *Supported file formats*.

Notă: You can share cloned VCS repositories using *Weblate internal URLs*. Using this feature is highly recommended when you have many components sharing the same VCS. It improves performance and decreases required disk space.

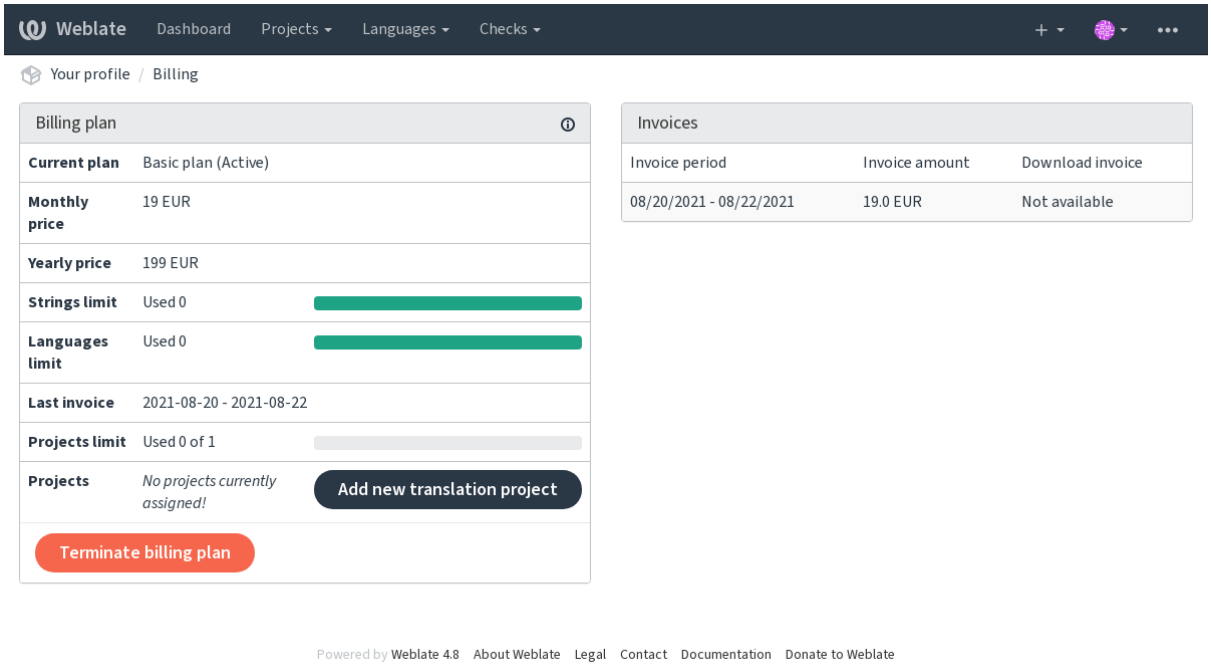
2.7.2 Adding translation projects and components

Schimbat în versiunea 3.2: An interface for adding projects and components is included, and you no longer have to use *Interfața de administrare Django*.

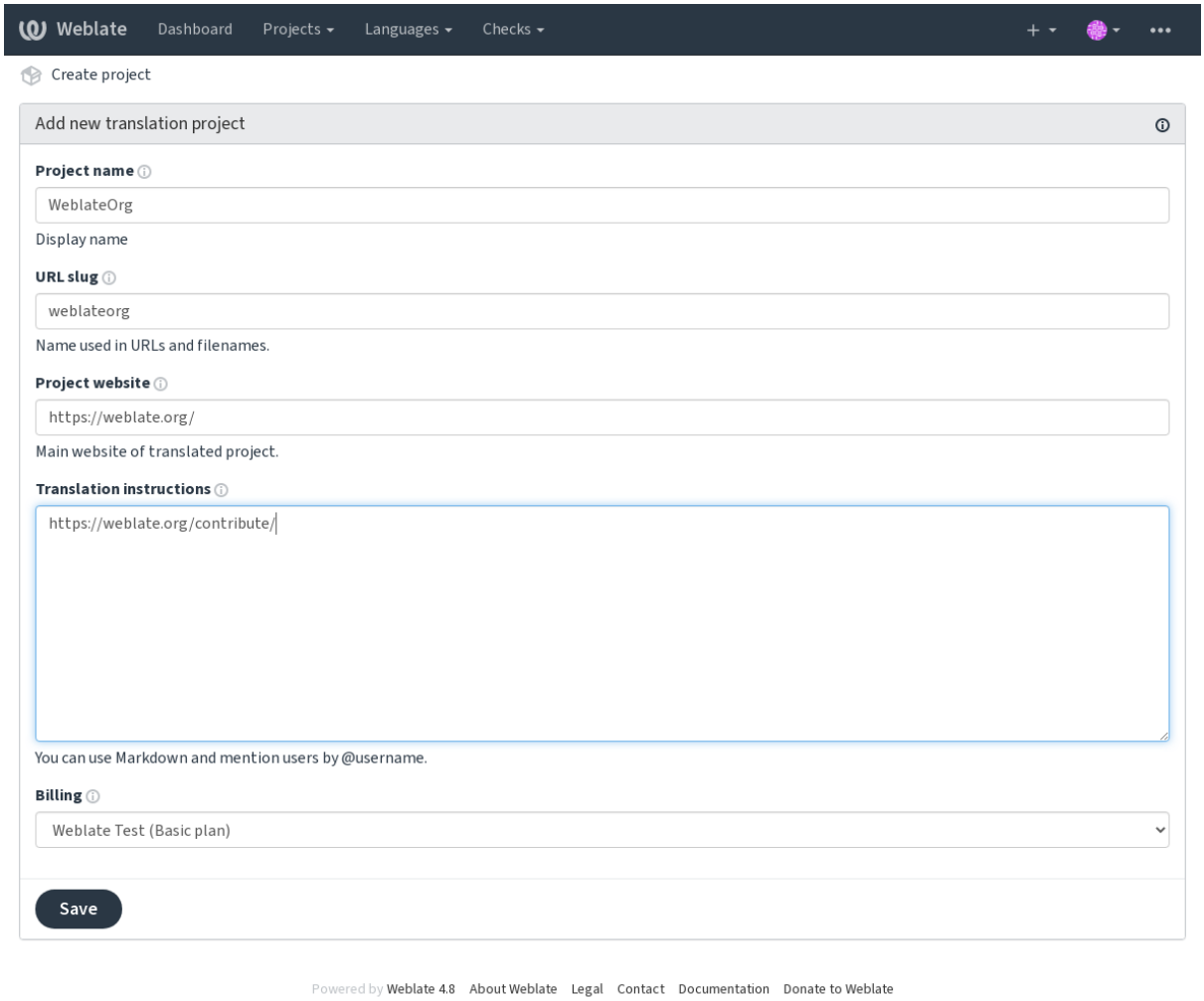
Schimbat în versiunea 3.4: The process of adding components is now multi staged, with automated discovery of most parameters.

Based on your permissions, new translation projects and components can be created. It is always permitted for users with the *Add new projects* permission, and if your instance uses billing (e.g. like https://hosted.weblate.org/ see *Facturare*), you can also create those based on your plans allowance from the user account that manages billing.

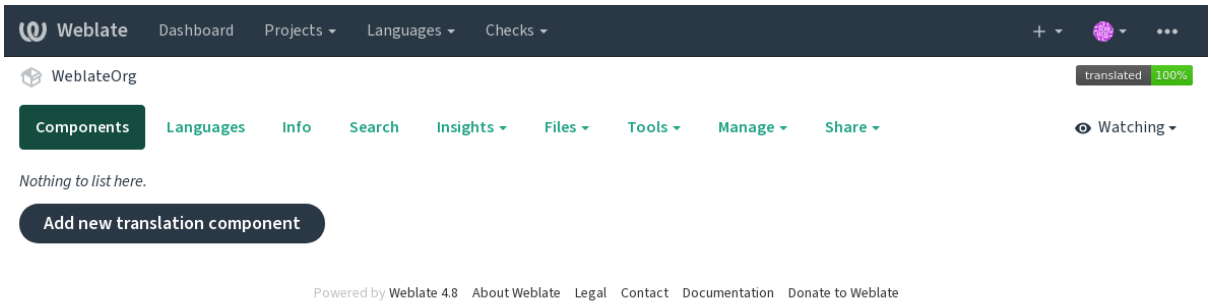
You can view your current billing plan on a separate page:



The project creation can be initiated from there, or using the menu in the navigation bar, filling in basic info about the translation project to complete addition of it:



After creating the project, you are taken directly to the project page:



Creating a new translation component can be initiated via a single click there. The process of creating a component is multi-staged and automatically detects most translation parameters. There are several approaches to creating component:

Din controlul versiunilor Creates component from remote version control repository.

Din componenta existentă Creates additional component to existing one by choosing different files.

Sucursală suplimentară Creates additional component to existing one, just for different branch.

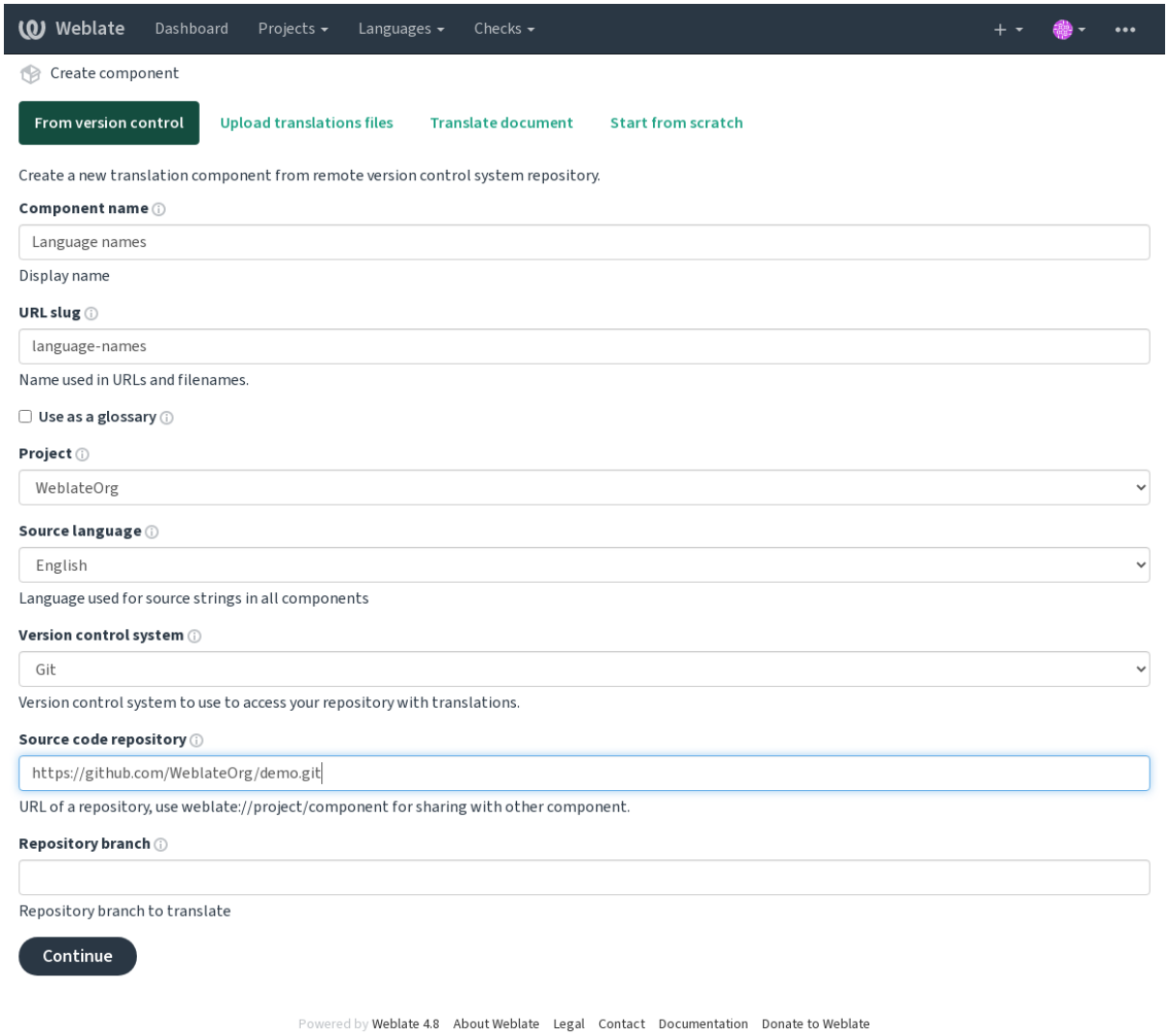
Încărcați fișiere de traducere Upload translation files to Weblate in case you do not have version control or do not want to integrate it with Weblate. You can later update the content using the web interface or *Weblate's REST API*.

Traduceți documentul Upload single document or translation file and translate that.

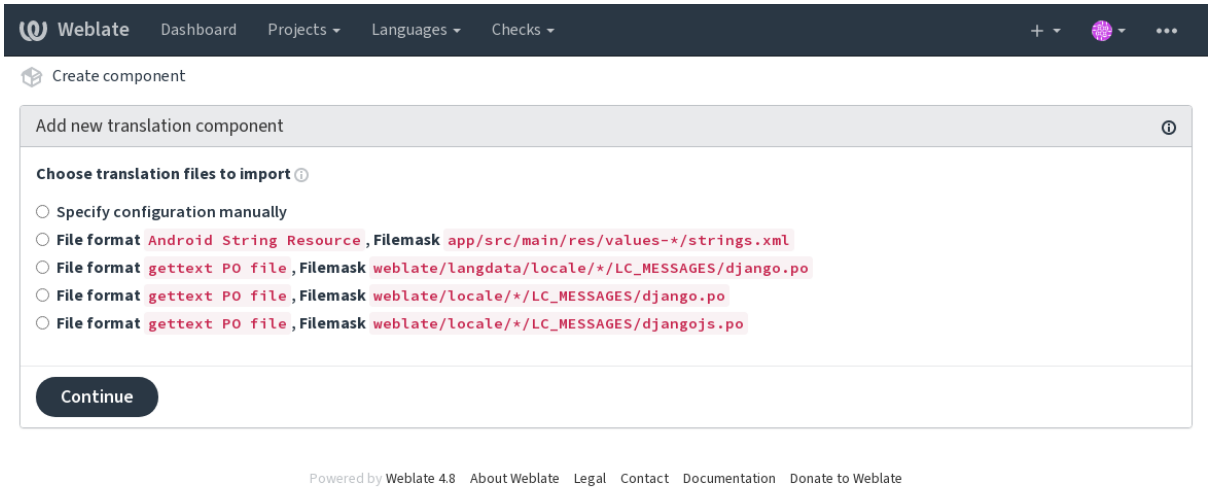
Începeți de la zero Create blank translation project and add strings manually.

Once you have existing translation components, you can also easily add new ones for additional files or branches using same repository.

First you need to fill in name and repository location:



On the next page, you are presented with a list of discovered translatable resources:



As a last step, you review the translation component info and fill in optional details:

Weblate Dashboard Projects Languages Checks

Create component

Add new translation component

Project

WeblateOrg

Component name

Language names

Display name

URL slug

language-names

Name used in URLs and filenames.

Version control system

Git

Version control system to use to access your repository containing translations. You can also choose additional integration with third party providers to submit merge requests.

Source code repository

https://github.com/WeblateOrg/demo.git

URL of a repository, use weblate://project/component to share it with other component.

Repository branch

Repository branch to translate

Repository push URL

URL of a push repository, pushing is turned off if empty.

Push branch

Branch for pushing changes, leave empty to use repository branch

Repository browser

https://github.com/WeblateOrg/demo/blob/{{branch}}/{{filename}}#L{{line}}

Link to repository browser, use {{branch}} for branch, {{filename}} and {{line}} as filename and line placeholders. You might want to strip leading directory by using {{filename|parentdir}}.

File format

gettext PO file

Filemask

weblate/langdata/locale/*/LC_MESSAGES/django.po

Path of files to translate relative to repository root, use * instead of language code, for example: po/*.po or locale/*/LC_MESSAGES/django.po.

Monolingual base language file

Filename of translation base file, containing all strings and their source; it is recommended for monolingual translation formats.

☑ **Edit base file**

Whether users will be able to edit the base file for monolingual translations.

Intermediate language file

Filename of intermediate translation file. In most cases this is a translation file provided by developers and is used when creating actual source strings.

Template for new translations

weblate/langdata/locale/django.pot

Filename of file used for creating new translations. For gettext choose .pot file.

Translation license

GNU General Public License v3.0 or later

Adding new translation

Create new language file

How to handle requests for creating new translations.

Language code style

Default based on the file format

Customize language code used to generate the filename for translations created by Weblate.

Language filter

^(cs|he|hu)$

Regular expression used to filter translation files when scanning for filemask.

Source language

English

Language used for source strings in all components

☐ **Use as a glossary**

You will be able to edit more options in the component settings after creating it.

Save

Powered by Weblate 4.8 About Weblate Legal Contact Documentation Donate to Weblate

Vezi și:

Interfața de administrare Django, *Project configuration*, *Component configuration*

2.7.3 Project configuration

Create a translation project and then add a new component for translation in it. The project is like a shelf, in which real translations are stacked. All components in the same project share suggestions and their dictionary; the translations are also automatically propagated through all components in a single project (unless turned off in the component configuration), see *Memorie de traducere*.

Vezi și:

/devel/integration

These basic attributes set up and inform translators of a project:

Denumire proiect

Verbose project name, used to display the project name.

URL slug

Project name suitable for URLs.

Website-ul proiectului

URL where translators can find more info about the project.

This is a required parameter unless turned off by `WEBSITE_REQUIRED`.

Instrucțiuni de traducere

URL to more site with more detailed instructions for translators.

Setați antetul „Language-Team”

Whether Weblate should manage the `Language-Team` header (this is a *GNU gettext* only feature right now).

Utilizați memoria de traducere partajată

Whether to use shared translation memory, see *Memorie partajată de traducere* for more details.

Default value is determined by `DEFAULT_SHARED_TM`.

Contribuie la memoria de traducere partajată

Whether to contribute to shared translation memory, see *Memorie partajată de traducere* for more details.

Default value is determined by `DEFAULT_SHARED_TM`.

Controlul accesului

Configure per project access control, see *Control acces proiect* for more details.

Default value can be changed by `DEFAULT_ACCESS_CONTROL`.

Activați recenzii

Enable review workflow for translations, see *Dedicated reviewers*.

Activați revizuirile sursei

Enable review workflow for source strings, see *Source strings reviews*.

Vezi și:

report-source, *Comentarii*

Activați cârligele

Whether unauthenticated *Cârlige de notificare* are to be used for this repository.

Vezi și:

Fișier de limbaj intermediar, *Quality gateway for the source strings*, *Bilingual and monolingual formats*, *Language definitions*

Aliasuri lingvistice

Define language codes mapping when importing translations into Weblate. Use this when language codes are inconsistent in your repositories and you want to get a consistent view in Weblate or in case you want to use non-standard naming of your translation files.

The typical use case might be mapping American English to English: `en_US:en`

Multiple mappings to be separated by comma: `en_GB:en,en_US:en`

Using non standard code: `ia_FOO:ia`

Sugestie: The language codes are mapped when matching the translation files and the matches are case sensitive, so make sure you use the source language codes in same form as used in the filenames.

Vezi și:

Parsing language codes

2.7.4 Component configuration

A component is a grouping of something for translation. You enter a VCS repository location and file mask for which files you want translated, and Weblate automatically fetches from this VCS, and finds all matching translatable files.

Vezi și:

/devel/integration

You can find some examples of typical configurations in the *Supported file formats*.

Notă: It is recommended to keep translation components to a reasonable size - split the translation by anything that makes sense in your case (individual apps or addons, book chapters or websites).

Weblate easily handles translations with 10000s of strings, but it is harder to split work and coordinate among translators with such large translation components.

Should the language definition for a translation be missing, an empty definition is created and named as „cs_CZ (generated)". You should adjust the definition and report this back to the Weblate authors, so that the missing languages can be included in next release.

The component contains all important parameters for working with the VCS, and for getting translations out of it:

Denumirea componentei

Verbose component name, used to display the component name.

Component slug

Component name suitable for URLs.

Component project

Project configuration where the component belongs.

Sistem de control al versiunilor

VCS to use, see *Integrarea controlului versiunilor* for details.

Vezi și:

Pushing changes from Weblate

Depozitul de cod sursă

VCS repository used to pull changes.

Vezi și:

See *Accessing repositories* for more details on specifying URLs.

Sugestie: This can either be a real VCS URL or `weblate://project/component` indicating that the repository should be shared with another component. See *Weblate internal URLs* for more details.

URL de împingere a depozitului

Repository URL used for pushing. This setting is used only for *Git* and *Mercurial* and push support is turned off for these when this is empty.

Vezi și:

See *Accessing repositories* for more details on how to specify a repository URL and *Pushing changes from Weblate* for more details on pushing changes from Weblate.

Navigator de depozite

URL of repository browser used to display source files (location of used messages). When empty, no such links will be generated. You can use *Template markup*.

For example on GitHub, use something like: `https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename}}#L{{line}}`

In case your paths are relative to different folder (path contains `..`), you might want to strip leading directory by `parentdir` filter (see *Template markup*): `https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename|parentdir}}#L{{line}}`

URL-ul depozitului exportat

URL where changes made by Weblate are exported. This is important when *Traducere continuă* is not used, or when there is a need to manually merge changes. You can use *Git exporter* to automate this for Git repositories.

Ramura de depozit

Which branch to checkout from the VCS, and where to look for translations.

Împingeți ramura

Branch for pushing changes, leave empty to use *Ramura de depozit*.

Notă: This is currently only supported for Git, GitLab and GitHub, it is ignored for other VCS integrations.

Vezi și:

Pushing changes from Weblate

File mask

Mask of files to translate, including path. It should include one „*” replacing language code (see *Language definitions* for info on how this is processed). In case your repository contains more than one translation file (e.g. more gettext domains), you need to create a component for each of them.

For example `po/*.po` or `locale/*/LC_MESSAGES/django.po`.

In case your filename contains special characters such as `[`, `]`, these need to be escaped as `[[]` or `[]]`.

Vezi și:

Bilingual and monolingual formats, *What does mean „There are more files for the single language (en)”?*

Fișier de limbă de bază monolingvă

Base file containing string definitions for *Componente monolingve*.

Vezi și:

Bilingual and monolingual formats, *What does mean „There are more files for the single language (en)”?*

Editarea fișierului de bază

Whether to allow editing the base file for *Componente monolingve*.

Fișier de limbaj intermediar

Intermediate language file for *Componente monolingve*. In most cases this is a translation file provided by developers and is used when creating actual source strings.

When set, the source strings are based on this file, but all other languages are based on *Fișier de limbă de bază monolingvă*. In case the string is not translated into the source langugage, translating to other languages is prohibited. This provides *Quality gateway for the source strings*.

Vezi și:

Quality gateway for the source strings, *Bilingual and monolingual formats*, *What does mean „There are more files for the single language (en)”?*

Șablon pentru traduceri noi

Base file used to generate new translations, e.g. `.pot` file with gettext.

Sugestie: In many monolingual formats Weblate starts with blank file by default. Use this in case you want to have all strings present with empty value when creating new translation.

Vezi și:

adding-translation, *Adding new translations*, *Adăugarea unei noi traduceri*, *Bilingual and monolingual formats*, *What does mean „There are more files for the single language (en)”?*

Formatul fișierului

Translation file format, see also *Supported file formats*.

Adresa de raportare a erorilor din șirul sursă

Email address used for reporting upstream bugs. This address will also receive notification about any source string comments made in Weblate.

Permiteți propagarea traducerii

You can turn off propagation of translations to this component from other components within same project. This really depends on what you are translating, sometimes it's desirable to have make use of a translation more than once.

It's usually a good idea to turn this off for monolingual translations, unless you are using the same IDs across the whole project.

Default value can be changed by `DEFAULT_TRANSLATION_PROPAGATION`.

Vezi și:

Keeping translations same across components

Enable suggestions

Whether translation suggestions are accepted for this component.

Votul prin sugestie

Turns on vote casting for suggestions, see *Votul prin sugestie*.

Sugestii de autoacceptare

Automatically accept voted suggestions, see *Votul prin sugestie*.

Steaguri de traducere

Customization of quality checks and other Weblate behavior, see *Personalizarea comportamentului cu ajutorul steguleţelor*.

Controale forțate

List of checks which can not be ignored, see *Executarea controalelor*.

Notă: Enforcing the check does not automatically enable it, you still should enabled it using *Personalizarea comportamentului cu ajutorul steguleţelor* in *Steaguri de traducere* or *Additional info on source strings*.

Licență de traducere

License of the translation (does not need to be the same as the source code license).

Acord de colaborare

Acordul de utilizare care trebuie aprobat înainte ca un utilizator să poată traduce această componentă.

Adăugarea unei noi traduceri

How to handle requests for creation of new languages. Available options:

Contactează întreținătorii User can select desired language and the project maintainers will receive a notification about this. It is up to them to add (or not) the language to the repository.

Indicați instrucțiunile de traducere URL User is presented a link to page which describes process of starting new translations. Use this in case more formal process is desired (for example forming a team of people before starting actual translation).

Crearea unui nou fișier de limbă User can select language and Weblate automatically creates the file for it and translation can begin.

Dezactivați adăugarea de noi traduceri There will be no option for user to start new translation.

Sugestie: The project admins can add new translations even if it is disabled here when it is possible (either *Șablon pentru traduceri noi* or the file format supports starting from an empty file).

Vezi și:

adding-translation, *Adding new translations*

Gestionați șirurile de caractere

Nou în versiunea 4.5.

Configures whether users in Weblate will be allowed to add new strings and remove existing ones. Adjust this to match your localization workflow - how the new strings are supposed to be introduced.

For bilingual formats, the strings are typically extracted from the source code (for example by using **`xgettext`**) and adding new strings in Weblate should be disabled (they would be discarded next time you update the translation files). In Weblate you can manage strings for every translation and it does not enforce the strings in all translations to be consistent.

For monolingual formats, the strings are managed only on source language and are automatically added or removed in the translations. The strings appear in the translation files once they are translated.

Vezi și:

Bilingual and monolingual formats, adding-new-strings, `POST /api/translations/(string:project)/(string:component)/(string:language)/units/`

Stil de cod lingvistic

Personalizați codul de limbă utilizat pentru a genera numele de fișier pentru traducerile create de Weblate.

Vezi și:

Adding new translations, *Codul limbii*, *Parsing language codes*

Stil de fuziune

You can configure how updates from the upstream repository are handled. This might not be supported for some VCSs. See *Merge or rebase* for more details.

Default value can be changed by `DEFAULT_MERGE_STYLE`.

Commit, add, delete, merge and addon messages

Message used when committing a translation, see *Template markup*.

Default value can be changed by `DEFAULT_ADD_MESSAGE`, `DEFAULT_ADDON_MESSAGE`, `DEFAULT_COMMIT_MESSAGE`, `DEFAULT_DELETE_MESSAGE`, `DEFAULT_MERGE_MESSAGE`.

Împingeți pe comitere

Whether committed changes should be automatically pushed to the upstream repository. When enabled, the push is initiated once Weblate commits changes to its underlying repository (see *Lazy commits*). To actually enable pushing *Repository push URL* has to be configured as well.

Vârsta modificărilor care urmează să fie confirmate

Sets how old (in hours) changes have to be before they are committed by background task or the `commit_pending` management command. All changes in a component are committed once there is at least one change older than this period.

Default value can be changed by `COMMIT_PENDING_HOURS`.

Sugestie: There are other situations where pending changes might be committed, see *Lazy commits*.

Blocare pe eroare

Locks the component (and linked components, see *Weblate internal URLs*) upon the first failed push or merge into its upstream repository, or pull from it. This avoids adding another conflicts, which would have to be resolved manually.

The component will be automatically unlocked once there are no repository errors left.

Limba sursă

Language used for source strings. Change this if you are translating from something else than English.

Sugestie: In case you are translating bilingual files from English, but want to be able to do fixes in the English translation as well, choose *English (Developer)* as a source language to avoid conflict between the name of the source language and the existing translation.

For monolingual translations, you can use intermediate translation in this case, see *Fișier de limbaj intermediar*.

Filtru lingvistic

Regular expression used to filter the translation when scanning for filemask. It can be used to limit the list of languages managed by Weblate.

Notă: You need to list language codes as they appear in the filename.

Some examples of filtering:

Filter description	Expresie regulată
Selected languages only	`^(cs|de|es)$`
Exclude languages	`^(?!(it|fr)$).+$`
Filter two letter codes only	`^..$`
Exclude non language files	`^(?!(blank)$).+$`
Include all files (default)	`^[^.]+$`

Variante expresie regulată

Regular expression used to determine the variants of a string, see variants.

Notă: Most of the fields can be edited by project owners or managers, in the Weblate interface.

Vezi și:

Does Weblate support other VCSes than Git and Mercurial?, alerts

Prioritate

Componentele cu prioritate mai mare sunt oferite mai întâi traducătorilor.

Restricted access

By default the component is visible to anybody who has access to the project, even if the person can not perform any changes in the component. This makes it easier to keep translation consistency within the project.

Restricting access at a component, or component-list level takes over access permission to a component, regardless of project-level permissions. You will have to grant access to it explicitly. This can be done through granting access to a new user group and putting users in it, or using the default *custom* or *private* access control groups.

The default value can be changed in `DEFAULT_RESTRICTED_COMPONENT`.

Sugestie: This applies to project admins as well — please make sure you will not loose access to the component after toggling the status.

Cota în proiecte

You can choose additional projects where the component will be visible. Useful for shared libraries which you use in several projects.

Notă: Sharing a component doesn't change its access control. It only makes it visible when browsing other projects. Users still need access to the actual component to browse or translate it.

Utilizarea ca glosar

Nou în versiunea 4.5.

Allows using this component as a glossary. You can configure how it will be listed using *Glosar culoare*.

The glossary will be accessible in all projects defined by *Cota în proiecte*.

It is recommended to enable *Gestionați șirurile de caractere* on glossaries in order to allow adding new words to them.

Vezi și:

Glosar

Glosar culoare

Display color for a glossary used when showing word matches.

2.7.5 Template markup

Weblate uses simple markup language in several places where text rendering is needed. It is based on The Django template language, so it can be quite powerful.

Currently it is used in:

- Commit message formatting, see *Component configuration*
- **Several addons**
  - *Descoperirea componentelor*
  - *Generator de statistici*
  - *Executing scripts from add-on*

There following variables are available in the component templates:

`{{ language_code }}` Codul limbii

`{{ language_name }}` Numele limbii

`{{ component_name }}` Denumirea componentei

`{{ component_slug }}` Component slug

`{{ project_name }}` Denumire proiect

`{{ project_slug }}` Project slug

`{{ url }}` Translation URL

`{{ filename }}` Numele fișierului de traducere

`{{ stats }}` Translation stats, this has further attributes, examples below.

`{{ stats.all }}` Total strings count

`{{ stats.fuzzy }}` Count of strings needing review

{{ stats.fuzzy_percent }} Percent of strings needing review

{{ stats.translated }} Translated strings count

{{ stats.translated_percent }} Translated strings percent

{{ stats.allchecks }} Number of strings with failing checks

{{ stats.allchecks_percent }} Percent of strings with failing checks

{{ author }} Author of current commit, available only in the commit scope.

{{ addon_name }} Name of currently executed addon, available only in the addon commit message.

The following variables are available in the repository browser or editor templates:

{{branch}} current branch

{{line}} line in file

{{filename}} filename, you can also strip leading parts using the `parentdir` filter, for example `{{filename|parentdir}}`

You can combine them with filters:

```
{{ component|title }}
```

You can use conditions:

```
{% if stats.translated_percent > 80 %}Well translated!{% endif %}
```

There is additional tag available for replacing characters:

```
{% replace component "-" " " %}
```

You can combine it with filters:

```
{% replace component|capfirst "-" " " %}
```

There are also additional filter to manipulate with filenames:

```
Directory of a file: {{ filename|dirname }}
File without extension: {{ filename|stripext }}
File in parent dir: {{ filename|parentdir }}
It can be used multiple times:  {{ filename|parentdir|parentdir }}
```

…and other Django template features.

2.7.6 Importing speed

Fetching VCS repository and importing translations to Weblate can be a lengthy process, depending on size of your translations. Here are some tips:

Optimize configuration

The default configuration is useful for testing and debugging Weblate, while for a production setup, you should do some adjustments. Many of them have quite a big impact on performance. Please check *Production setup* for more details, especially:

- Configure Celery for executing background tasks (see *Background tasks using Celery*)
- *Enable caching*
- *Use a powerful database engine*
- *Disable debug mode*

Check resource limits

If you are importing huge translations or repositories, you might be hit by resource limitations of your server.

- Check the amount of free memory, having translation files cached by the operating system will greatly improve performance.
- Disk operations might be bottleneck if there is a lot of strings to process—the disk is pushed by both Weblate and the database.
- Additional CPU cores might help improve performance of background tasks (see *Background tasks using Celery*).

Disable unneeded checks

Some quality checks can be quite expensive, and if not needed, can save you some time during import if omitted. See `CHECK_LIST` for info on configuration.

2.7.7 Automatic creation of components

In case your project has dozen of translation files (e.g. for different gettext domains, or parts of Android apps), you might want to import them automatically. This can either be achieved from the command line by using `import_project` or `import_json`, or by installing the *Descoperirea componentelor* addon.

To use the addon, you first need to create a component for one translation file (choose the one that is the least likely to be renamed or removed in future), and install the addon on this component.

For the management commands, you need to create a project which will contain all components and then run `import_project` or `import_json`.

Vezi și:

Management commands, *Descoperirea componentelor*

2.8 Language definitions

To present different translations properly, info about language name, text direction, plural definitions and language code is needed.

2.8.1 Parsing language codes

While parsing translations, Weblate attempts to map language code (usually the ISO 639-1 one) to any existing language object.

You can further adjust this mapping at project level by *Aliasuri lingvistice*.

If no exact match can be found, an attempt will be made to best fit it into an existing language. Following steps are tried:

- Case insensitive lookups.
- Normalizing underscores and dashes.
- Looking up built-in language aliases.
- Looking up by language name.
- Ignoring the default country code for a given language—choosing `cs` instead of `cs_CZ`.

Should that also fail, a new language definition will be created using the defaults (left to right text direction, one plural). The automatically created language with code `xx_XX` will be named as *xx_XX (generated)*. You might want to change this in the admin interface later, (see *Changing language definitions*) and report it to the issue tracker (see *Contributing to Weblate*), so that the proper definition can be added to the upcoming Weblate release.

Sugestie: In case you see something unwanted as a language, you might want to adjust *Filtru lingvistic* to ignore such file when parsing translations.

Vezi și:

Codul limbii, *Adding new translations*

2.8.2 Changing language definitions

You can change language definitions in the languages interface (`/languages/` URL).

While editing, make sure all fields are correct (especially plurals and text direction), otherwise translators will be unable to properly edit those translations.

2.8.3 Built-in language definitions

Definitions for about 600 languages are included in Weblate and the list is extended in every release. Whenever Weblate is upgraded (more specifically whenever **`weblate migrate`** is executed, see *Generic upgrade instructions*) the database of languages is updated to include all language definitions shipped in Weblate.

This feature can be disable using *`UPDATE_LANGUAGES`*. You can also enforce updating the database to match Weblate built-in data using *`setuplang`*.

Vezi și:

Extending built-in language definitions

2.8.4 Ambiguous language codes and macrolanguages

In many cases it is not a good idea to use macro language code for a translation. The typical problematic case might be Kurdish language, which might be written in Arabic or Latin script, depending on actual variant. To get correct behavior in Weblate, it is recommended to use individual language codes only and avoid macro languages.

Vezi și:

Macrolanguages definition, List of macrolanguages

2.8.5 Language definitions

Each language consists of following fields:

Codul limbii

Code identifying the language. Weblate prefers two letter codes as defined by ISO 639-1, but uses ISO 639-2 or ISO 639-3 codes for languages that do not have two letter code. It can also support extended codes as defined by BCP 47.

Vezi și:

Parsing language codes, *Adding new translations*

Numele limbii

Visible name of the language. The language names included in Weblate are also being localized depending on user interface language.

Direcția textului

Determines whether language is written right to left or left to right. This property is autodetected correctly for most of the languages.

Plural number

Number of plurals used in the language.

Formula plurală

Gettext compatible plural formula used to determine which plural form is used for given count.

Vezi și:

Pluralul, GNU gettext utilities: Plural forms, Language Plural Rules by the Unicode Consortium

2.8.6 Adding new translations

Schimbat în versiunea 2.18: In versions prior to 2.18 the behaviour of adding new translations was file format specific.

Weblate can automatically start new translation for all of the file formats.

Some formats expect to start with an empty file and only translated strings to be included (for example *Android string resources*), while others expect to have all keys present (for example *GNU gettext*). In some situations this really doesn't depend on the format, but rather on the framework you use to handle the translation (for example with *JSON files*).

When you specify *Șablon pentru traduceri noi* in *Component configuration*, Weblate will use this file to start new translations. Any exiting translations will be removed from the file when doing so.

When *Șablon pentru traduceri noi* is empty and the file format supports it, an empty file is created where new strings will be added once they are translated.

The *Stil de cod lingvistic* allows you to customize language code used in generated filenames:

Implicit, în funcție de formatul fișierului Dependent on file format, for most of them POSIX is used.

Stil POSIX care utilizează sublinierea ca separator Typically used by gettext and related tools, produces language codes like `pt_BR`.

Stil POSIX care utilizează sublinierea ca separator, inclusiv codul de țară POSIX style language code including the country code even when not necessary (for example `cs_CZ`).

Stilul BCP care utilizează cratima ca separator Typically used on web platforms, produces language codes like `pt-BR`.

Stil BCP care utilizează cratima ca separator, inclusiv codul de țară BCP style language code including the country code even when not necessary (for example `cs-CZ`).

Stil Android Only used in Android apps, produces language codes like `pt-rBR`.

stil Java Used by Java—mostly BCP with legacy codes for Chinese.

Additionally, any mappings defined in *Aliasuri lingvistice* are applied in reverse.

Notă: Weblate recognizes any of these when parsing translation files, the above settings only influences how new files are created.

Vezi și:

Codul limbii, *Parsing language codes*

2.9 Traducere continuă

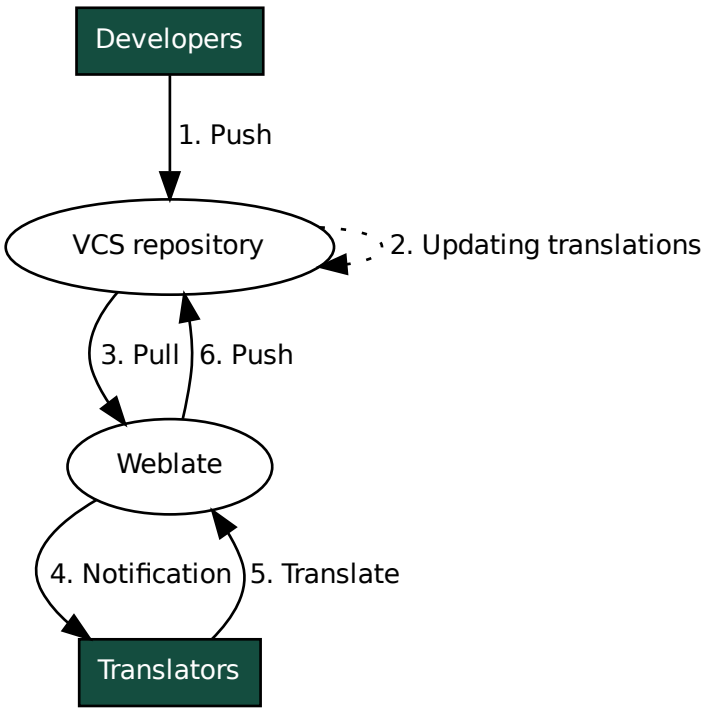
There is infrastructure in place so that your translation closely follows development. This way translators can work on translations the entire time, instead of working through huge amount of new text just prior to release.

Vezi și:

/devel/integration describes basic ways to integrate your development with Weblate.

This is the process:

1. Developers make changes and push them to the VCS repository.
2. Optionally the translation files are updated (this depends on the file format, see *Why does Weblate still show old translation strings when I've updated the template?*).
3. Weblate pulls changes from the VCS repository, see *Updating repositories*.
4. Once Weblate detects changes in translations, translators are notified based on their subscription settings.
5. Translators submit translations using the Weblate web interface, or upload offline changes.
6. Once the translators are finished, Weblate commits the changes to the local repository (see *Lazy commits*) and pushes them back if it has permissions to do so (see *Pushing changes from Weblate*).



2.9.1 Updating repositories

You should set up some way of updating backend repositories from their source.

- Use *Cârlige de notificare* to integrate with most of common code hosting services:
  - *Automatically receiving changes from GitHub*
  - *Automatically receiving changes from GitLab*
  - *Automatically receiving changes from Bitbucket*
  - *Automatically receiving changes from Pagure*
  - *Automatically receiving changes from Azure Repos*
- Manually trigger update either in the repository management or using *Weblate's REST API* or *Weblate Client*
- Enable `AUTO_UPDATE` to automatically update all components on your Weblate instance
- Execute `updategit` (with selection of project or `--all` to update all)

Whenever Weblate updates the repository, the post-update addons will be triggered, see *Add-ons*.

Avoiding merge conflicts

The merge conflicts from Weblate arise when same file was changed both in Weblate and outside it. There are two approaches to deal with that - avoid edits outside Weblate or integrate Weblate into your updating process, so that it flushes changes prior to updating the files outside Weblate.

The first approach is easy with monolingual files - you can add new strings within Weblate and leave whole editing of the files there. For bilingual files, there is usually some kind of message extraction process to generate translatable files from the source code. In some cases this can be split into two parts - one for the extraction generates template (for example gettext POT is generated using **xgettext**) and then further process merges it into actual translations (the gettext PO files are updated using **msgmerge**). You can perform the second step within Weblate and it will make sure that all pending changes are included prior to this operation.

The second approach can be achieved by using *Weblate's REST API* to force Weblate to push all pending changes and lock the translation while you are doing changes on your side.

The script for doing updates can look like this:

```
# Lock Weblate translation
wlc lock
# Push changes from Weblate to upstream repository
wlc push
# Pull changes from upstream repository to your local copy
git pull
# Update translation files, this example is for Django
./manage.py makemessages --keep-pot -a
git commit -m 'Locale updates' -- locale
# Push changes to upstream repository
git push
# Tell Weblate to pull changes (not needed if Weblate follows your repo
# automatically)
wlc pull
# Unlock translations
wlc unlock
```

If you have multiple components sharing same repository, you need to lock them all separately:

```
wlc lock foo/bar
wlc lock foo/baz
wlc lock foo/baj
```

Notă: The example uses *Weblate Client*, which needs configuration (API keys) to be able to control Weblate remotely. You can also achieve this using any HTTP client instead of wlc, e.g. curl, see *Weblate's REST API*.

Vezi și:

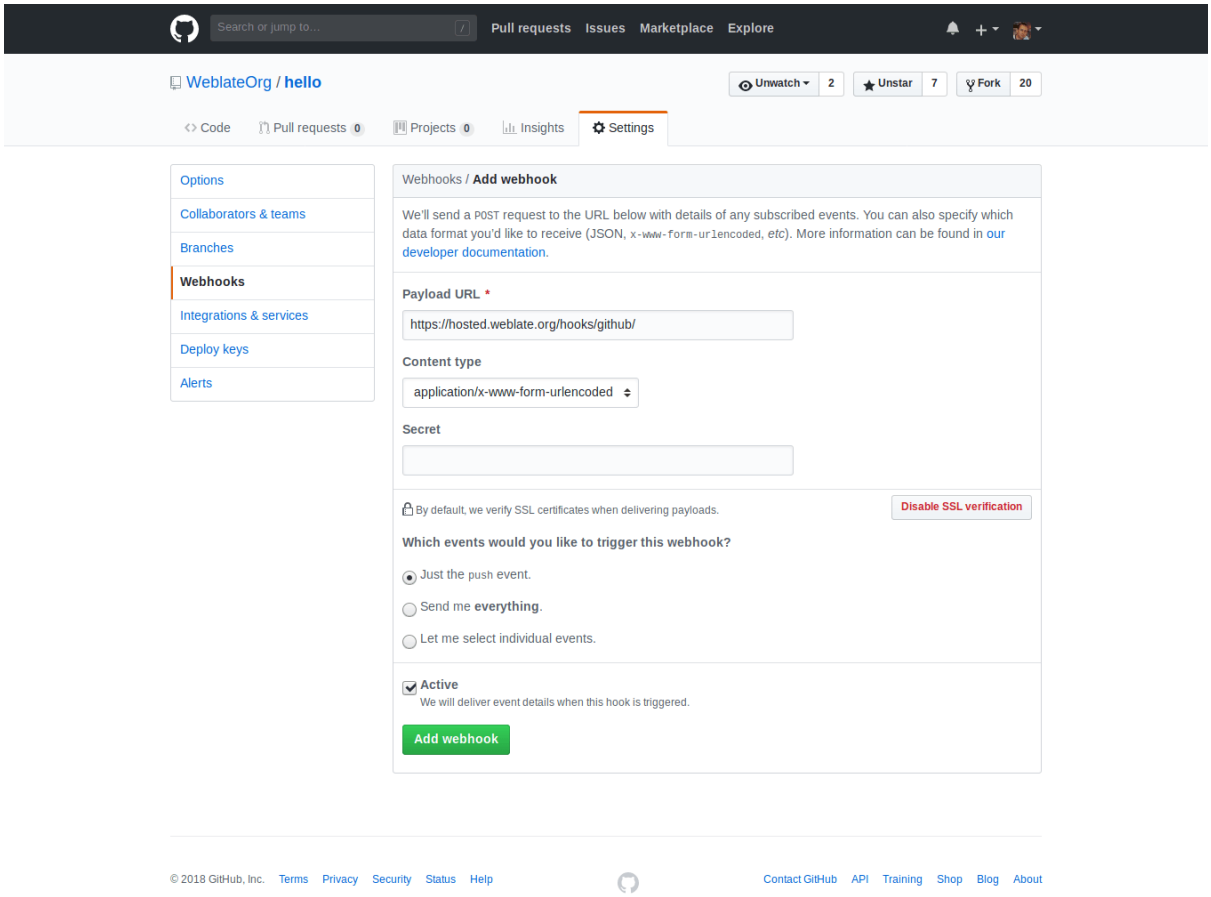
Weblate Client

Automatically receiving changes from GitHub

Weblate comes with native support for GitHub.

If you are using Hosted Weblate, the recommended approach is to install the Weblate app, that way you will get the correct setup without having to set much up. It can also be used for pushing changes back.

To receive notifications on every push to a GitHub repository, add the Weblate Webhook in the repository settings (*Webhooks*) as shown on the image below:



For the payload URL, append `/hooks/github/` to your Weblate URL, for example for the Hosted Weblate service, this is `https://hosted.weblate.org/hooks/github/`.

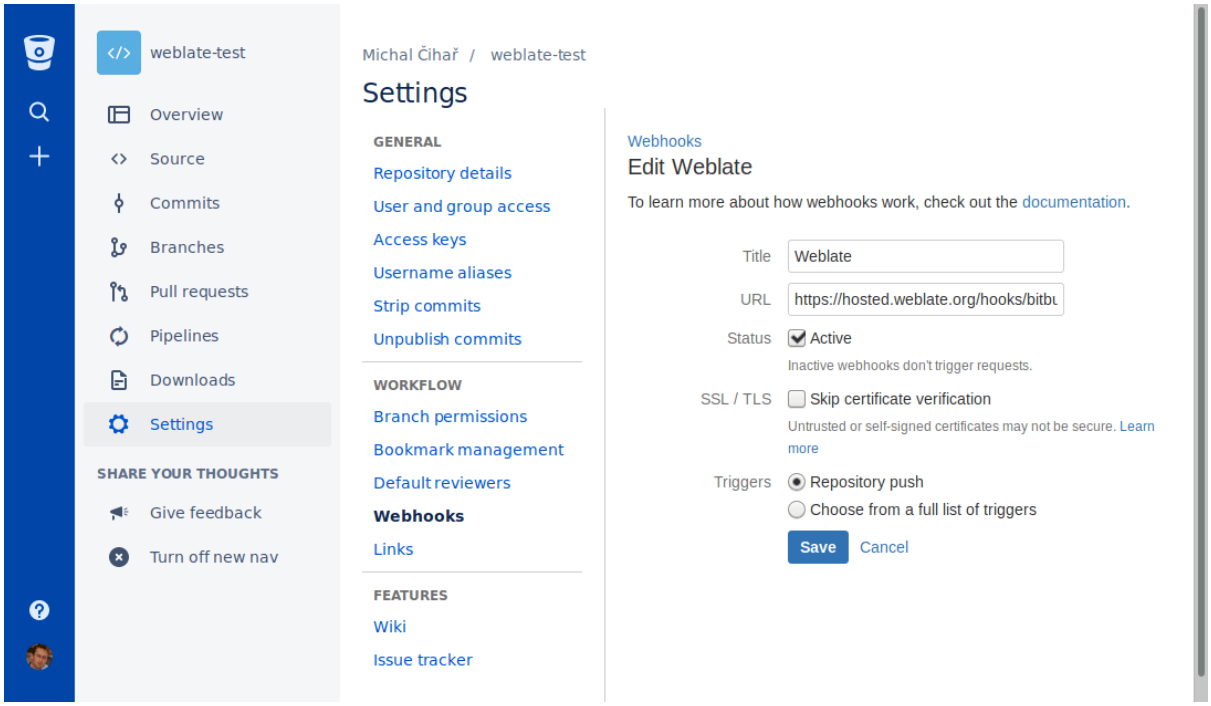
You can leave other values at default settings (Weblate can handle both content types and consumes just the *push* event).

Vezi și:

`POST /hooks/github/`, *Accessing repositories from Hosted Weblate*

Automatically receiving changes from Bitbucket

Weblate has support for Bitbucket webhooks, add a webhook which triggers upon repository push, with destination to `/hooks/bitbucket/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/bitbucket/`).



Vezi și:

`POST /hooks/bitbucket/`, *Accessing repositories from Hosted Weblate*

Automatically receiving changes from GitLab

Weblate has support for GitLab hooks, add a project webhook with destination to `/hooks/gitlab/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/gitlab/`).

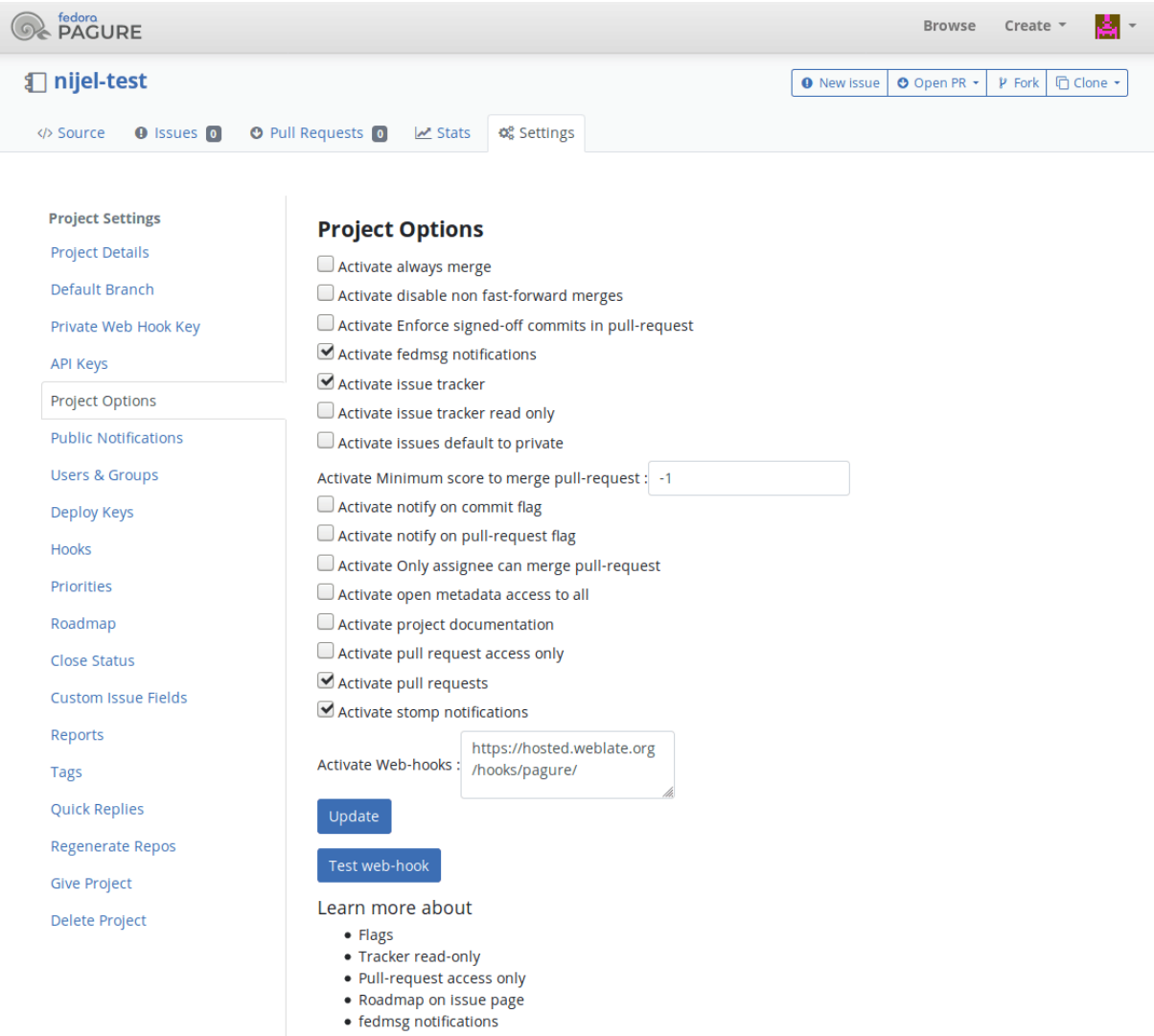
Vezi și:

`POST /hooks/gitlab/`, *Accessing repositories from Hosted Weblate*

Automatically receiving changes from Pagure

Nou în versiunea 3.3.

Weblate has support for Pagure hooks, add a webhook with destination to `/hooks/pagure/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/pagure/`). This can be done in *Activate Web-hooks* under *Project options*:



Vezi și:

`POST /hooks/pagure/`, *Accessing repositories from Hosted Weblate*

Automatically receiving changes from Azure Repos

Nou în versiunea 3.8.

Weblate has support for Azure Repos web hooks, add a webhook for *Code pushed* event with destination to `/hooks/azure/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/azure/`). This can be done in *Service hooks* under *Project settings*.

Vezi și:

Web hooks in Azure DevOps manual, `POST /hooks/azure/`, *Accessing repositories from Hosted Weblate*

Automatically receiving changes from Gitea Repos

Nou în versiunea 3.9.

Weblate has support for Gitea webhooks, add a *Gitea Webhook* for *Push events* event with destination to `/hooks/gitea/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/gitea/`). This can be done in *Webhooks* under repository *Settings*.

Vezi și:

Webhooks in Gitea manual, `POST /hooks/gitea/`, *Accessing repositories from Hosted Weblate*

Automatically receiving changes from Gitee Repos

Nou în versiunea 3.9.

Weblate has support for Gitee webhooks, add a *WebHook* for *Push* event with destination to `/hooks/gitee/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/gitee/`). This can be done in *WebHooks* under repository *Management*.

Vezi și:

Webhooks in Gitee manual, `POST /hooks/gitee/`, *Accessing repositories from Hosted Weblate*

Automatically updating repositories nightly

Weblate automatically fetches remote repositories nightly to improve performance when merging changes later. You can optionally turn this into doing nightly merges as well, by enabling `AUTO_UPDATE`.

2.9.2 Pushing changes from Weblate

Each translation component can have a push URL set up (see *URL de împingere a depozitului*), and in that case Weblate will be able to push change to the remote repository. Weblate can be also be configured to automatically push changes on every commit (this is default, see *Împingeți pe comitere*). If you do not want changes to be pushed automatically, you can do that manually under *Repository maintenance* or using API via `wlc push`.

The push options differ based on the *Integrarea controlului versiunilor* used, more details are found in that chapter.

In case you do not want direct pushes by Weblate, there is support for *GitHub*, *GitLab*, *Pagure* pull requests or *Gerrit* reviews, you can activate these by choosing *GitHub*, *GitLab*, *Gerrit* or *Pagure* as *Sistem de control al versiunilor* in *Component configuration*.

Overall, following options are available with Git, GitHub and GitLab:

Desired setup	*Sistem de control al versiunilor*	*URL de împingere a depozitului*	*Împingeți ramura*
No push	*Git*	*empty*	*empty*
Push directly	*Git*	SSH URL	*empty*
Push to separate branch	*Git*	SSH URL	Branch name
GitHub pull request from fork	*GitHub*	*empty*	*empty*
GitHub pull request from branch	*GitHub*	SSH URL[1]	Branch name
GitLab merge request from fork	*GitLab*	*empty*	*empty*
GitLab merge request from branch	*GitLab*	SSH URL[Pagina 65, 1]	Branch name
Pagure merge request from fork	*Pagure*	*empty*	*empty*
Pagure merge request from branch	*Pagure*	SSH URL[Pagina 65, 1]	Branch name

Notă: You can also enable automatic pushing of changes after Weblate commits, this can be done in *Împingeți pe comitere*.

Vezi și:

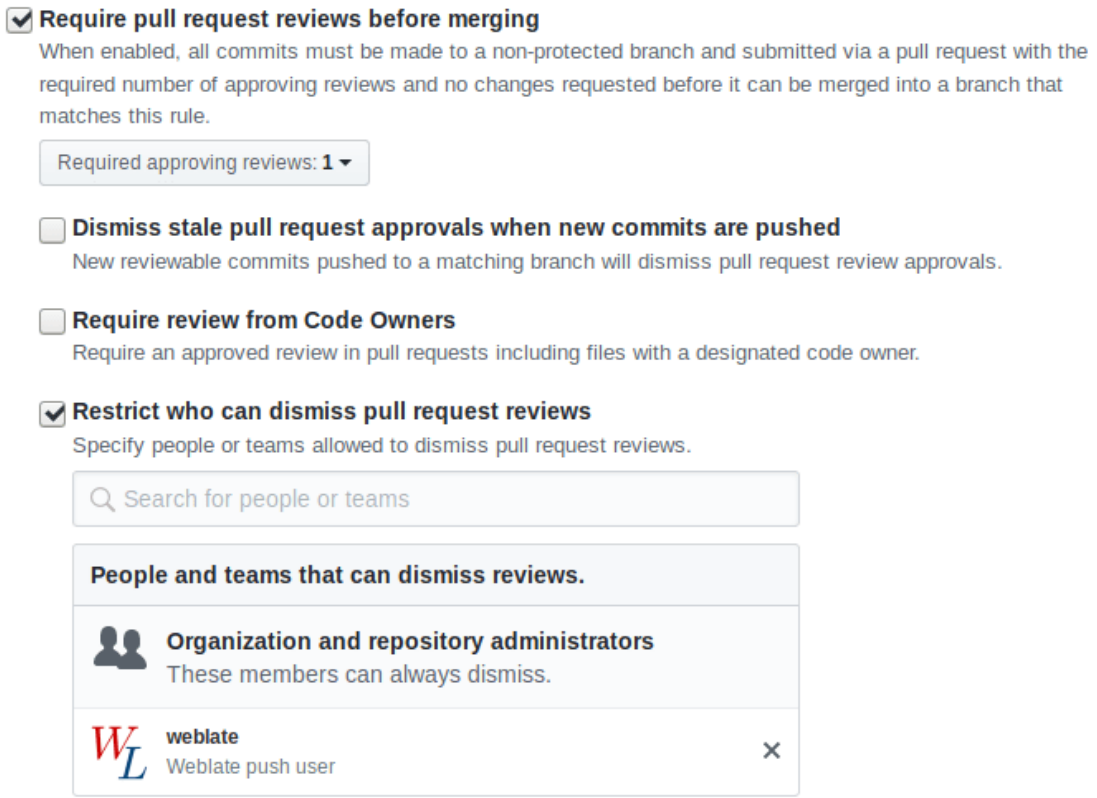
See *Accessing repositories* for setting up SSH keys, and *Lazy commits* for info about when Weblate decides to commit changes.

Protected branches

If you are using Weblate on protected branch, you can configure it to use pull requests and perform actual review on the translations (what might be problematic for languages you do not know). An alternative approach is to waive this limitation for the Weblate push user.

For example on GitHub this can be done in the repository configuration:

[1] Can be empty in case *Depozitul de cod sursă* supports pushing.



2.9.3 Merge or rebase

By default, Weblate merges the upstream repository into its own. This is the safest way in case you also access the underlying repository by other means. In case you don't need this, you can enable rebasing of changes on upstream, which will produce a history with fewer merge commits.

Notă: Rebasing can cause you trouble in case of complicated merges, so carefully consider whether or not you want to enable them.

2.9.4 Interacting with others

Weblate makes it easy to interact with others using its API.

Vezi și:

Weblate's REST API

2.9.5 Lazy commits

The behaviour of Weblate is to group commits from the same author into one commit if possible. This greatly reduces the number of commits, however you might need to explicitly tell it to do the commits in case you want to get the VCS repository in sync, e.g. for merge (this is by default allowed for the *Managers* group, see *List of privileges*).

The changes in this mode are committed once any of the following conditions are fulfilled:

- Somebody else changes an already changed string.
- A merge from upstream occurs.
- An explicit commit is requested.
- Change is older than period defined as *Vârsta modificărilor care urmează să fie confirmate* on *Component configuration*.

Sugestie: Commits are created for every component. So in case you have many components you will still see lot of commits. You might utilize *Comenzi Git Squash* addon in that case.

If you want to commit changes more frequently and without checking of age, you can schedule a regular task to perform a commit:

```
CELERY_BEAT_SCHEDULE = {
    # Unconditionally commit all changes every 2 minutes
    "commit": {
        "task": "weblate.trans.tasks.commit_pending",
        # Ommiting hours will honor per component settings,
        # otherwise components with no changes older than this
        # won't be committed
        "kwargs": {"hours": 0},
        # How frequently to execute the job in seconds
        "schedule": 120,
    }
}
```

2.9.6 Processing repository with scripts

The way to customize how Weblate interacts with the repository is *Add-ons*. Consult *Executing scripts from add-on* for info on how to execute external scripts through addons.

2.9.7 Keeping translations same across components

Once you have multiple translation components, you might want to ensure that the same strings have same translation. This can be achieved at several levels.

Translation propagation

With *Permiteți propagarea traducerii* enabled (what is the default, see *Component configuration*), all new translations are automatically done in all components with matching strings. Such translations are properly credited to currently translating user in all components.

Notă: The translation propagation requires the key to be match for monolingual translation formats, so keep that in mind when creating translation keys.

Consistency check

The *Inconsecvent* check fires whenever the strings are different. You can utilize this to review such differences manually and choose the right translation.

Traducere automată

Automatic translation based on different components can be way to synchronize the translations across components. You can either trigger it manually (see *Traducere automată*) or make it run automatically on repository update using addon (see *Traducere automată*).

2.10 Licensing translations

You can specify which license translations are contributed under. This is especially important to do if translations are open to the public, to stipulate what they can be used for.

You should specify *Component configuration* license info. You should avoid requiring a contributor license agreement, though it is possible.

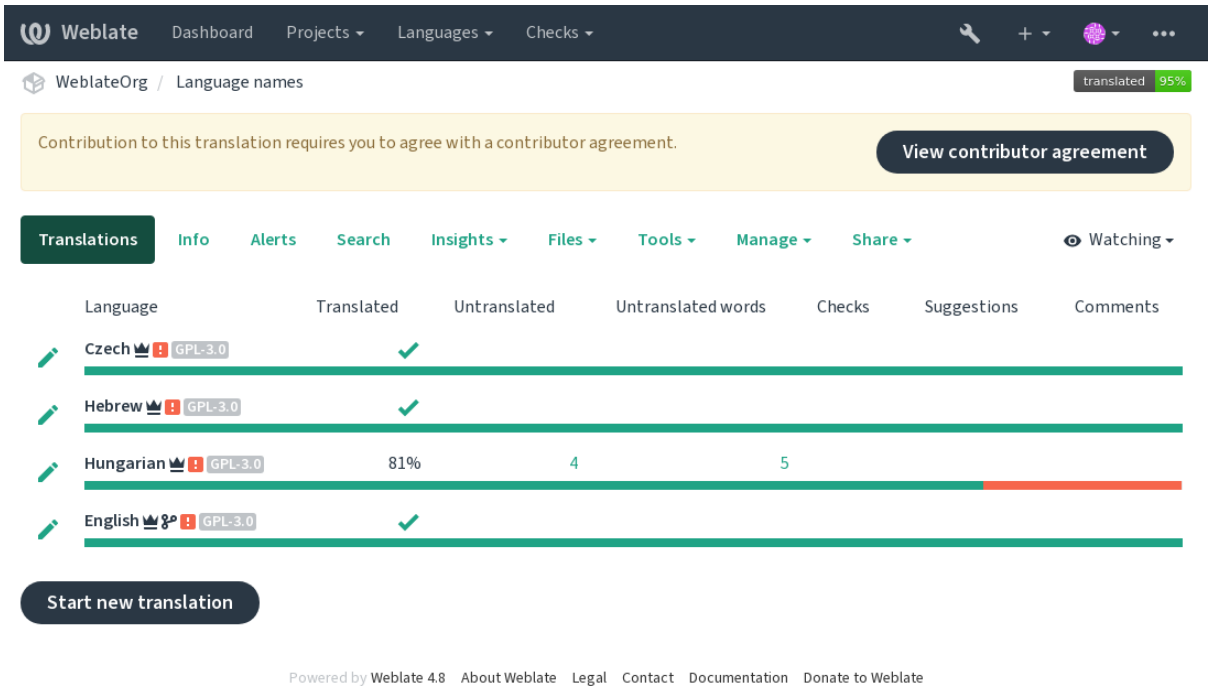
2.10.1 License info

Upon specifying license info (license name and URL), this info is shown in the translation info section of the respective *Component configuration*.

Usually this is best place to post licensing info if no explicit consent is required. If your project or translation is not libre you most probably need prior consent.

2.10.2 Acord de colaborare

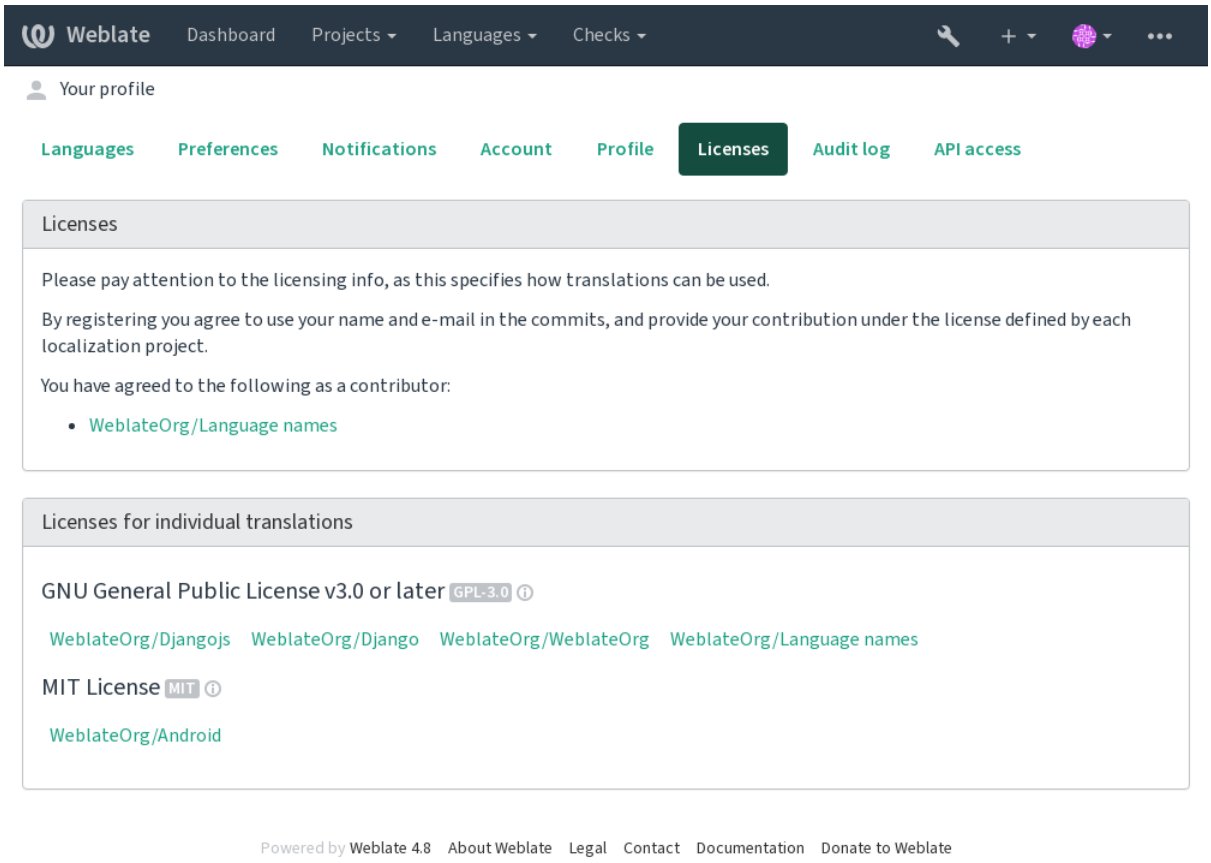
If you specify a contributor license agreement, only users who have agreed to it will be able to contribute. This is a clearly visible step when accessing the translation:



The entered text is formatted into paragraphs and external links can be included. HTML markup can not be used.

2.10.3 User licenses

Any user can review all translation licenses of all public projects on the instance from their profile:



2.11 Procesul de traducere

2.11.1 Votul prin sugestie

Everyone can add suggestions by default, to be accepted by signed in users. Suggestion voting can be used to make use of a string when more than one signed-in user agrees, by setting up the *Component configuration* with *Suggestion voting* to turn on voting, and *Autoaccept suggestions* to set a threshold for accepted suggestions (this includes a vote from the user making the suggestion if it is cast).

Notă: Once automatic acceptance is set up, normal users lose the privilege to directly save translations or accept suggestions. This can be overridden with the *Edit string when suggestions are enforced permission*.

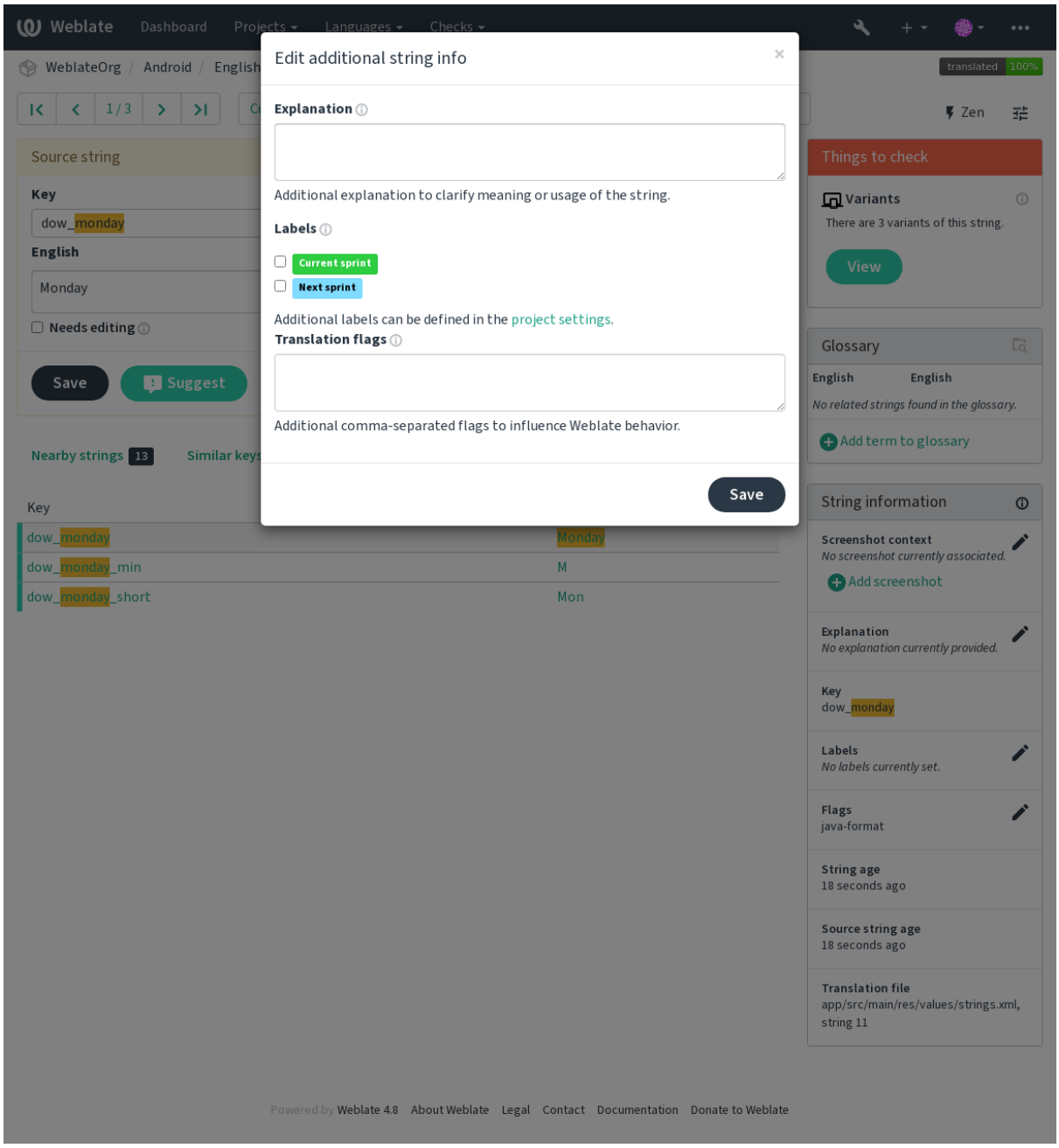
You can combine these with *access control* into one of the following setups:

- Users suggest and vote for suggestions and a limited group controls what is accepted. - Turn on voting. - Turn off automatic acceptance. - Don't let users save translations.
- Users suggest and vote for suggestions with automatic acceptance once the defined number of them agree. - Turn on voting. - Set the desired number of votes for automatic acceptance.

- Optional voting for suggestions. (Can optionally be used by users when they are unsure about a translation by making multiple suggestions.) - Only turn on voting.

2.11.2 Additional info on source strings

Enhance the translation process by adding additional info to the strings including explanations, string priorities, check flags and visual context. Some of that info may be extracted from the translation files and some may be added by editing the additional string info:



Access this directly from the translation interface by clicking the „Edit” icon next to *Screenshot context* or *Flags*.

Weblate Dashboard Projects Languages Checks

WeblateOrg / Django / Czech / Translate translated 96%

11 / 26 All strings Position and priority Zen

Translation

Explanation

Help text for automatic translation tool

English

Automatic translation via machine translation uses active machine translation engines to get the best possible translations and applies them in this project.

Czech

NBS

Automatický překlad prostřednictvím strojového překladu používá aktivní enginy strojového překladu pro získání nejlepších možných překladů a použije je na tento projekt.

Needs editing 169/1570 · 157

Save Suggest Skip

Nearby strings 26 Comments Automatic suggestions Other languages 4 History

Context	English	Czech
	Files	Soubory
	Automatic translation	Automatický překlad
	Add new translation string	Add new translation string
	Translation status	Stav překladu
	%(count)s word	%(count)s slovo
	Other components	Další součásti
	Translation file	Soubor s překladem
	Download	Stáhnout
	Browse all translation changes	Procházet všechny změny v překladu.
	Automatic translation takes existing translations in this project and applies them to the current component. It can be used to push translations to a different branch, to fix inconsistent translations or to translate a new component using translation memory.	Automatický překlad použije stávající překlady v projektu na tuto součást. Může být užitečný pro sloučení překladů z jiné větve, opravu nekonzistentních překladů nebo překlad nové součásti pomocí překladové paměti.
	Automatic translation via machine translation uses active machine translation engines to get the best possible translations and applies them in this project.	Automatický překlad prostřednictvím strojového překladu používá aktivní enginy strojového překladu pro získání nejlepších možných překladů a použije je na tento projekt.
	You can add new translation string here, it will automatically appear in all translations.	Zde můžete přidat nový řetězec k překladu, automaticky se objeví ve všech jazycích.
	The uploaded file will be merged with the current translation. In case you want to overwrite already translated strings, don't forget to enable it.	Nahraný soubor bude sloučen se stávajícími překlady. Pokud chcete přepsat již přeložené řetězce, nezapomeňte to povolit.
	The uploaded file will be merged with the current translation.	Nahraný soubor bude sloučen se stávajícími překlady.
	The fulltext search might not work properly as the fulltext index for this translation is not yet up to date.	Fulltextové vyhledávání nemusí fungovat správně, protože fulltextový index pro tento překlad ještě není plně zpracován.
	Review	Kontrola
	Review translations touched by other users.	Zkontrolovat překlady od ostatních uživatelů.
	Start review	Začít kontrolu
	Percent	Procenta
	Total	Celkem
	Failing check	Neúspěšných kontrol
	Last activity	Poslední aktivita
	Last change	Poslední změna
	Last author	Poslední autor
Question for a mathematics-based CAPTCHA, the %s is an arithmetic problem	What is %s?	Kolik to je?
	The string uses three dots (...) instead of an ellipsis character (…)	

Glossary

English	Czech	
machine translation	strojový překlad	WeblateOrg
project	projekt	WeblateOrg

Add term to glossary

String information

Screenshot context
No screenshot currently associated.
Add screenshot

Explanation
Help text for automatic translation tool

Labels
No labels currently set.

Flags
No flags currently set.

Source string location
weblate/templates/translation.html:212

String age
5 seconds ago

Source string age
6 seconds ago

Translation file
weblate/locale/cs/LC_MESSAGES/djangojo.po, string 11

Powered by Weblate 4.8 About Weblate Legal Contact Documentation Donate to Weblate

Strings prioritization

Nou în versiunea 2.0.

String priority can be changed to offer higher priority strings for translation earlier by using the `priority` flag.

Sugestie: This can be used to order the flow of translation in a logical manner.

Vezi și:

Quality checks

Steaguri de traducere

Nou în versiunea 2.4.

Schimbat în versiunea 3.3: Previously called *Quality checks flags*, it no longer configures only checks.

The default set of translation flags is determined by the translation *Component configuration* and the translation file. However, you might want to use it to customize this per source string.

Vezi și:

Quality checks, *Personalizarea comportamentului cu ajutorul steguleţelor*

Explicație

Schimbat în versiunea 4.1: In previous versions this has been called *Extra context*.

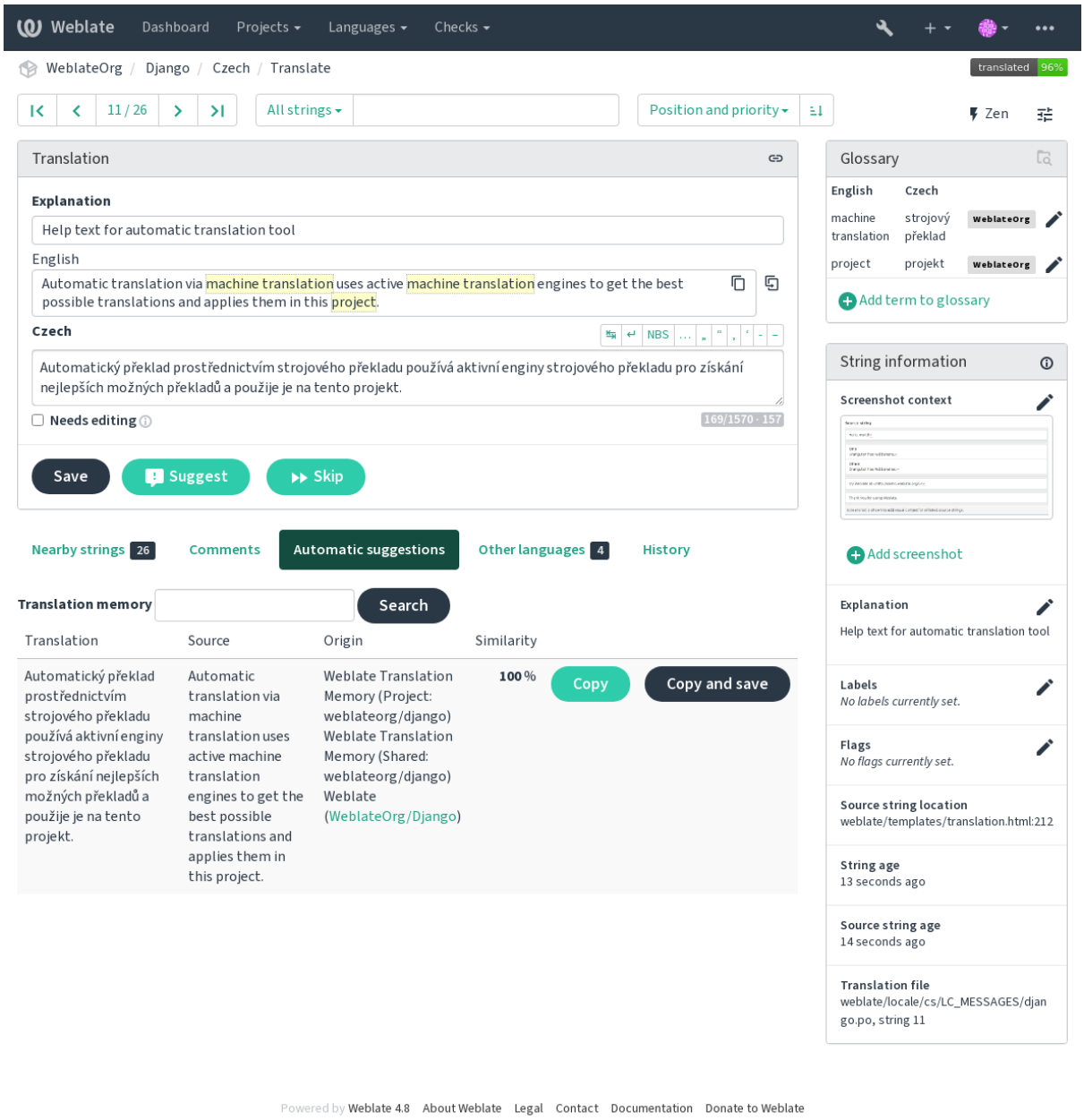
Use the explanation to clarify scope or usage of the translation. You can use Markdown to include links and other markup.

Visual context for strings

Nou în versiunea 2.9.

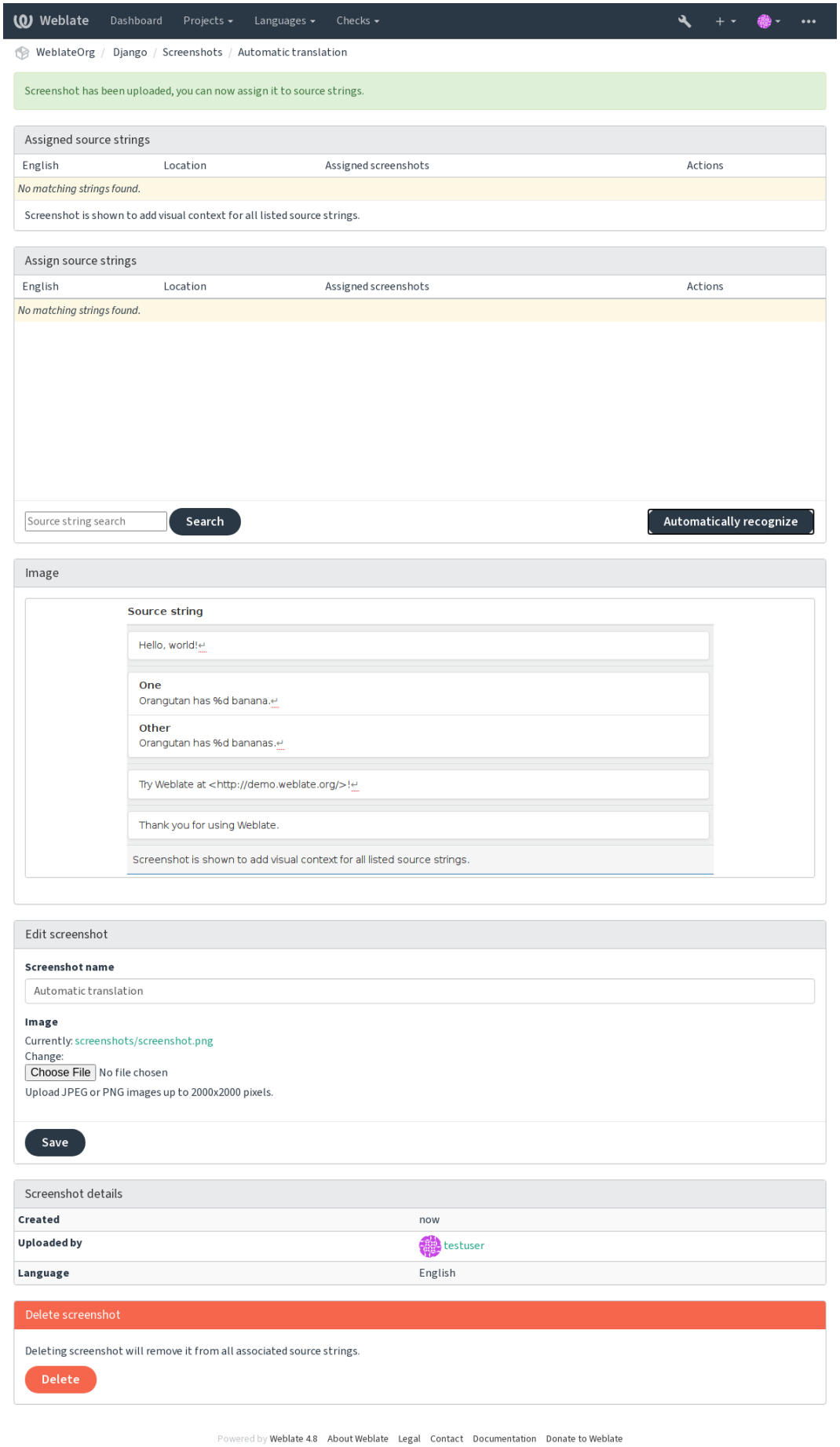
You can upload a screenshot showing a given source string in use within your program. This helps translators understand where it is used, and how it should be translated.

The uploaded screenshot is shown in the translation context sidebar:



In addition to *Additional info on source strings*, screenshots have a separate management interface under the *Tools* menu. Upload screenshots, assign them to source strings manually, or use optical character recognition to do so.

Once a screenshot is uploaded, this interface handles management and source string association:


Weblate
Dashboard
Projects
Languages
Checks
WeblateOrg / Django / Screenshots / Automatic translation
Screenshot has been uploaded, you can now assign it to source strings.
Assigned source strings
English
Location
Assigned screenshots
Actions
No matching strings found.
Screenshot is shown to add visual context for all listed source strings.
Assign source strings
English
Location
Assigned screenshots
Actions
No matching strings found.
Source string search
Search
Automatically recognize
Image
Source string
Hello, world!
One
Orangutan has %d banana.
Other
Orangutan has %d bananas.
Try Weblate at <http://demo.weblate.org/>!
Thank you for using Weblate.
Screenshot is shown to add visual context for all listed source strings.
Edit screenshot
Screenshot name
Automatic translation
Image
Currently: screenshots/screenshot.png
Change:
Choose File
No file chosen
Upload JPEG or PNG images up to 2000x2000 pixels.
Save
Screenshot details
Created
now
Uploaded by
testuser
Language
English
Delete screenshot
Deleting screenshot will remove it from all associated source strings.
Delete
Powered by Weblate 4.8
About Weblate
Legal
Contact
Documentation
Donate to Weblate

2.12 Verificări și reparații

2.12.1 Reparații automate personalizate

Puteți, de asemenea, să vă implementați propriile corecturi automate în plus față de cele standard și să le includeți în *AUTOFIX_LIST*.

Corecțiile automate sunt puternice, dar pot provoca, de asemenea, daune; aveți grijă când scrieți una.

De exemplu, următoarea corecție automată ar înlocui fiecare apariție a șirului `foo` într-o traducere cu `bar`:

```
#
# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#

from django.utils.translation import gettext_lazy as _

from weblate.trans.autofixes.base import AutoFix


class ReplaceFooWithBar(AutoFix):
    """Replace foo with bar."""

    name = _("Foobar")

    def fix_single_target(self, target, source, unit):
        if "foo" in target:
            return target.replace("foo", "bar"), True
        return target, False
```

Pentru a instala verificări personalizate, furnizați o cale complet calificată către clasa Python în *AUTOFIX_LIST*, a se vedea *Custom quality checks, addons and auto-fixes*.

2.12.2 Personalizarea comportamentului cu ajutorul steguleţelor

Puteți ajusta cu precizie comportamentul Weblate (în principal verificările) pentru fiecare șir sursă (în revizuirea șirurilor sursă, a se vedea *Additional info on source strings*) sau în *Component configuration* (*Steaguri de traducere*). Unele formate de fișiere permit, de asemenea, specificarea steguleţelor direct în format (a se vedea *Supported file formats*).

Steguleţele sunt separate prin virgulă, iar parametrii sunt separați prin două puncte. Puteți utiliza ghilimele pentru a include spații albe sau caractere speciale în șir. De exemplu:

```
placeholders:"special:value":"other value", regex:.*
```

Iată o listă a marcajelor acceptate în prezent:

rst-text Tratează un text ca un document reStructuredText, afectează *Traducere neschimbată*.

dos-eol Folosește markeri de sfârșit de linie DOS în loc de cei Unix (`\r\n` instead of `\n`).

read-only Șirul este doar pentru citire și nu trebuie editat în Weblate, a se vedea *Șiruri de caractere numai pentru citire*.

priority:N Prioritatea șirului. Șirurile cu prioritate mai mare sunt prezentate mai întâi pentru traducere. Prioritatea implicită este 100. Cu cât un șir are o prioritate mai mare, cu atât este oferit mai devreme pentru traducere.

max-length:N Limitează lungimea maximă a unui șir de caractere la N caractere, a se vedea *Lungimea maximă a traducerii*.

xml-text Tratează textul ca document XML, afectează *Sintaxa XML* și *Marcarea XML*.

font-family:NAME Definește font-family pentru verificările de redare, a se vedea *Gestionarea fonturilor*.

font-weight:WEIGHT Definiți font-weight pentru verificările de redare, a se vedea *Gestionarea fonturilor*.

font-size:SIZE Definiți dimensiunea fontului pentru verificările de redare, a se vedea *Gestionarea fonturilor*.

font-spacing:SPACING` Definiți spațierea literelor pentru verificările de redare, a se vedea *Gestionarea fonturilor*.

placeholders:NAME:NAME2:... Șiruri de caractere de poziție așteptate în traducere, a se vedea *Semne de poziție*.

replacements:FROM:TO:FROM2:TO2... Înlocuirile care trebuie efectuate la verificarea parametrilor textului rezultat (de exemplu, în *Dimensiunea maximă a traducerii* sau *Lungimea maximă a traducerii*). Cazul tipic de utilizare pentru acest lucru este de a extinde placeables pentru a se asigura că textul se potrivește chiar și cu valori lungi, de exemplu: `replacements:%s: "John Doe"`.

variants:SOURCE Marchează acest șir ca fiind o variantă de șir cu sursă corespunzătoare. A se vedea variants.

regex:REGEX Regular expression to match translation, see *Expresie regulată*.

forbidden Indică o traducere interzisă într-un glosar, a se vedea *Forbidden translations*.

strict-same Faceți ca „Traducerea neschimbată” să evite utilizarea listei negre a cuvintelor încorporate, a se vedea *Traducere neschimbată*.

check-glossary Enable the *Nu respectă glosarul* quality check.

angularjs-format Enable the *Șir de interpolare AngularJS* quality check.

c-format Enable the *Format C* quality check.

c-sharp-format Enable the *C# format* quality check.

es-format Enable the *Șabloane literale ECMAScript* quality check.

i18next-interpolation Enable the *Interpolare i18next* quality check.

java-format Enable the *format Java* quality check.

java-messageformat Enable the *Mesaj în format Java* quality check.

javascript-format Enable the *Format JavaScript* quality check.

lua-format Enable the *Formatul Lua* quality check.

object-pascal-format Enable the *Object Pascal format* quality check.

percent-placeholders Enable the *Procentaje de marcaj* quality check.

perl-format Enable the *Formatul Perl* quality check.

php-format Enable the *Formatul PHP* quality check.

python-brace-format Enable the *Formatul de bretea Python* quality check.

python-format Enable the *Format Python* quality check.

qt-format Enable the *Formatul Qt* quality check.

qt-plural-format Enable the *Formatul plural Qt* quality check.

ruby-format Enable the *Formatul Ruby* quality check.

scheme-format Enable the *Formatul schemei* quality check.

vue-format Enable the *Formatare Vue I18n* quality check.

md-text Treat text as a Markdown document. Enable *Legături Markdown*, *Referințe Markdown*, and *Sintaxa Markdown* quality checks.

safe-html Enable the *HTML nesigur* quality check.

url The string should consist of only a URL. Enable the *URL* quality check.

ignore-bbcode Skip the *Marcaj BBcode* quality check.

ignore-duplicate Skip the *Cuvinte consecutive duplicate* quality check.

ignore-check-glossary Skip the *Nu respectă glosarul* quality check.

ignore-double-space Skip the *Spațiu dublu* quality check.

ignore-angularjs-format Skip the *Șir de interpolare AngularJS* quality check.

ignore-c-format Skip the *Format C* quality check.

ignore-c-sharp-format Skip the *C# format* quality check.

ignore-es-format Skip the *Șabloane literale ECMAScript* quality check.

ignore-i18next-interpolation Skip the *Interpolare i18next* quality check.

ignore-java-format Skip the *format Java* quality check.

ignore-java-messageformat Skip the *Mesaj în format Java* quality check.

ignore-javascript-format Skip the *Format JavaScript* quality check.

ignore-lua-format Skip the *Formatul Lua* quality check.

ignore-object-pascal-format Skip the *Object Pascal format* quality check.

ignore-percent-placeholders Skip the *Procentaje de marcaj* quality check.

ignore-perl-format Skip the *Formatul Perl* quality check.

ignore-php-format Skip the *Formatul PHP* quality check.

ignore-python-brace-format Skip the *Formatul de bretea Python* quality check.

ignore-python-format Skip the *Format Python* quality check.

ignore-qt-format Skip the *Formatul Qt* quality check.

ignore-qt-plural-format Skip the *Formatul plural Qt* quality check.

ignore-ruby-format Skip the *Formatul Ruby* quality check.

ignore-scheme-format Skip the *Formatul schemei* quality check.

ignore-vue-format Skip the *Formatare Vue I18n* quality check.

ignore-translated Skip the *A fost tradus* quality check.

ignore-inconsistent Skip the *Inconsecvent* quality check.

ignore-kashida Skip the *Scrisoarea Kashida folosită* quality check.

ignore-md-link Skip the *Legături Markdown* quality check.

ignore-md-reflink Skip the *Referințe Markdown* quality check.

ignore-md-syntax Skip the *Sintaxa Markdown* quality check.

ignore-max-length Skip the *Lungimea maximă a traducerii* quality check.

ignore-max-size Skip the *Dimensiunea maximă a traducerii* quality check.

ignore-escaped-newline Skip the *Mismatched \n* quality check.

ignore-end-colon Skip the *Două puncte nepotrivite* quality check.

ignore-end-ellipsis Skip the *Elipsa nepotrivită* quality check.

ignore-end-exclamation Skip the *Semnul exclamării nepotrivit* quality check.

ignore-end-stop Skip the *Punct final nepotrivit* quality check.

ignore-end-question Skip the *Semn de întrebare nepotrivit* quality check.

ignore-end-semicolon Skip the *Punct și virgulă nepotrivit* quality check.

ignore-newline-count Skip the *Întreruperi de linie nepotrivite* quality check.

ignore-plurals Skip the *Lipsește pluralul* quality check.

ignore-placeholders Skip the *Semne de poziție* quality check.

ignore-punctuation-spacing Skip the *Spațierea punctuației* quality check.

ignore-regex Skip the *Expresie regulată* quality check.

ignore-same-plurals Skip the *Același plural* quality check.

ignore-begin-newline Skip the *Începutul liniei noi* quality check.

ignore-begin-space Skip the *Spații de pornire* quality check.

ignore-end-newline Skip the *Linie nouă de urmărire* quality check.

ignore-end-space Skip the *Spațiu de urmărire* quality check.

ignore-same Skip the *Traducere neschimbată* quality check.

ignore-safe-html Skip the *HTML nesigur* quality check.

ignore-url Skip the *URL* quality check.

ignore-xml-tags Skip the *Marcarea XML* quality check.

ignore-xml-invalid Skip the *Sintaxa XML* quality check.

ignore-zero-width-space Skip the *Spațiu cu lățime zero* quality check.

ignore-ellipsis Skip the *Elipsă* quality check.

ignore-long-untranslated Skip the *Lungă netradusă* quality check.

ignore-multiple-failures Skip the *Controale multiple eșuate* quality check.

ignore-unnamed-format Skip the *Variabile multiple fără nume* quality check.

ignore-optional-plural Skip the *Nepluralizat* quality check.

Notă: În general, regula se numește `ignore-*` pentru orice verificare, folosind identificatorul acesteia, astfel încât puteți folosi acest lucru chiar și pentru verificările dumneavoastră personalizate.

Aceste indicatori sunt înțeleși atât în setările *Component configuration*, cât și în setările pentru șirul sursă și în fișierul de traducere în sine (de exemplu, în GNU gettext).

2.12.3 Executarea controalelor

Nou în versiunea 3.11.

Puteți configura o listă de verificări care nu pot fi ignorate prin setarea *Controale forțate* în *Component configuration*. Fiecare verificare din listă nu poate fi ignorată în interfața cu utilizatorul, iar orice șir care nu trece de această verificare este marcat ca *Needs editing* (a se vedea *Translation states*).

2.12.4 Gestionarea fonturilor

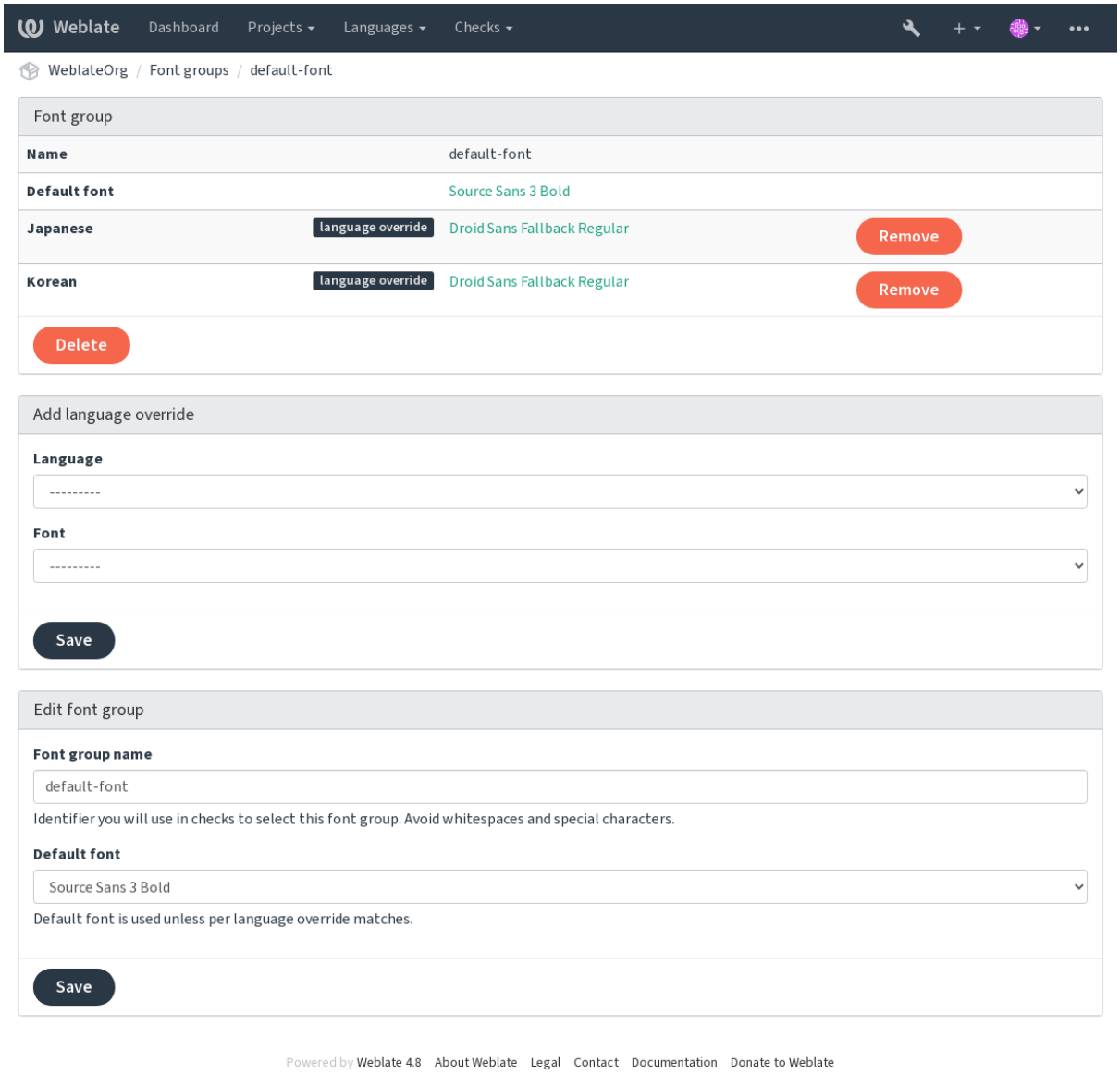
Nou în versiunea 3.7.

Sugestie: Fonturile încărcate în Weblate sunt folosite doar în scopul verificării *Dimensiunea maximă a traducerii*, ele nu au niciun efect în interfața cu utilizatorul Weblate.

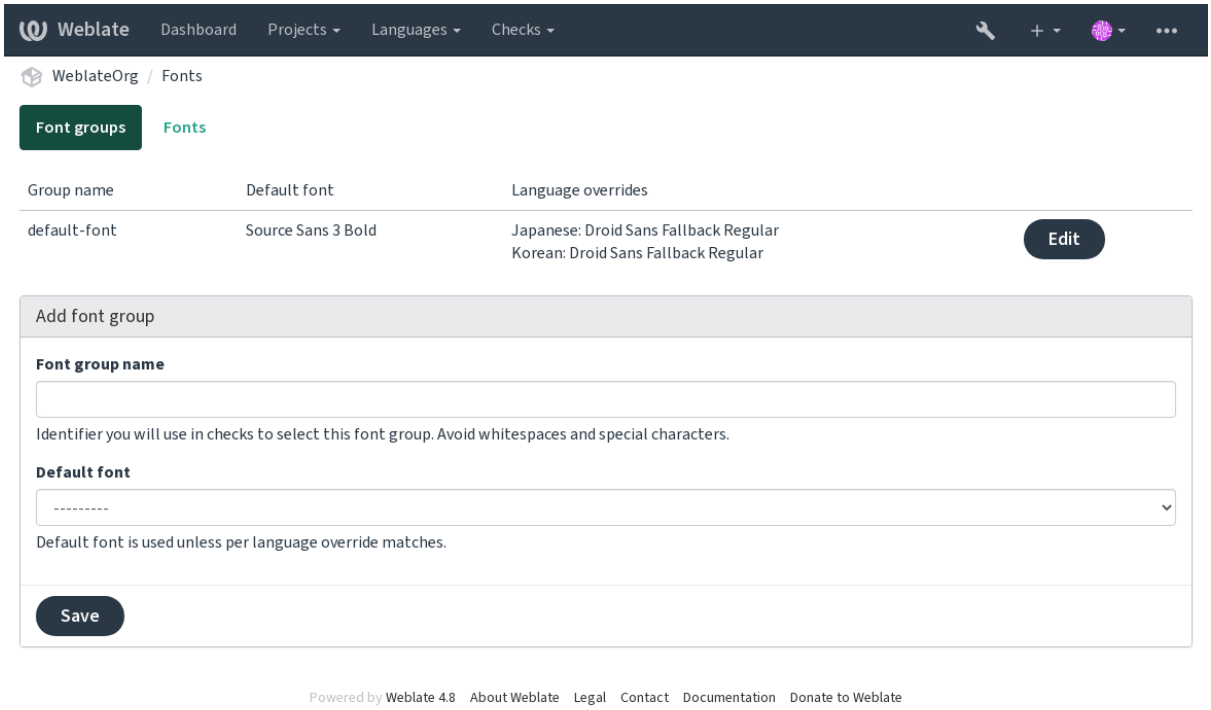
Verificarea *Dimensiunea maximă a traducerii* utilizată pentru a calcula dimensiunile textului redat are nevoie ca fontul să fie încărcat în Weblate și selectat cu ajutorul unui indicator de traducere (see *Personalizarea comportamentului cu ajutorul steguleţelor*).

Instrumentul de gestionare a fonturilor Weblate din *Fonts* din meniul *Manage* al proiectului de traducere oferă o interfață pentru încărcarea și gestionarea fonturilor. Se pot încărca fonturi TrueType sau OpenType, configura grupuri de fonturi și se pot utiliza cele din verificare.

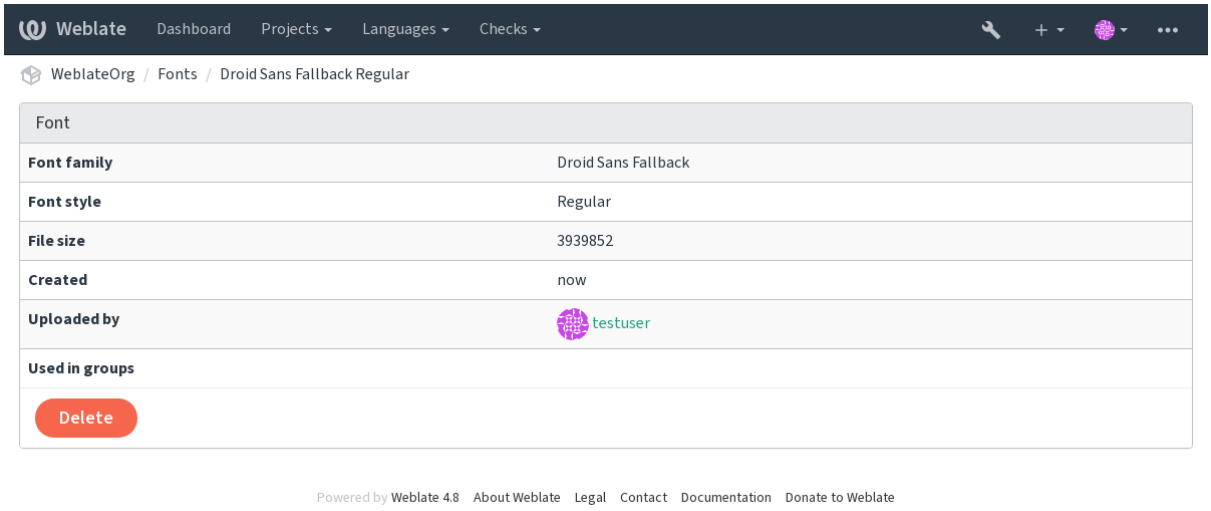
Grupurile de fonturi vă permit să definiți fonturi diferite pentru diferite limbi, ceea ce este de obicei necesar pentru limbile nelatine:



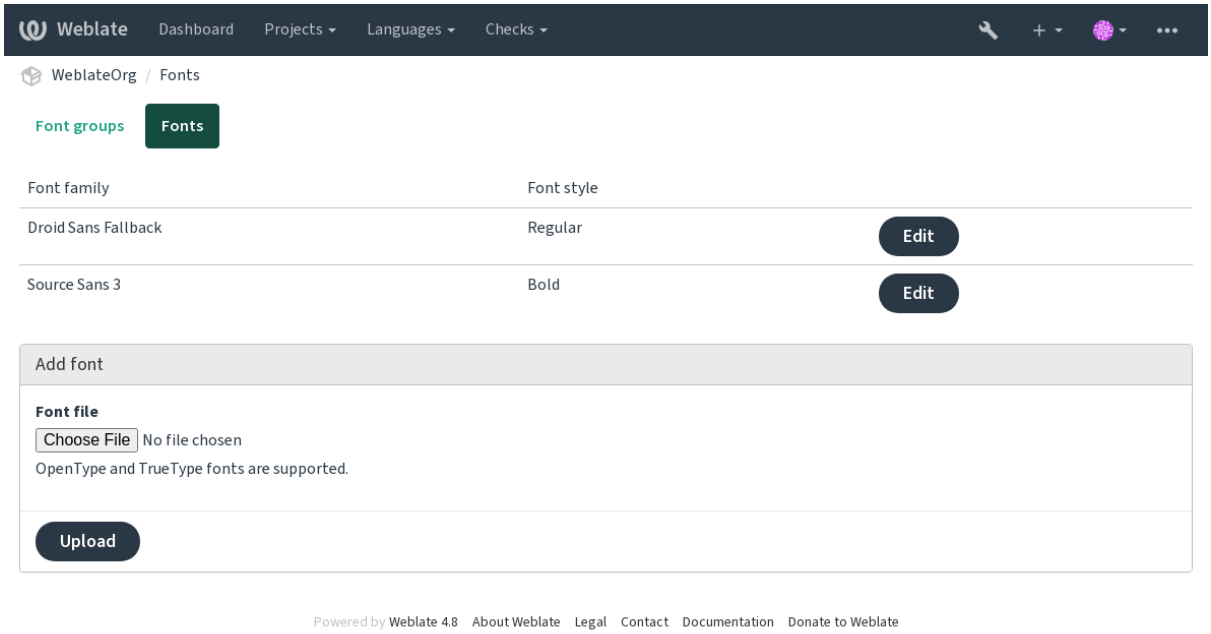
Grupurile de fonturi sunt identificate printr-un nume, care nu poate conține spații albe sau caractere speciale, astfel încât să poată fi utilizat cu ușurință în definiția verificării:



Familia de caractere și stilul sunt recunoscute automat după încărcarea lor:



Puteți avea mai multe fonturi încărcate în Weblate:



Pentru a utiliza fonturile pentru verificarea lungimii șirului de caractere, treceți-i indicatoarele corespunzătoare (a se vedea *Personalizarea comportamentului cu ajutorul steguleţelor*). Probabil că veți avea nevoie de următoarele:

`max-size:500` Define lățimea maximă.

`font-family:ubuntu` Definește grupul de fonturi care urmează să fie utilizat prin specificarea identificatorului acestuia.

`font-size:22` Define dimensiunea fontului.

2.12.5 Întocmirea propriilor cecuri

Există o gamă largă de verificări ale calității încorporate (a se vedea *Quality checks*), deși s-ar putea să nu acopere tot ceea ce doriți să verificați. Lista de verificări efectuate poate fi ajustată folosind `CHECK_LIST` și puteți adăuga și verificări personalizate.

1. Subclasa *weblate.checks.Check*
2. Setați câteva atribute.
3. Implementați fie `check` (dacă doriți să vă ocupați de plural în codul dumneavoastră), fie metoda `check_single` (care o face pentru dumneavoastră).

Câteva exemple:

Pentru a instala verificări personalizate, furnizați o cale complet calificată către clasa Python în `CHECK_LIST`, a se vedea *Custom quality checks, addons and auto-fixes*.

Verificarea faptului că textul traducerii nu conține „foo”

Aceasta este o verificare destul de simplă, care verifică doar dacă în traducere lipsește șirul „foo”.

```
#
# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
```

(continues on next page)

(continuare din pagina precedentă)

```
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Simple quality check example."""

from django.utils.translation import gettext_lazy as _

from weblate.checks.base import TargetCheck


class FooCheck(TargetCheck):

    # Used as identifier for check, should be unique
    # Has to be shorter than 50 characters
    check_id = "foo"

    # Short name used to display failing check
    name = _("Foo check")

    # Description for failing check
    description = _("Your translation is foo")

    # Real check code
    def check_single(self, source, target, unit):
        return "foo" in target
```

Verificarea faptului că pluralul din textul traducerii în limba cehă diferă

Verificați cu ajutorul informațiilor lingvistice pentru a verifica dacă cele două forme de plural în limba cehă nu sunt identice.

```
#
# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Quality check example for Czech plurals."""
```

(continues on next page)

(continuare din pagina precedentă)

```
from django.utils.translation import gettext_lazy as _

from weblate.checks.base import TargetCheck


class PluralCzechCheck(TargetCheck):

    # Used as identifier for check, should be unique
    # Has to be shorter than 50 characters
    check_id = "foo"

    # Short name used to display failing check
    name = _("Foo check")

    # Description for failing check
    description = _("Your translation is foo")

    # Real check code
    def check_target_unit(self, sources, targets, unit):
        if self.is_language(unit, ("cs",)):
            return targets[1] == targets[2]
        return False

    def check_single(self, source, target, unit):
        """We don't check target strings here."""
        return False
```

2.13 Traducere automată

Built-in support for several machine translation services and can be turned on by the administrator using *MT_SERVICES* for each one. They come subject to their terms of use, so ensure you are allowed to use them how you want.

The source language can be configured at *Project configuration*.

2.13.1 amaGama

Special installation of *tmserver* run by the authors of Virtaal.

Turn on this service by adding `weblate.machinery.tmserver.AmagamaTranslation` to *MT_SERVICES*.

Vezi și:

Installing amaGama, Amagama, amaGama Translation Memory

2.13.2 Apertium

A libre software machine translation platform providing translations to a limited set of languages.

The recommended way to use Apertium is to run your own Apertium-APy server.

Turn on this service by adding `weblate.machinery.apertium.ApertiumAPYTranslation` to *MT_SERVICES* and set *MT_APERTIUM_APY*.

Vezi și:

MT_APERTIUM_APY, Apertium website, Apertium APy documentation

2.13.3 AWS

Nou în versiunea 3.1.

Amazon Translate is a neural machine translation service for translating text to and from English across a breadth of supported languages.

1. Turn on this service by adding `weblate.machinery.aws.AWSTranslation` to *MT_SERVICES*.
2. Install the *boto3* module.
3. Configure Weblate.

Vezi și:

MT_AWS_REGION, *MT_AWS_ACCESS_KEY_ID*, *MT_AWS_SECRET_ACCESS_KEY* Amazon Translate Documentation

2.13.4 Baidu API machine translation

Nou în versiunea 3.2.

Machine translation service provided by Baidu.

This service uses an API and you need to obtain an ID and API key from Baidu to use it.

Turn on this service by adding `weblate.machinery.baidu.BaiduTranslation` to *MT_SERVICES* and set *MT_BAIDU_ID* and *MT_BAIDU_SECRET*.

Vezi și:

MT_BAIDU_ID, *MT_BAIDU_SECRET* Baidu Translate API

2.13.5 DeepL

Nou în versiunea 2.20.

DeepL is paid service providing good machine translation for a few languages. You need to purchase *DeepL API* subscription or you can use legacy *DeepL Pro (classic)* plan.

Turn on this service by adding `weblate.machinery.deepl.DeepLTranslation` to *MT_SERVICES* and set *MT_DEEPL_KEY*.

Sugestie: In case you have subscription for CAT tools, you are supposed to use „v1 API" instead of default „v2" used by Weblate (it is not really an API version in this case). In case you are on a free instead of a paid plan, you have to use `https://api-free.deepl.com/` instead of `https://api.deepl.com/` You can adjust both parameters by *MT_DEEPL_API_URL*.

Vezi și:

MT_DEEPL_KEY, *MT_DEEPL_API_URL*, DeepL website, DeepL pricing, DeepL API documentation

2.13.6 LibreTranslate

Nou în versiunea 4.7.1.

LibreTranslate is a free and open-source service for machine translations. The public instance requires an API key, but LibreTranslate can be self-hosted and there are several mirrors available to use the API for free.

Turn on this service by adding `weblate.machinery.libretranslate.LibreTranslateTranslation` to *MT_SERVICES* and set *MT_LIBRETRANSLATE_API_URL*. If your instance requires an API key, you must also set *MT_LIBRETRANSLATE_KEY*.

Vezi și:

MT_LIBRETRANSLATE_KEY, *MT_LIBRETRANSLATE_API_URL*, LibreTranslate website, LibreTranslate repository, LibreTranslate mirrors

2.13.7 Glosbe

Free dictionary and translation memory for almost every living language.

The API is gratis to use, but subject to the used data source license. There is a limit of calls that may be done from one IP in a set period of time, to prevent abuse.

Turn on this service by adding `weblate.machinery.glosbe.GlosbeTranslation` to *MT_SERVICES*.

Vezi și:

Glosbe website

2.13.8 Google Translate

Machine translation service provided by Google.

This service uses the Google Translation API, and you need to obtain an API key and turn on billing in the Google API console.

To turn on this service, add `weblate.machinery.google.GoogleTranslation` to *MT_SERVICES* and set *MT_GOOGLE_KEY*.

Vezi și:

MT_GOOGLE_KEY, Google translate documentation

2.13.9 Google Translate API V3 (Advanced)

Machine translation service provided by Google Cloud services.

This service differs from the former one in how it authenticates. To enable service, add `weblate.machinery.googlev3.GoogleV3Translation` to *MT_SERVICES* and set

- *MT_GOOGLE_CREDENTIALS*
- *MT_GOOGLE_PROJECT*

If *location* fails, you may also need to specify *MT_GOOGLE_LOCATION*.

Vezi și:

MT_GOOGLE_CREDENTIALS, *MT_GOOGLE_PROJECT*, *MT_GOOGLE_LOCATION* Google translate documentation

2.13.10 Microsoft Cognitive Services Translator

Nou în versiunea 2.10.

Machine translation service provided by Microsoft in Azure portal as a one of Cognitive Services.

Weblate implements Translator API V3.

To enable this service, add `weblate.machinery.microsoft.MicrosoftCognitiveTranslation` to *MT_SERVICES* and set *MT_MICROSOFT_COGNITIVE_KEY*.

Translator Text API V2

The key you use with Translator API V2 can be used with API 3.

Translator Text API V3

You need to register at Azure portal and use the key you obtain there. With new Azure keys, you also need to set *MT_MICROSOFT_REGION* to locale of your service.

Vezi și:

MT_MICROSOFT_COGNITIVE_KEY, *MT_MICROSOFT_REGION*, Cognitive Services - Text Translation API, Microsoft Azure Portal

2.13.11 Microsoft Terminology Service

Nou în versiunea 2.19.

The Microsoft Terminology Service API allows you to programmatically access the terminology, definitions and user interface (UI) strings available in the Language Portal through a web service.

Turn this service on by adding `weblate.machinery.microsoftterminology.MicrosoftTerminologyService` to *MT_SERVICES*.

Vezi și:

Microsoft Terminology Service API

2.13.12 ModernMT

Nou în versiunea 4.2.

Turn this service on by adding `weblate.machinery.modernmt.ModernMTTranslation` to *MT_SERVICES* and configure *MT_MODERNMT_KEY*.

Vezi și:

ModernMT API, *MT_MODERNMT_KEY*, *MT_MODERNMT_URL*

2.13.13 MyMemory

Huge translation memory with machine translation.

Free, anonymous usage is currently limited to 100 requests/day, or to 1000 requests/day when you provide a contact e-mail address in *MT_MYMEMORY_EMAIL*. You can also ask them for more.

Turn on this service by adding `weblate.machinery.mymemory.MyMemoryTranslation` to *MT_SERVICES* and set *MT_MYMEMORY_EMAIL*.

Vezi și:

MT_MYMEMORY_EMAIL, *MT_MYMEMORY_USER*, *MT_MYMEMORY_KEY*, MyMemory website

2.13.14 NetEase Sight API machine translation

Nou în versiunea 3.3.

Machine translation service provided by NetEase.

This service uses an API, and you need to obtain key and secret from NetEase.

Turn on this service by adding `weblate.machinery.youdao.NeteaseSightTranslation` to *MT_SERVICES* and set *MT_NETEASE_KEY* and *MT_NETEASE_SECRET*.

Vezi și:

MT_NETEASE_KEY, *MT_NETEASE_SECRET* NetEase Sight Translation Platform

2.13.15 tmserver

You can run your own translation memory server by using the one bundled with Translate-toolkit and let Weblate talk to it. You can also use it with an amaGama server, which is an enhanced version of tmserver.

1. First you will want to import some data to the translation memory:

2. Turn on this service by adding `weblate.machinery.tmserver.TMServerTranslation` to *MT_SERVICES*.

```
build_tmdb -d /var/lib/tm/db -s en -t cs locale/cs/LC_MESSAGES/django.po
build_tmdb -d /var/lib/tm/db -s en -t de locale/de/LC_MESSAGES/django.po
build_tmdb -d /var/lib/tm/db -s en -t fr locale/fr/LC_MESSAGES/django.po
```

3. Start tmserver to listen to your requests:

```
tmserver -d /var/lib/tm/db
```

4. Configure Weblate to talk to it:

```
MT_TMSERVER = "http://localhost:8888/tmserver/"
```

Vezi și:

MT_TMSERVER, tmserver Installing amaGama, Amagama, Amagama Translation Memory

2.13.16 Yandex Translate

Machine translation service provided by Yandex.

This service uses a Translation API, and you need to obtain an API key from Yandex.

Turn on this service by adding `weblate.machinery.yandex.YandexTranslation` to *MT_SERVICES*, and set *MT_YANDEX_KEY*.

Vezi și:

MT_YANDEX_KEY, Yandex Translate API, Powered by Yandex.Translate

2.13.17 Youdao Zhiyun API machine translation

Nou în versiunea 3.2.

Machine translation service provided by Youdao.

This service uses an API, and you need to obtain an ID and an API key from Youdao.

Turn on this service by adding `weblate.machinery.youdao.YoudaoTranslation` to *MT_SERVICES* and set *MT_YOUDAO_ID* and *MT_YOUDAO_SECRET*.

Vezi și:

MT_YOUDAO_ID, *MT_YOUDAO_SECRET* Youdao Zhiyun Natural Language Translation Service

2.13.18 Weblate

Weblate can be the source of machine translations as well. It is based on the Woosh fulltext engine, and provides both exact and inexact matches.

Turn on these services by adding `weblate.machinery.weblatetm.WeblateTranslation` to *MT_SERVICES*.

2.13.19 Weblate Translation Memory

Nou în versiunea 2.20.

The *Memorie de traducere* can be used as a source for machine translation suggestions as well.

Turn on these services by adding `weblate.memory.machine.WeblateMemory` to the *MT_SERVICES*. This service is turned on by default.

2.13.20 SAP Translation Hub

Machine translation service provided by SAP.

You need to have a SAP account (and the SAP Translation Hub enabled in the SAP Cloud Platform) to use this service.

Turn on this service by adding `weblate.machinery.saptranslationhub.SAPTranslationHub` to *MT_SERVICES* and set the appropriate access to either the sandbox or the production API.

Notă: To access the Sandbox API, you need to set *MT_SAP_BASE_URL* and *MT_SAP_SANDBOX_APIKEY*.

To access the productive API, you need to set *MT_SAP_BASE_URL*, *MT_SAP_USERNAME* and *MT_SAP_PASSWORD*.

Vezi şi:

MT_SAP_BASE_URL, *MT_SAP_SANDBOX_APIKEY*, *MT_SAP_USERNAME*, *MT_SAP_PASSWORD*, *MT_SAP_USE_MT* SAP Translation Hub API

2.13.21 Custom machine translation

You can also implement your own machine translation services using a few lines of Python code. This example implements machine translation in a fixed list of languages using `dictionary` Python module:

```
#
# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Machine translation example."""

import dictionary

from weblate.machinery.base import MachineTranslation


class SampleTranslation(MachineTranslation):
    """Sample machine translation interface."""

    name = "Sample"

    def download_languages(self):
        """Return list of languages your machine translation supports."""
        return {"cs"}

    def download_translations(
        self,
        source,
        language,
        text: str,
        unit,
        user,
        search: bool,
        threshold: int = 75,
    ):
        """Return tuple with translations."""
        for t in dictionary.translate(text):
            yield {"text": t, "quality": 100, "service": self.name, "source": text}
```

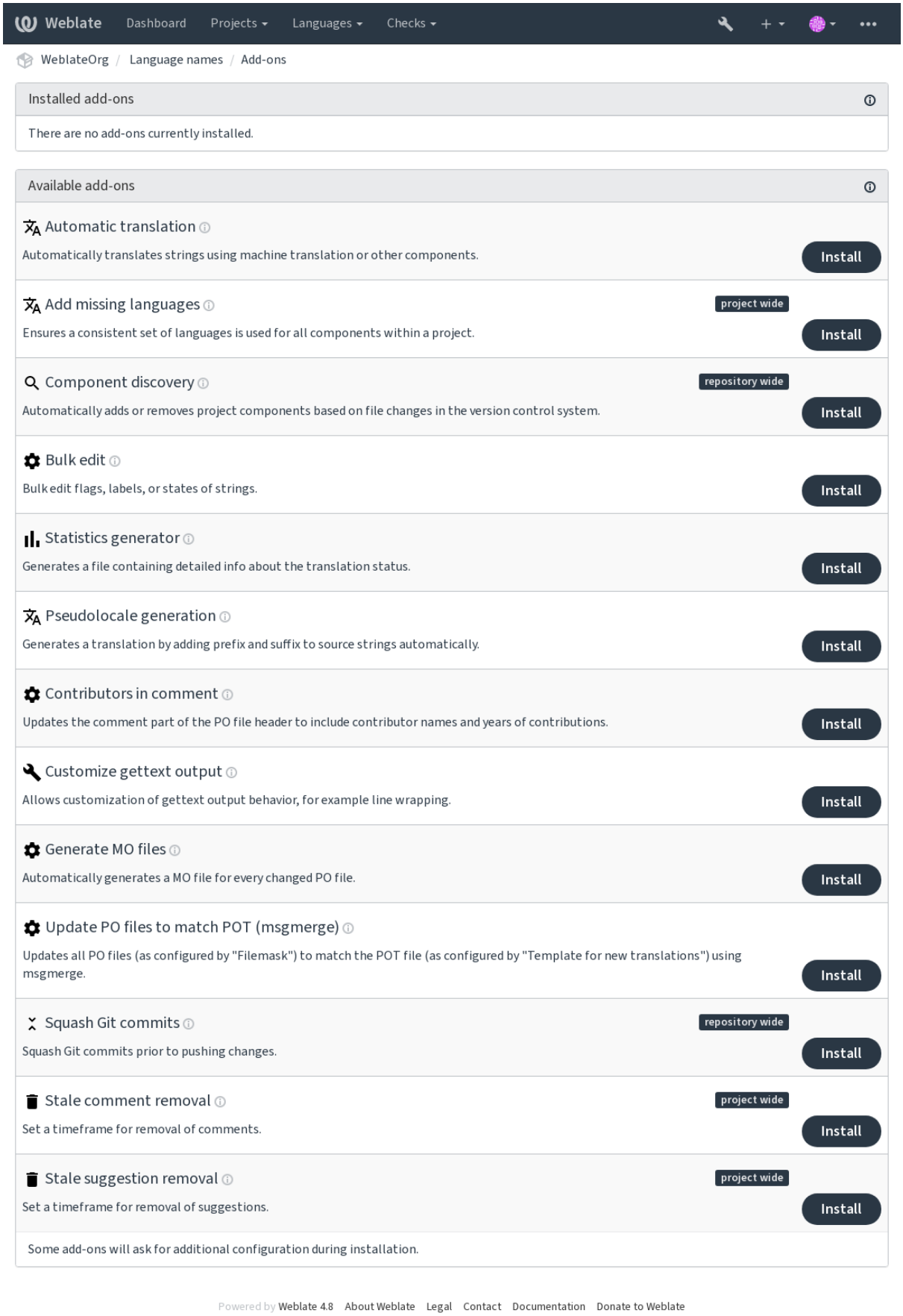
You can list your own class in *MT_SERVICES* and Weblate will start using that.

2.14 Add-ons

Nou în versiunea 2.19.

Add-ons provide ways to customize and automate the translation workflow. Admins can add and manage add-ons from the *Manage* ↓ *Add-ons* menu of each respective translation component.

Sugestie: You can also configure add-ons using *API*, `DEFAULT_ADDONS`, or `install_addon`.


Weblate
Dashboard
Projects
Languages
Checks
WeblateOrg / Language names / Add-ons
Installed add-ons
There are no add-ons currently installed.
Available add-ons
Automatic translation
Automatically translates strings using machine translation or other components.
Install
Add missing languages
project wide
Ensures a consistent set of languages is used for all components within a project.
Install
Component discovery
repository wide
Automatically adds or removes project components based on file changes in the version control system.
Install
Bulk edit
Bulk edit flags, labels, or states of strings.
Install
Statistics generator
Generates a file containing detailed info about the translation status.
Install
Pseudolocale generation
Generates a translation by adding prefix and suffix to source strings automatically.
Install
Contributors in comment
Updates the comment part of the PO file header to include contributor names and years of contributions.
Install
Customize gettext output
Allows customization of gettext output behavior, for example line wrapping.
Install
Generate MO files
Automatically generates a MO file for every changed PO file.
Install
Update PO files to match POT (msgmerge)
Updates all PO files (as configured by "Filemask") to match the POT file (as configured by "Template for new translations") using msgmerge.
Install
Squash Git commits
repository wide
Squash Git commits prior to pushing changes.
Install
Stale comment removal
project wide
Set a timeframe for removal of comments.
Install
Stale suggestion removal
project wide
Set a timeframe for removal of suggestions.
Install
Some add-ons will ask for additional configuration during installation.
Powered by Weblate 4.8 About Weblate Legal Contact Documentation Donate to Weblate

2.14.1 Built-in add-ons

Traducere automată

Nou în versiunea 3.9.

Add-on ID `weblate.autotranslate.autotranslate`

Configurație

`mode`	Modul de traducere automată	Available choices: `suggest` – Add as suggestion `translate` – Add as translation `fuzzy` – Add as needing edit
`filter_type`	Filtru de căutare	Available choices: `all` – Toate șirurile de caractere `nottranslated` – Not translated strings `todo` – Șiruri de caractere care au nevoie de o acțiune `fuzzy` – Strings marked for edit `check:inconsistent` – Failed check: Inconsistent
`auto_source`	Sursă de traducere automată	Available choices: `others` – Other translation components `mt` – Machine translation
`component`	Componente	Activați contribuția la memoria de traducere partajată pentru proiect pentru a avea acces la componente suplimentare.
`engines`	Motoare de traducere automată	
`threshold`	Prag de punctaj	

Triggers component update, daily

Traduce automat șirurile de caractere cu ajutorul traducerii automate sau al altor componente.

It is triggered:

- When new strings appear in a component.
- O dată pe lună pentru fiecare componentă, acest lucru poate fi configurat folosind *BACKGROUND_TASKS*.

Vezi și:

Traducere automată, *Keeping translations same across components*

CDN de localizare JavaScript

Nou în versiunea 4.2.

Add-on ID `weblate.cdn.cdnjs`

Configurație

`threshold`	Pragul de traducere	Pragul de includere a traducerilor.
`css_selector`	Selector CSS	Selector CSS pentru detectarea elementelor localizabile.
`cookie_name`	Numele cookie-ului de limbă	Numele modulului cookie care stochează preferințele lingvistice.
`files`	Extrageți șiruri de caractere din fișiere HTML	Listă de nume de fișiere din depozitul curent sau de adrese URL de la distanță care trebuie analizate pentru șiruri de caractere traductibile.

Triggers daily, repository post-commit, repository post-update

Publică traducerile în rețeaua de livrare de conținut pentru a fi utilizate în localizarea JavaScript sau HTML.

Poate fi utilizat pentru a localiza pagini HTML statice sau pentru a încărca localizarea în codul JavaScript.

Generează un URL unic pentru componenta dvs. pe care îl puteți include în paginile HTML pentru a le localiza. Consultați weblate-cdn pentru mai multe detalii.

Vezi și:

cdn-addon-config, weblate-cdn, cdn-addon-extract, cdn-addon-html

Eliminați șirurile de caractere goale

Nou în versiunea 4.4.

Add-on ID `weblate.cleanup.blank`

Configurație *This add-on has no configuration.*

Triggers repository post-commit, repository post-update

Elimină șirurile de caractere fără traducere din fișierele de traducere.

Utilizați această opțiune pentru a nu avea șiruri goale în fișierele de traducere (de exemplu, dacă biblioteca de localizare le afișează ca fiind lipsă în loc să revină la șirul sursă).

Vezi și:

Does Weblate update translation files besides translations?

Curățați fișierele de traducere

Add-on ID `weblate.cleanup.generic`

Configurație *This add-on has no configuration.*

Triggers repository pre-commit, repository post-update

Actualizarea tuturor fișierelor de traducere pentru a se potrivi cu fișierul de bază monolingv. Pentru majoritatea formatelor de fișiere, acest lucru înseamnă eliminarea cheilor de traducere vechi care nu mai sunt prezente în fișierul de bază.

Vezi și:

Does Weblate update translation files besides translations?

Add missing languages

Add-on ID `weblate.consistency.languages`

Configurație *This add-on has no configuration.*

Triggers daily, repository post-add

Ensures a consistent set of languages is used for all components within a project.

Limbile lipsă sunt verificate o dată la 24 de ore și atunci când sunt adăugate noi limbi în Weblate.

Unlike most others, this add-on affects the whole project.

Sugestie: Traduceți automat șirurile nou adăugate cu *Traducere automată*.

Descoperirea componentelor

Add-on ID `weblate.discovery.discovery`

Configurație

`match`	Expresie regulată pentru a potrivi fișierele de traducere cu	
`file_format`	Formatul fișierului	
`name_template`	Personalizați numele componentei	
`base_file_template`	Definiți numele fișierului de bază monolingv	Se lasă gol pentru fișierele de traducere bilingve.
`new_base_template`	Definiți fișierul de bază pentru noile traduceri	Numele fișierului utilizat pentru crearea de noi traduceri. Pentru gettext, alegeți fișierul .pot.
`language_regex`	Filtru lingvistic	Expresie regulată pentru filtrarea fișierelor de traducere la scanarea pentru filemask.
`copy_addons`	Clonarea addon-urilor din componenta principală în cele nou create	
`remove`	Eliminarea componentelor pentru fișiere inexistente	
`confirm`	Confirm că meciurile de mai sus par corecte	

Triggers repository post-update

Adaugă sau elimină automat componente ale proiectului pe baza modificărilor de fișiere din sistemul de control al versiunilor.

Se declanșează de fiecare dată când VCS este actualizat și, în rest, este similară comenzii de gestionare `import_project`. În acest fel, puteți urmări mai multe componente de traducere în cadrul unui VCS.

The matching is done using regular expressions enabling complex configuration, but some knowledge is required to do so. Some examples for common use cases can be found in the add-on help section.

După ce apăsați pe *Save*, va fi prezentată o previzualizare a componentelor corespunzătoare, de unde puteți verifica dacă configurația se potrivește cu nevoile dumneavoastră:

Weblate Dashboard Projects Languages Checks

WeblateOrg / Language names / Add-ons / Component discovery

Configure add-on

- Please review and confirm the matched components.

Component	Matched files
The following components would be created	
Djangojs	`weblate/locale/cs/LC_MESSAGES/djangojs.po` (cs) `weblate/locale/he/LC_MESSAGES/djangojs.po` (he) `weblate/locale/hu/LC_MESSAGES/djangojs.po` (hu)
Django	`weblate/locale/he/LC_MESSAGES/django.po` (he) `weblate/locale/hu/LC_MESSAGES/django.po` (hu) `weblate/locale/cs/LC_MESSAGES/django.po` (cs)

☐ **I confirm the above matches look correct**

Regular expression to match translation files against

weblate/locale/(?P<language>[^/]*)/LC_MESSAGES/(?P<component>[^/]*)\.po

File format

gettext PO file

Customize the component name

{{ component|title }}

Define the monolingual base filename

Leave empty for bilingual translation files.

Define the base file for new translations

weblate/locale/{{ component }}.pot

Filename of file used for creating new translations. For gettext choose .pot file.

Language filter

^(cs|he|hu)$

Regular expression to filter translation files against when scanning for filemask.

☑ **Clone addons from the main component to the newly created ones**

☐ **Remove components for inexistant files**

The regular expression to match translation files has to contain two named groups to match component and language, some examples:

Regular expression	Example matched files	Description

`(?P<language>[^/.]*)/(?P<component>[^/]*)\.po`	`cs/application.po`
	`cs/website.po`
	`de/application.po`
	`de/website.po`
		One folder per language containing translation files for components.

`locale/(?P<language>[^/.]*)/LC_MESSAGES/(?P<component>[^/]*)\.po`	`locale/cs/LC_MESSAGES/application.po`
	`locale/cs/LC_MESSAGES/website.po`
	`locale/de/LC_MESSAGES/application.po`
	`locale/de/LC_MESSAGES/website.po`
		Usual structure for storing gettext PO files.

`src/locale/(?P<component>[^/]*)\.(?P<language>[^/.]*)\.po`	`src/locale/application.cs.po`
	`src/locale/website.cs.po`
	`src/locale/application.de.po`
	`src/locale/website.de.po`
		Using both component and language name within filename.

`locale/(?P<language>[^/.]*)/(?P<component>[^/]*)/(?P=language)\.po`	`locale/cs/application/cs.po`
	`locale/cs/website/cs.po`
	`locale/de/application/de.po`
	`locale/de/website/de.po`
		Using language in both path and filename.

`res/values-(?P<language>[^/.]*)/strings-(?P<component>[^/]*)\.xml`	`res/values-cs/strings-about.xml`
	`res/values-cs/strings-help.xml`
	`res/values-de/strings-about.xml`
	`res/values-de/strings-help.xml`
		Android resource strings, split into several files.

You can use Django template markup in both component name and the monolingual base filename, for example:

`{{ component }}`
Component filename match
`{{ component|title }}`
Component filename with upper case first letter

Save

Powered by Weblate 4.8 About Weblate Legal Contact Documentation Donate to Weblate

Sugestie: Component discovery add-on uses *Weblate internal URLs*. It's a convenient way to share VCS setup between multiple components. Linked components use the local repository of the main component set up by filling `weblate://project/main-component` into the *Depozitul de cod sursă* field (in *Manage* ↓ *Settings* ↓ *Version control system*) of each respective component. This saves time with configuration and system resources too.

Vezi și:

Template markup

Editare în masă

Nou în versiunea 3.11.

Add-on ID `weblate.flags.bulk`

Configurație

`q`	Interogare prin exemplu	
`state`	Stat pentru a seta	Available choices: `-1` – Do not change `10` – Needs editing `20` – Translated `30` – Approved
`add_flags`	Steguleț de traducere de adăugat	
`remove_flags`	Indicatoare de traducere de eliminat	
`add_labels`	Etichete de adăugat	
`remove_labels`	Etichete de îndepărtat	

Triggers component update

Editarea în bloc a steguleților, etichetelor sau stărilor șirurilor de caractere.

Automatizați etichetarea începând cu interogarea de căutare `NOT has:label` și adăugați etichete până când toate șirurile au toate etichetele necesare. Se pot efectua și alte operațiuni automatizate pentru metadatele Weblate.

Examples:

Tabelul 4: Etichetarea automată a șirurilor noi

Search query	`NOT has:label`
Etichete de adăugat	*recent*

Tabelul 5: Marcarea tuturor intrărilor din registrul de modificări Fișiere de metadate ale magazinului de aplicații doar pentru citire

Search query	`language:en AND key:changelogs/`
Steguleț de traducere de adăugat	`read-only`

Vezi și:

Editare în masă, *Personalizarea comportamentului cu ajutorul steguleților*, labels

Marcați traducerile neschimbate ca fiind „Needs editing”

Nou în versiunea 3.1.

Add-on ID `weblate.flags.same_edit`

Configurație *This add-on has no configuration.*

Triggers unit post-create

Ori de câte ori un nou șir traductibil este importat din VCS și se potrivește cu un șir sursă, acesta este marcat ca necesitând editare în Weblate. Este deosebit de util pentru formatele de fișiere care includ șiruri sursă pentru șiruri netraduse.

Sugestie: De asemenea, este posibil să doriți să întăriți verificarea *Traducere neschimbată* prin adăugarea indicatorului `strict-same` la *Steaguri de traducere*.

Vezi și:

Translation states

Marcați noile șiruri sursă ca fiind „Necesită editare”

Add-on ID `weblate.flags.source_edit`

Configurație *This add-on has no configuration.*

Triggers unit post-create

Ori de câte ori un nou șir sursă este importat din VCS, acesta este marcat ca necesitând editare în Weblate. În acest fel, puteți filtra și edita cu ușurință șirurile sursă scrise de dezvoltatori.

Vezi și:

Translation states

Marcați noile traduceri ca fiind „Needs editing”

Add-on ID `weblate.flags.target_edit`

Configurație *This add-on has no configuration.*

Triggers unit post-create

Ori de câte ori un nou șir traductibil este importat din VCS, acesta este marcat ca necesitând editare în Weblate. În acest fel, puteți filtra și edita cu ușurință traducerile create de dezvoltatori.

Vezi și:

Translation states

Generator de statistici

Add-on ID `weblate.generate.generate`

Configurație

`filename`	Numele fișierului generat	
`template`	Conținutul fișierului generat	

Triggers repository pre-commit

Generează un fișier care conține informații detaliate despre starea traducerii.

Puteți utiliza un șablon Django atât în numele fișierului, cât și în conținut; consultați *Template markup* pentru o descriere detaliată a marcajului.

For example generating a summary file for each translation:

Numele fișierului generat `locale/{{ language_code }}.json`

Conținut

```
{
   "language": "{{ language_code }}",
   "strings": "{{ stats.all }}",
   "translated": "{{ stats.translated }}",
   "last_changed": "{{ stats.last_changed }}",
   "last_author": "{{ stats.last_author }}",
}
```

Vezi și:

Template markup

Generarea de pseudolocale

Add-on ID `weblate.generate.pseudolocale`

Configurație

`source`	Șiruri de surse	
`target`	Traducerea țintă	
`prefix`	Prefixul șirului	
`suffix`	Sufixul șirului	

Triggers component update, daily

Generează o traducere prin adăugarea automată a prefixului și sufixului la șirurile sursă.

Pseudolocalele sunt utile pentru a găsi șiruri de caractere care nu sunt pregătite pentru localizare. Acest lucru se realizează prin modificarea tuturor șirurilor sursă traductibile pentru a facilita identificarea șirurilor nealterate la rularea aplicației în limba pseudolocală.

De asemenea, este posibil să se găsească șiruri de caractere ale căror omologi localizați ar putea să nu se potrivească cu aspectul.

Sugestie: Puteți folosi limbi reale pentru testare, dar există pseudolocale dedicate disponibile în Weblate - *en_XA* și *ar_XB*.

Sugestie: You can use this add-on to start translation to a new locale of an existing language or similar language. Once you add the translation to the component, follow to the add-on. *Example:* If you have *fr* and want to start *fr_CA* translation, simply set *fr* as the source, *fr_CA* as the target, and leave the prefix and suffix blank.

Uninstall the add-on once you have the new translation filled to prevent Weblate from changing the translations made after the copying.

Contributori în comentariu

Add-on ID `weblate.gettext.authors`

Configurație *This add-on has no configuration.*

Triggers repository pre-commit

Actualizează partea de comentarii din antetul fișierului PO pentru a include numele contribuabililor și anii de contribuție.

Antetul fișierului PO va arăta în felul următor:

```
# Michal Čihař <michal@cihar.com>, 2012, 2018, 2019, 2020.
# Pavel Borecki <pavel@example.com>, 2018, 2019.
# Filip Hron <filip@example.com>, 2018, 2019.
# anonymous <noreply@weblate.org>, 2019.
```

Actualizarea variabilei ALL_LINGUAS în fișierul „configure”

Add-on ID `weblate.gettext.configure`

Configurație *This add-on has no configuration.*

Triggers repository post-add, daily

Actualizează variabila ALL_LINGUAS din `configure`, `configure.in` sau din orice fișier `configure.ac`, atunci când se adaugă o nouă traducere.

Personalizați ieșirea gettext

Add-on ID `weblate.gettext.customize`

Configurație

`width`	Înfășurarea liniilor lungi	În mod implicit, gettext înfășoară liniile la 77 de caractere și la liniile noi. Dacă se folosește parametrul –no-wrap, se face o înfășurare numai la liniile noi. Available choices: `77` – Wrap lines at 77 characters and at newlines `65535` – Only wrap lines at newlines `-1` – No line wrapping

Triggers storage post-load

Permite personalizarea comportamentului ieșirii gettext, de exemplu înfășurarea liniilor.

Acesta oferă următoarele opțiuni:

- Întoarceți liniile la 77 de caractere și la liniile noi
- Întoarceți liniile numai la liniile noi
- Fără înfășurare de linie

Notă: În mod implicit, gettext înfășoară liniile la 77 de caractere și la liniile noi. Dacă se folosește parametrul `--no-wrap`, se face o încadrare numai la liniile noi.

Actualizarea fișierului LINGUAS

Add-on ID `weblate.gettext.linguas`

Configurație *This add-on has no configuration.*

Triggers repository post-add, daily

Actualizează fișierul LINGUAS atunci când se adaugă o nouă traducere.

Generarea de fișiere MO

Add-on ID `weblate.gettext.mo`

Configurație

`path`	Calea de acces a fișierului MO generat	Dacă nu este specificat, se va utiliza locația fișierului PO.

Triggers repository pre-commit

Generează automat un fișier MO pentru fiecare fișier PO modificat.

Locația fișierului MO generat poate fi personalizată, iar câmpul pentru acesta utilizează *Template markup*.

Actualizarea fișierelor PO pentru a se potrivi cu POT (msgmerge)

Add-on ID `weblate.gettext.msgmerge`

Configurație

`previous`	Păstrați mesajele anterioare ale șirurilor traduse	
`no_location`	Eliminați locațiile șirurilor traduse	
`fuzzy`	Utilizați potrivirea fuzzy	

Triggers repository post-update

Actualizează toate fișierele PO (as configured by *File mask*) pentru a se potrivi cu fișierul POT (as configured by *Șablon pentru traduceri noi*) utilizând **`msgmerge`**.

Triggered whenever new changes are pulled from the upstream repository. Most msgmerge command-line options can be set up through the add-on configuration.

Vezi și:

Does Weblate update translation files besides translations?

Comenzi Git Squash

Add-on ID `weblate.git.squash`

Configurație

`squash`	Compromiteți strivirea	Available choices: `all` – All commits into one `language` – Per language `file` – Per file `author` – Per author
`append_trailers`	Adăugați trailere la mesajul de confirmare comprimat	Liniile de final sunt linii care se aseamănă cu anteturile de e-mail RFC 822, la sfârșitul părții de formă liberă a unui mesaj de confirmare, cum ar fi „Co-autor: …”.
`commit_message`	Commit message	Acest mesaj de confirmare va fi utilizat în locul mesajelor de confirmare combinate de la confirmările strivite.

Triggers repository post-commit

Eliminarea modificărilor din Git înainte de a împinge modificările.

Comițările Git pot fi eliminate înainte de împingerea modificărilor în unul dintre următoarele moduri:

Nou în versiunea 3.4.

- Toate angajamentele într-una singură
- Pe limbă
- Pe fișier

Nou în versiunea 3.5.

- Pentru fiecare autor în parte

Mesajele de confirmare originale sunt păstrate, dar se pierde paternitatea, cu excepția cazului în care se selectează *Per author* sau mesajul de confirmare este personalizat pentru a o include.

Nou în versiunea 4.1.

Mesajele de confirmare originale pot fi opțional înlocuite cu un mesaj de confirmare personalizat.

Remetele (commit lines like `Co-authored-by: …`) pot fi opțional eliminate din mesajele de confirmare originale și adăugate la sfârșitul mesajului de confirmare strivit. Acest lucru generează, de asemenea, creditul corespunzător `Co-authored-by:` pentru fiecare traducător.

Personalizați ieșirea JSON

Add-on ID `weblate.json.customize`

Configurație

`sort_keys`	Sortarea cheilor JSON	
`indent`	Indentarea JSON	
`style`	JSON indentation style	Available choices: `spaces` – Spaces `tabs` – Tabs

Triggers storage post-load

Permite ajustarea comportamentului de ieșire JSON, de exemplu, indentarea sau sortarea.

Formatează fișierul de proprietăți Java

Add-on ID `weblate.properties.sort`

Configurație *This add-on has no configuration.*

Triggers repository pre-commit

Sortează fișierul de proprietăți Java.

Îndepărtarea modului învechit

Nou în versiunea 3.7.

Add-on ID `weblate.removal.comments`

Configurație

`age`	Zile de păstrare	

Triggers daily

Stabiliți un interval de timp pentru eliminarea comentariilor.

Acest lucru poate fi util pentru a elimina comentariile vechi care ar putea fi depășite. Folosiți-o cu grijă, deoarece faptul că comentariile devin vechi nu înseamnă că și-au pierdut importanța.

Eliminarea sugestiilor învechite

Nou în versiunea 3.7.

Add-on ID `weblate.removal.suggestions`

Configurație

`age`	Zile de păstrare	
`votes`	Pragul de vot	Pragul de eliminare. Acest câmp nu are niciun efect dacă votul este dezactivat.

Triggers daily

Stabiliți un termen pentru eliminarea sugestiilor.

Poate fi foarte utilă în legătură cu votarea sugestiilor (see *Peer review*) pentru a elimina sugestiile care nu primesc suficiente voturi pozitive într-un anumit interval de timp.

Actualizarea fișierelor RESX

Nou în versiunea 3.9.

Add-on ID `weblate.resx.update`

Configurație *This add-on has no configuration.*

Triggers repository post-update

Actualizarea tuturor fișierelor de traducere pentru a se potrivi cu fișierul de bază monolingv din amonte. Șirurile nefolosite sunt eliminate, iar cele noi sunt adăugate ca copii ale șirului sursă.

Sugestie: Utilizați *Curățați fișierele de traducere* dacă doriți să eliminați doar cheile de traducere vechi.

Vezi și:

Does Weblate update translation files besides translations?

Personalizați ieșirea YAML

Nou în versiunea 3.10.2.

Add-on ID `weblate.yaml.customize`

Configurație

`indent`	Indentarea YAML	
`width`	Înfășurarea liniilor lungi	Available choices: `80` – Wrap lines at 80 chars `100` – Wrap lines at 100 chars `120` – Wrap lines at 120 chars `180` – Wrap lines at 180 chars `65535` – No line wrapping
`line_break`	Pauze de linie	Available choices: `dos` – DOS (\r\n) `unix` – UNIX (\n) `mac` – MAC (\r)

Triggers storage post-load

Permite ajustarea comportamentului de ieșire YAML, de exemplu, lungimea liniei sau liniile noi.

2.14.2 Customizing list of add-ons

The list of add-ons is configured by `WEBLATE_ADDONS`. To add another add-on, simply include the absolute class name in this setting.

2.14.3 Writing add-on

You can write your own add-ons too, create a subclass of `weblate.addons.base.BaseAddon` to define the addon metadata, and then implement a callback to do the processing.

Vezi și:

Developing addons

2.14.4 Executing scripts from add-on

Add-ons can also be used to execute external scripts. This used to be integrated in Weblate, but now you have to write some code to wrap your script with an add-on.

```
#
# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
```

(continues on next page)

(continuare din pagina precedentă)

```
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Example pre commit script."""


from django.utils.translation import gettext_lazy as _

from weblate.addons.events import EVENT_PRE_COMMIT
from weblate.addons.scripts import BaseScriptAddon


class ExamplePreAddon(BaseScriptAddon):
    # Event used to trigger the script
    events = (EVENT_PRE_COMMIT,)
    # Name of the addon, has to be unique
    name = "weblate.example.pre"
    # Verbose name and long descrption
    verbose = _("Execute script before commit")
    description = _("This addon executes a script.")

    # Script to execute
    script = "/bin/true"
    # File to add in commit (for pre commit event)
    # does not have to be set
    add_file = "po/{{ language_code }}.po"
```

Pentru instrucțiuni de instalare, consultați *Custom quality checks, addons and auto-fixes*.

Scriptul este executat cu directorul curent setat la rădăcina depozitului VCS pentru orice componentă dată.

În plus, sunt disponibile următoarele variabile de mediu:

WL_VCS
Sistemul de control al versiunilor utilizat.

WL_REPO
URL-ul depozitului din amonte.

WL_PATH
Calea absolută către depozitul VCS.

WL_BRANCH
Nou în versiunea 2.11.

Ramura de depozit configurată în componenta curentă.

WL_FILEMASK
Masca de fișier pentru componenta curentă.

WL_TEMPLATE
Numele de fișier al șablonului pentru traducerile monolingve (poate fi gol).

WL_NEW_BASE
Nou în versiunea 2.14.

Numele de fișier al fișierului utilizat pentru crearea de noi traduceri (poate fi gol).

WL_FILE_FORMAT
Format de fișier utilizat în componenta curentă.

WL_LANGUAGE
Limba traducerii procesate în prezent (nu este disponibilă pentru cârligele la nivel de componentă).

WL_PREVIOUS_HEAD
HEAD anterior după actualizare (disponibil numai după rularea cârligului post-update).

WL_COMPONENT_SLUG
Nou în versiunea 3.9.

Denumirea componentei utilizată pentru a construi URL-ul.

WL_PROJECT_SLUG
Nou în versiunea 3.9.

Denumirea proiectului utilizată pentru a construi URL-ul.

WL_COMPONENT_NAME
Nou în versiunea 3.9.

Denumirea componentei.

WL_PROJECT_NAME
Nou în versiunea 3.9.

Numele proiectului.

WL_COMPONENT_URL
Nou în versiunea 3.9.

URL-ul componentei.

WL_ENGAGE_URL
Nou în versiunea 3.9.

URL de angajare a proiectului.

Vezi și:

Component configuration

Post-update repository processing

Poate fi utilizat pentru a actualiza fișierele de traducere atunci când sursa VCS din amonte se modifică. Pentru a realiza acest lucru, vă rugăm să rețineți că Weblate vede numai fișierele trimise în VCS, deci trebuie să trimiteți modificările ca parte a scriptului.

De exemplu, cu Gulp puteți face acest lucru folosind următorul cod:

```
#! /bin/sh
gulp --gulpfile gulp-i18n-extract.js
git commit -m 'Update source strings' src/languages/en.lang.json
```

Pre-commit processing of translations

Utilizați scriptul de trimitere pentru a modifica automat o traducere înainte de a fi trimisă în depozit.

Acesta este transmis ca un singur parametru care constă în numele de fișier al unei traduceri curente.

2.15 Memorie de traducere

Nou în versiunea 2.20.

Weblate comes with a built-in translation memory consisting of the following:

- Manually imported translation memory (see *User interface*).
- Automatically stored translations performed in Weblate (depending on *Translation memory scopes*).
- Automatically imported past translations.

Content in the translation memory can be applied one of two ways:

- Manually, *Sugestii automate* view while translating.
- Automatically, by translating strings using *Traducere automată*, or *Traducere automată* addon.

For installation tips, see *Weblate Translation Memory*, which is turned on by default.

2.15.1 Translation memory scopes

Nou în versiunea 3.2: In earlier versions translation memory could be only loaded from a file corresponding to the current imported translation memory scope.

The translation memory scopes are there to allow both privacy and sharing of translations, to suit the desired behavior.

Imported translation memory

Importing arbitrary translation memory data using the `import_memory` command makes memory content available to all users and projects.

Per user translation memory

Stores all user translations automatically in the personal translation memory of each respective user.

Per project translation memory

All translations within a project are automatically stored in a project translation memory only available for this project.

Memorie partajată de traducere

All translation within projects with shared translation memory turned on are stored in a shared translation memory available to all projects.

Please consider carefully whether to turn this feature on for shared Weblate installations, as it can have severe implications:

- The translations can be used by anybody else.
- This might lead to disclosing secret information.

2.15.2 Managing translation memory

User interface

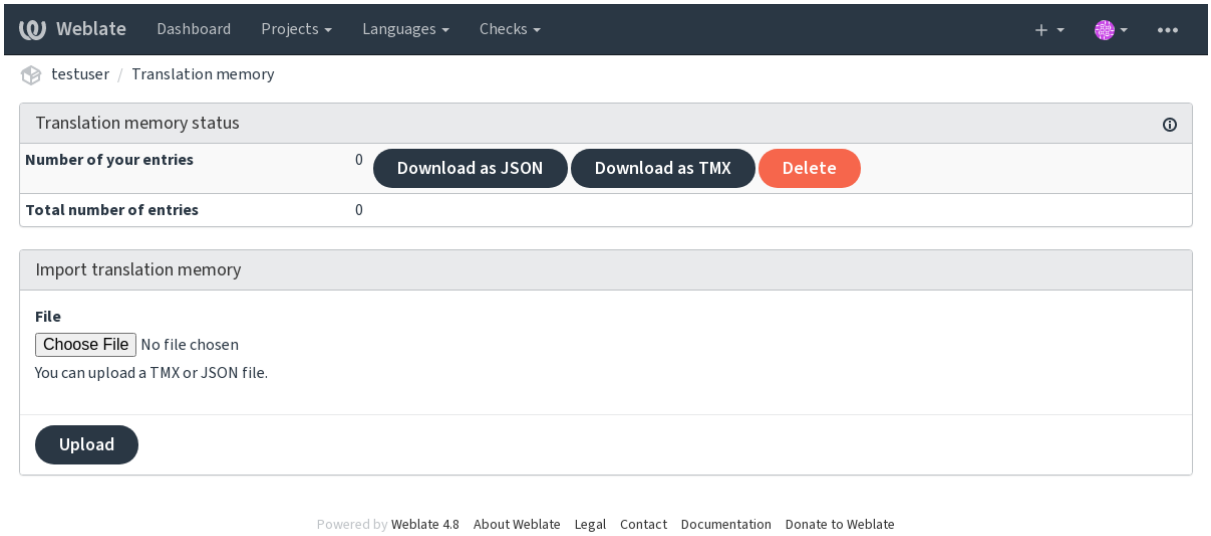
Nou în versiunea 3.2.

In the basic user interface you can manage per user and per project translation memories. It can be used to download, wipe or import translation memory.

Sugestie: Translation memory in JSON can be imported into Weblate, TMX is provided for interoperability with other tools.

Vezi și:

Weblate Translation Memory Schema



Interfața de gestionare

There are several management commands to manipulate the translation memory content. These operate on the translation memory as whole, unfiltered by scopes (unless requested by parameters):

dump_memory Exports the memory into JSON

import_memory Imports TMX or JSON files into the translation memory

2.16 Configurație

Toate setările sunt stocate în `settings.py` (așa cum este normal pentru Django).

Notă: După modificarea oricăreia dintre aceste setări, trebuie să reporniți Weblate - atât procesele WSGI, cât și Celery.

În cazul în care este rulat ca `mod_wsgi`, trebuie să reporniți Apache pentru a reîncărca configurația.

Vezi și:

De asemenea, consultați Documentația Django pentru parametrii de configurare a lui Django.

2.16.1 AKISMET_API_KEY

Weblate poate utiliza Akismet pentru a verifica dacă sugestiile anonime primite sunt spam. Vizitați akismet.com pentru a achiziționa o cheie API și a o asocia cu un site.

2.16.2 ANONYMOUS_USER_NAME

Numele de utilizator al utilizatorilor care nu sunt conectați.

Vezi și:

Controlul accesului

2.16.3 AUDITLOG_EXPIRY

Nou în versiunea 3.6.

Câte zile trebuie să păstreze Weblate jurnalele de audit, care conțin informații despre activitatea contului.

Valoarea implicită este de 180 de zile.

2.16.4 AUTH_LOCK_ATTEMPTS

Nou în versiunea 2.14.

Numărul maxim de încercări de autentificare eșuate înainte de a se aplica limitarea vitezei.

Aceasta se aplică în prezent în următoarele locații:

- Autentificare. Șterge parola contului, împiedicând utilizatorul să se conecteze fără a solicita o nouă parolă.
- Parola se resetează. Împiedică trimiterea de noi e-mailuri, evitând spamul utilizatorilor cu prea multe încercări de resetare a parolei.

Valoarea implicită este 10.

Vezi și:

Rate limiting,

2.16.5 AUTO_UPDATE

Nou în versiunea 3.2.

Schimbat în versiunea 3.11: Opțiunea originală de activare/dezactivare a fost schimbată pentru a diferenția ce șiruri de caractere sunt acceptate.Opțiunea originală de activare/dezactivare a fost schimbată pentru a diferenția ce șiruri de caractere sunt acceptate.

Actualizează zilnic toate depozitele.

Sugestie: Util dacă nu folosiți *Cârlige de notificare* pentru a actualiza automat depozitele Weblate.

Notă: Pe lângă selectarea șirurilor de caractere, există opțiuni de activare/dezactivare pentru compatibilitate retroactivă.

Opțiunile sunt:

`"none"` Nu există actualizări zilnice.

"remote" also False Actualizați numai telecomenzile.

"full" also True Actualizarea telecomenzilor și îmbinarea copiei de lucru.

Notă: Acest lucru necesită ca *Background tasks using Celery* să funcționeze și va intra în vigoare după ce va fi repornit.

2.16.6 AVATAR_URL_PREFIX

Prefix pentru construirea URL-urilor avatarurilor ca: `${{AVATAR_URL_PREFIX}}/avatar/${MAIL_HASH}?${{PARAMS}}`. Se știe că funcționează următoarele servicii:

Gravatar (implicit), conform https://gravatar.com/ `AVATAR_URL_PREFIX = 'https://www.gravatar.com/'`

Libravatar, conform https://www.libravatar.org/ `AVATAR_URL_PREFIX = 'https://www.libravatar.org/'`

Vezi și:

Avatar caching, `ENABLE_AVATARS`, *Avatars*

2.16.7 AUTH_TOKEN_VALID

Nou în versiunea 2.14.

Cât timp sunt valabile simbolul de autentificare și parola temporară din e-mailurile de resetare a parolei. Se setează în număr de secunde, implicit la 172800 (2 zile).

2.16.8 AUTH_PASSWORD_DAYS

Nou în versiunea 2.15.

Câte zile de utilizare a aceleiași parole trebuie să fie permise.

Notă: Modificările de parole efectuate înainte de Weblate 2.15 nu vor fi luate în considerare în această politică.

Valoarea implicită este de 180 de zile.

2.16.9 AUTOFIX_LIST

Lista de corecții automate care trebuie aplicate la salvarea unui șir de caractere.

Notă: Furnizați o cale complet calificată către clasa Python care implementează interfața autofixer.

Corecții disponibile:

weblate.trans.autofixes.whitespace.SameBookendingWhitespace Potrivește spațiile albe de la începutul și sfârșitul șirului cu sursa.

weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis Înlocuiește punctele de sfârșit (….) dacă șirul sursă are o elipsă corespunzătoare (…).

weblate.trans.autofixes.chars.RemoveZeroSpace Îndepărtează caracterele de spațiu de lățime zero dacă sursa nu conține niciunul.

weblate.trans.autofixes.chars.RemoveControlChars Elimină caracterele de control în cazul în care sursa nu conține niciunul.

weblate.trans.autofixes.html.BleachHTML Îndepărtează marcajele HTML nesigure din șirurile marcate ca `safe-html` (see *HTML nesigur*).

Puteți selecta pe care să le utilizați:

```
AUTOFIX_LIST = (
    "weblate.trans.autofixes.whitespace.SameBookendingWhitespace",
    "weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis",
)
```

Vezi și:

Automatic fixups, *Reparații automate personalizate*

2.16.10 BACKGROUND_TASKS

Nou în versiunea 4.5.2.

Definește cât de des trebuie declanșate sarcinile de întreținere de lungă durată pentru o componentă.

În momentul de față acest lucru controlează:

- *Traducere automată* addon
- *Verificări și reparații* recalculare

Opțiuni posibile:

- `monthly` (aceasta este valoarea implicită)
- `weekly`
- `daily`
- `never`

Notă: Creșterea frecvenței nu este recomandată atunci când Weblate conține mii de componente.

2.16.11 BASE_DIR

Directorul de bază în care sunt localizate sursele Weblate. Utilizat pentru a deriva în mod implicit alte câteva căi de acces:

- *DATA_DIR*

Valoarea implicită: Directorul de nivel superior al surselor Weblate.

2.16.12 BASIC_LANGUAGES

Nou în versiunea 4.4.

List of languages to offer users for starting new translation. When not specified built-in list is used which includes all commonly used languages, but without country specific variants.

Acest lucru limitează doar utilizatorii neprivilegiați să adauge limbi nedorite. Administratorilor de proiect li se prezintă în continuare o selecție completă a limbilor definite în Weblate.

Notă: Acest lucru nu definește noi limbi pentru Weblate, ci doar le filtrează pe cele existente în baza de date.

Example:

```
BASIC_LANGUAGES = {"cs", "it", "ja", "en"}
```

Vezi și:

Language definitions

2.16.13 CSP_SCRIPT_SRC, CSP_IMG_SRC, CSP_CONNECT_SRC, CSP_STYLE_SRC, CSP_FONT_SRC

Personalizați antetul `Content-Security-Policy` pentru Weblate. Antetul este generat automat pe baza integrărilor activate cu servicii terțe (Matomo, Google Analytics, Sentry, …).

Toate acestea au ca valoare implicită o listă goală.

Example:

```
# Enable Cloudflare Javascript optimizations
CSP_SCRIPT_SRC = ["ajax.cloudflare.com"]
```

Vezi și:

Content security policy, Content Security Policy (CSP)

2.16.14 CHECK_LIST

Lista controalelor de calitate care trebuie efectuate asupra unei traduceri.

Notă: Furnizați o cale complet calificată către clasa Python care implementează interfața de verificare.

Ajustați lista de controale pentru a le include pe cele relevante pentru dumneavoastră.

Toate verificările încorporate *Quality checks* sunt activate în mod implicit, de unde puteți modifica aceste setări. În mod implicit, acestea sunt comentate în *Sample configuration*, astfel încât sunt utilizate valorile implicite. Se efectuează apoi noi verificări pentru fiecare nouă versiune Weblate.

Puteți dezactiva toate verificările:

```
CHECK_LIST = ()
```

Puteți porni doar câteva:

```
CHECK_LIST = (
    "weblate.checks.chars.BeginNewlineCheck",
    "weblate.checks.chars.EndNewlineCheck",
    "weblate.checks.chars.MaxLengthCheck",
)
```

Notă: Modificarea acestei setări afectează doar traducerile nou modificate, verificările existente vor fi stocate în continuare în baza de date. Pentru a aplica, de asemenea, modificările la traducerile stocate, rulați `updatechecks`.

Vezi și:

Quality checks, *Personalizarea comportamentului cu ajutorul steguleților*

2.16.15 COMMENT_CLEANUP_DAYS

Nou în versiunea 3.6.

Ștergeți comentariile după un anumit număr de zile. Valoarea implicită este `None`, ceea ce înseamnă că nu se șterge deloc.

2.16.16 COMMIT_PENDING_HOURS

Nou în versiunea 2.10.

Numărul de ore care trec între confirmarea modificărilor în așteptare prin intermediul sarcinii de fundal.

Vezi și:

Component configuration, *Vârsta modificărilor care urmează să fie confirmate*, *Running maintenance tasks*, `commit_pending`

2.16.17 CONTACT_FORM

Nou în versiunea 4.6.

Configures how e-mail from the contact form is being sent. Choose a configuration that matches your mail server configuration.

`"reply-to"` The sender is used in as *`Reply-To`*, this is the default behaviour.

`"from"` The sender is used in as *`From`*. Your mail server needs to allow sending such e-mails.

2.16.18 DATA_DIR

Dosarul în care Weblate stochează toate datele. Acesta conține linkuri către depozitele VCS, un index fulltext și diverse fișiere de configurare pentru instrumente externe.

De obicei, există următoarele subdirectoare:

`home` Directorul principal utilizat pentru apelarea scripturilor.

`ssh` Chei SSH și configurare.

`static` Locația implicită pentru fișierele Django statice, specificată de `STATIC_ROOT`. Vezi *Serving static files*.

The Docker container uses a separate volume for this, see *Docker container volumes*.

`media` Default location for Django media files, specified by `MEDIA_ROOT`. Contains uploaded screenshots, see *Visual context for strings*.

`vcs` Version control repositories for translations.

`backups` Datele zilnice de backup, vă rugăm să verificați *Date descărcate pentru copii de rezervă* pentru detalii.

`celery` Date despre programatorul Celery, a se vedea *Background tasks using Celery*.

`fonts:` Fonturi încărcate de utilizator, vezi *Gestionarea fonturilor*.

Notă: Acest director trebuie să poată fi scris de Weblate. Rularea ca uWSGI înseamnă că utilizatorul `www-data` trebuie să aibă acces de scriere la el.

Cel mai simplu mod de a realiza acest lucru este de a transforma utilizatorul în proprietar al directorului:

```
sudo chown www-data:www-data -R $DATA_DIR
```

Valoarea implicită este `$BASE_DIR/data`.

Vezi și:

`BASE_DIR`, *Filesystem permissions*, *Copierea de rezervă și mutarea Weblate*

2.16.19 DATABASE_BACKUP

Nou în versiunea 3.1.

Dacă copiile de rezervă ale bazei de date trebuie să fie stocate ca text simplu, comprimate sau sărite. Valorile autorizate sunt:

- `"plain"`
- `"comprimat"`
- `"none"`

Vezi și:

Copierea de rezervă și mutarea Weblate

2.16.20 DEFAULT_ACCESS_CONTROL

Nou în versiunea 3.3.

Setarea implicită de control al accesului pentru proiectele noi:

`0` *Public*

`1` *Protected*

`100` *Private*

`200` *Custom*

Utilizați *Custom* dacă gestionați ACL manual, ceea ce înseamnă că nu vă bazați pe gestionarea internă a Weblate.

Vezi și:

Control acces proiect, *Controlul accesului*

2.16.21 DEFAULT_AUTO_WATCH

Nou în versiunea 4.5.

Configurează dacă *Automatically watch projects on contribution* trebuie activat pentru utilizatorii noi. Valoarea implicită este `True`.

Vezi și:

Notificări

2.16.22 DEFAULT_RESTRICTED_COMPONENT

Nou în versiunea 4.1.

Valoarea implicită pentru restricția componentelor.

Vezi și:

Restricted access, *Scope of groups*

2.16.23 DEFAULT_ADD_MESSAGE, DEFAULT_ADDON_MESSAGE, DEFAULT_COMMIT_MESSAGE, DEFAULT_DELETE_MESSAGE, DEFAULT_MERGE_MESSAGE

Mesaje de confirmare implicite pentru diferite operații, vă rugăm să verificați *Component configuration* pentru detalii.

Vezi și:

Template markup, *Component configuration*, *Commit, add, delete, merge and addon messages*

2.16.24 DEFAULT_ADDONS

Suplimentele implicite care se instalează pe fiecare componentă creată.

Notă: Această setare afectează numai componentele nou create.

Exemplu:

```
DEFAULT_ADDONS = {
    # Add-on with no parameters
    "weblate.flags.target_edit": {},
    # Add-on with parameters
    "weblate.autotranslate.autotranslate": {
        "mode": "suggest",
        "filter_type": "todo",
        "auto_source": "mt",
        "component": "",
        "engines": ["weblate-translation-memory"],
        "threshold": "80",
    },
}
```

Vezi și:

`install_addon`, *Add-ons*, `WEBLATE_ADDONS`

2.16.25 DEFAULT_COMMITER_EMAIL

Nou în versiunea 2.4.

Adresă de e-mail a autorului de modificare implicită `noreply@weblate.org`.

Vezi și:

`DEFAULT_COMMITER_NAME`

2.16.26 DEFAULT_COMMITER_NAME

Nou în versiunea 2.4.

Numele de comisionar implicit la `Weblate`.

Vezi și:

DEFAULT_COMMITER_EMAIL

2.16.27 DEFAULT_LANGUAGE

Nou în versiunea 4.3.2.

Limba sursă implicită de utilizat, de exemplu în *Limba sursă*.

Valoarea implicită este *en*. Obiectul de limbă corespunzător trebuie să existe în baza de date.

Vezi și:

Language definitions, *Limba sursă*

2.16.28 DEFAULT_MERGE_STYLE

Nou în versiunea 3.4.

Stilul de fuziune pentru orice componentă nouă.

- *rebase* - default
- *merge*

Vezi și:

Component configuration, *Stil de fuziune*

2.16.29 DEFAULT_SHARED_TM

Nou în versiunea 3.2.

Configurează valoarea implicită a *Utilizați memoria de traducere partajată* și *Contribuie la memoria de traducere partajată*.

2.16.30 DEFAULT_TRANSLATION_PROPAGATION

Nou în versiunea 2.5.

Setare implicită pentru propagarea traducerii, valoarea implicită este `True`.

Vezi și:

Component configuration, *Permiteți propagarea traducerii*

2.16.31 DEFAULT_PULL_MESSAGE

Titlu pentru noile cereri de extragere, implicit `'Update from Weblate'`.

2.16.32 ENABLE_AVATARS

Dacă se activează avatarele bazate pe Gravatar pentru utilizatori. În mod implicit, această opțiune este activată.

Avatarurile sunt preluate și stocate în memoria cache pe server, ceea ce reduce riscul de scurgere a informațiilor private și accelerează experiența utilizatorului.

Vezi și:

Avatar caching, `AVATAR_URL_PREFIX`, *Avatars*

2.16.33 ENABLE_HOOKS

Dacă se activează cârligele anonime de la distanță.

Vezi și:

Cârlige de notificare

2.16.34 ENABLE_HTTPS

Dacă să trimiteți linkuri către Weblate ca HTTPS sau HTTP. Această setare afectează e-mailurile trimise și URL-urile absolute generate.

În configurația implicită, acesta este, de asemenea, utilizat pentru mai multe setări Django legate de HTTPS - activează cookie-urile securizate, activează HSTS sau permite redirecționarea către un URL HTTPS.

Redirecționarea HTTPS ar putea fi problematică în unele cazuri și s-ar putea să vă confruntați cu o redirecționare infinită în cazul în care utilizați un proxy invers care face terminarea SSL și care nu transmite corect antetele de protocol către Django. Vă rugăm să modificați configurația proxy-ului dvs. invers pentru a emite antetele `X-Forwarded-Proto` sau `Forwarded` sau configurați `SECURE_PROXY_SSL_HEADER` pentru a permite Django să detecteze corect starea SSL.

Vezi și:

`SESSION_COOKIE_SECURE`, `CSRF_COOKIE_SECURE`, `SECURE_SSL_REDIRECT`, `SECURE_PROXY_SSL_HEADER` *Set correct site domain*

2.16.35 ENABLE_SHARING

Activați/dezactivați meniul *Share* pentru ca utilizatorii să poată partaja progresul traducerii pe rețelele sociale.

2.16.36 GET_HELP_URL

Nou în versiunea 4.5.2.

URL-ul unde poate fi găsit suport pentru instanța Weblate.

2.16.37 GITLAB_CREDENTIALS

Nou în versiunea 4.3.

Lista pentru acreditările pentru serverele GitLab.

Sugestie: Utilizați acest lucru în cazul în care doriți ca Weblate să interacționeze cu mai multe dintre ele, pentru un singur punct final GitLab folosiți `GITLAB_USERNAME` și `GITLAB_TOKEN`.

```
GITLAB_CREDENTIALS = {
    "gitlab.com": {
        "username": "weblate",
        "token": "your-api-token",
    },
    "gitlab.example.com": {
        "username": "weblate",
        "token": "another-api-token",
    },
}
```

2.16.38 GITLAB_USERNAME

Numele de utilizator GitLab utilizat pentru a trimite cereri de fuziune pentru actualizări de traduceri.

Vezi și:

`GITLAB_CREDENTIALS`, *GitLab*

2.16.39 GITLAB_TOKEN

Nou în versiunea 4.3.

Token de acces personal GitLab utilizat pentru a efectua apeluri API pentru actualizări de traduceri.

Vezi și:

`GITLAB_CREDENTIALS`, *GitLab*, GitLab: Personal access token

2.16.40 GITHUB_CREDENTIALS

Nou în versiunea 4.3.

Lista pentru acreditările pentru serverele GitHub.

Sugestie: Utilizați acest lucru în cazul în care doriți ca Weblate să interacționeze cu mai multe dintre ele, pentru un singur punct final GitHub folosiți `GITHUB_USERNAME` și `GITHUB_TOKEN`.

```
GITHUB_CREDENTIALS = {
    "api.github.com": {
        "username": "weblate",
        "token": "your-api-token",
    },
    "github.example.com": {
        "username": "weblate",
        "token": "another-api-token",
    },
}
```

2.16.41 GITHUB_USERNAME

Numele de utilizator GitHub utilizat pentru a trimite cereri de actualizare a traducerilor.

Vezi și:

GITHUB_CREDENTIALS, *GitHub*

2.16.42 GITHUB_TOKEN

Nou în versiunea 4.3.

Token de acces personal GitHub utilizat pentru a efectua apeluri API pentru a trimite cereri de extragere pentru actualizări de traduceri.

Vezi și:

GITHUB_CREDENTIALS, *GitHub*, Creating a GitHub personal access token

2.16.43 GOOGLE_ANALYTICS_ID

ID-ul Google Analytics pentru a activa monitorizarea Weblate cu ajutorul Google Analytics.

2.16.44 HIDE_REPO_CREDENTIALS

Ascundeți acreditările de depozit din interfața web. În cazul în care aveți un URL de depozit cu utilizator și parolă, Weblate îl va ascunde atunci când informațiile aferente sunt afișate utilizatorilor.

De exemplu, în loc de `https://user:password@git.example.com/repo.git`, va apărea doar `https://git.example.com/repo.git`. În mod similar, încearcă să curețe și mesajele de eroare VCS.

Notă: Acest lucru este activat în mod implicit.

2.16.45 HIDE_VERSION

Nou în versiunea 4.3.1.

Ascunde informațiile despre versiune de utilizatorii neautentificați. De asemenea, toate legăturile de documentație indică ultima versiune în loc de documentația care corespunde versiunii instalate în prezent.

Ascunderea versiunii este o practică de securitate recomandată în unele corporații, dar aceasta nu împiedică atacatorul să afle versiunea prin sondarea comportamentului.

Notă: Acest lucru este dezactivat în mod implicit.

2.16.46 IP_BEHIND_REVERSE_PROXY

Nou în versiunea 2.14.

Indică dacă Weblate rulează în spatele unui proxy invers.

Dacă este setat la `True`, Weblate obține adresa IP dintr-un antet definit de *IP_PROXY_HEADER*.

> **Atenționare:** Asigurați-vă că utilizați într-adevăr un proxy invers și că acesta setează acest antet, altfel utilizatorii vor putea falsifica adresa IP.

Notă: Acest lucru nu este activat în mod implicit.

Vezi și:

Running behind reverse proxy, *Rate limiting*, *IP_PROXY_HEADER*, *IP_PROXY_OFFSET*

2.16.47 IP_PROXY_HEADER

Nou în versiunea 2.14.

Indică antetul din care Weblate ar trebui să obțină adresa IP atunci când *IP_BEHIND_REVERSE_PROXY* este activat.

Valoarea implicită este `HTTP_X_FORWARDED_FOR`.

Vezi și:

Running behind reverse proxy, :ref:``rate-limit`, `SECURE_PROXY_SSL_HEADER`, *IP_BEHIND_REVERSE_PROXY*, *IP_PROXY_OFFSET*

2.16.48 IP_PROXY_OFFSET

Nou în versiunea 2.14.

Indică ce parte din *IP_PROXY_HEADER* este utilizată ca adresă IP a clientului.

În funcție de configurația dvs., acest antet poate fi format din mai multe adrese IP (de exemplu `X-Forwarded-For: a, b, client-ip`) și puteți configura aici care adresă din antet este utilizată ca adresă IP a clientului.

> **Atenționare:** Această setare afectează securitatea instalației dvs. și ar trebui să o configurați pentru a utiliza numai proxy-uri de încredere pentru a determina adresa IP.

Valoarea implicită este 0.

Vezi și:

Running behind reverse proxy, *Rate limiting*, `SECURE_PROXY_SSL_HEADER`, *IP_BEHIND_REVERSE_PROXY*, *IP_PROXY_HEADER*

2.16.49 LEGAL_URL

Nou în versiunea 3.5.

URL-ul unde instanța Weblate își afișează documentele legale.

Sugestie: Utile dacă vă găzduiți documentele juridice în afara Weblate pentru a le încorpora în Weblate, vă rugăm să verificați *Legal* pentru detalii.

Exemplu:

```
LEGAL_URL = "https://weblate.org/terms/"
```

Vezi și:

PRIVACY_URL

2.16.50 LICENSE_EXTRA

Licențe suplimentare de inclus în opțiunile de licență.

Notă: Fiecare definiție a licenței trebuie să fie un tandem format din numele scurt, numele lung și o adresă URL.

De exemplu:

```
LICENSE_EXTRA = [
    (
        "AGPL-3.0",
        "GNU Affero General Public License v3.0",
        "https://www.gnu.org/licenses/agpl-3.0-standalone.html",
    ),
]
```

2.16.51 LICENSE_FILTER

Schimbat în versiunea 4.3: Setarea acestei valori la o valoare goală dezactivează acum alerta de licență.

Filtrați lista de licențe care trebuie afișate. Aceasta dezactivează, de asemenea, alerta de licență atunci când este setată la gol.

Notă: Acest filtru utilizează denumirile scurte ale licențelor.

De exemplu:

```
LICENSE_FILTER = {"AGPL-3.0", "GPL-3.0-or-later"}
```

Următoarele dezactivează alerta de licență:

```
LICENSE_FILTER = set()
```

Vezi și:

alerts

2.16.52 LICENSE_REQUIRED

Definește dacă atributul license din *Component configuration* este necesar.

Notă: Acest lucru este dezactivat în mod implicit.

2.16.53 LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH

Dacă lungimea unei anumite traduceri trebuie limitată. Restricția este lungimea șirului sursă * 10 caractere.

Sugestie: Setați această opțiune la `False` pentru a permite traduceri mai lungi (până la 10.000 de caractere), indiferent de lungimea șirului sursă.

Notă: Valoarea implicită este `True`.

2.16.54 LOCALIZE_CDN_URL și LOCALIZE_CDN_PATH

Aceste setări configurează addon-ul *CDN de localizare JavaScript*. *LOCALIZE_CDN_URL* definește URL-ul rădăcină unde este disponibil CDN-ul de localizare și *LOCALIZE_CDN_PATH* definește calea unde Weblate ar trebui să stocheze fișierele generate care vor fi servite la *LOCALIZE_CDN_URL*.

Sugestie: Pe Hosted Weblate, acesta folosește `https://weblate-cdn.com/`.

Vezi și:

CDN de localizare JavaScript

2.16.55 LOGIN_REQUIRED_URLS

O listă de URL-uri la care doriți să solicitați conectarea. (În afară de regulile standard integrate în Weblate).

Sugestie: Acest lucru vă permite să protejați cu parolă o întreagă instalație folosind:

```
LOGIN_REQUIRED_URLS = (r"/(.*)$",)
REST_FRAMEWORK["DEFAULT_PERMISSION_CLASSES"] = [
    "rest_framework.permissions.IsAuthenticated"
]
```

Sugestie: Este de dorit să se blocheze și accesul la API, așa cum se arată în exemplul de mai sus.

Vezi și:

REQUIRE_LOGIN

2.16.56 LOGIN_REQUIRED_URLS_EXCEPTIONS

Lista de excepții pentru *LOGIN_REQUIRED_URLS*. Dacă nu este specificat, utilizatorilor li se permite să acceseze pagina de conectare.

Unele dintre excepțiile pe care ați putea dori să le includeți:

```
LOGIN_REQUIRED_URLS_EXCEPTIONS = (
    r"/accounts/(.*)$",  # Required for sign in
    r"/static/(.*)$",  # Required for development mode
    r"/widgets/(.*)$",  # Allowing public access to widgets
    r"/data/(.*)$",  # Allowing public access to data exports
    r"/hooks/(.*)$",  # Allowing public access to notification hooks
    r"/api/(.*)$",  # Allowing access to API
    r"/js/i18n/$",  # JavaScript localization
)
```

2.16.57 MATOMO_SITE_ID

ID-ul unui site din Matomo (fostul Piwik) pe care doriți să îl urmăriți.

Notă: Această integrare nu este compatibilă cu Matomo Tag Manager.

Vezi și:

MATOMO_URL

2.16.58 MATOMO_URL

URL-ul complet (inclusiv bara oblică finală) al unei instalații Matomo (fostă Piwik) pe care doriți să o utilizați pentru a urmări utilizarea Weblate. Vă rugăm să consultați <https://matomo.org/> pentru mai multe detalii.

Sugestie: Această integrare nu este compatibilă cu Matomo Tag Manager.

De exemplu:

```
MATOMO_SITE_ID = 1
MATOMO_URL = "https://example.matomo.cloud/"
```

Vezi și:

MATOMO_SITE_ID

2.16.59 MT_SERVICES

Schimbat în versiunea 3.0: Setarea a fost redenumită din `MACHINE_TRANSLATION_SERVICES` în `MT_SERVICES` pentru a fi în concordanță cu alte setări de traducere automată.

Lista de servicii de traducere automată care pot fi utilizate.

Notă: Multe dintre servicii au nevoie de o configurație suplimentară, cum ar fi cheile API, vă rugăm să verificați documentația lor *Traducere automată* pentru mai multe detalii.

Sugestie: When using Docker container, this configuration is automatically generated based on provided API keys, see *Machine translation settings*.

```
MT_SERVICES = (
    "weblate.machinery.apertium.ApertiumAPYTranslation",
    "weblate.machinery.deepl.DeepLTranslation",
    "weblate.machinery.glosbe.GlosbeTranslation",
    "weblate.machinery.google.GoogleTranslation",
    "weblate.machinery.libretranslate.LibreTranslateTranslation",
    "weblate.machinery.microsoft.MicrosoftCognitiveTranslation",
    "weblate.machinery.microsoftterminology.MicrosoftTerminologyService",
    "weblate.machinery.mymemory.MyMemoryTranslation",
    "weblate.machinery.tmserver.AmagamaTranslation",
    "weblate.machinery.tmserver.TMServerTranslation",
    "weblate.machinery.yandex.YandexTranslation",
    "weblate.machinery.weblatetm.WeblateTranslation",
    "weblate.machinery.saptranslationhub.SAPTranslationHub",
    "weblate.memory.machine.WeblateMemory",
)
```

Vezi și:

Traducere automată, *Sugestii automate*

2.16.60 MT_APERTIUM_APY

URL of the Apertium-APy server, https://wiki.apertium.org/wiki/Apertium-apy

Vezi și:

Apertium, *Traducere automată*, *Sugestii automate*

2.16.61 MT_AWS_ACCESS_KEY_ID

ID-ul cheii de acces pentru Amazon Translate.

Vezi și:

AWS, *Traducere automată*, *Sugestii automate*

2.16.62 MT_AWS_SECRET_ACCES_KEY

Cheia secretă API pentru Amazon Translate.

Vezi și:

AWS, *Traducere automată*, *Sugestii automate*

2.16.63 MT_AWS_REGION

Numele regiunii de utilizat pentru Amazon Translate.

Vezi și:

AWS, *Traducere automată*, *Sugestii automate*

2.16.64 MT_BAIDU_ID

ID-ul de client pentru Baidu Zhiyun API, vă puteți înregistra la https://api.fanyi.baidu.com/api/trans/product/index

Vezi și:

Baidu API machine translation, *Traducere automată*, *Sugestii automate*

2.16.65 MT_BAIDU_SECRET

Client secret pentru Baidu Zhiyun API, vă puteți înregistra la https://api.fanyi.baidu.com/api/trans/product/index

Vezi și:

Baidu API machine translation, *Traducere automată*, *Sugestii automate*

2.16.66 MT_DEEPL_API_URL

Schimbat în versiunea 4.7: The full API URL is now configured to allow using the free plan. Before, it was only possible to configure the API version using `MT_DEEPL_API_VERSION`.

API URL to use with the DeepL service. At the time of writing, there is the v1 API as well as a free and a paid version of the v2 API.

`https://api.deepl.com/v2/` (default in Weblate) Is meant for API usage on the paid plan, and the subscription is usage-based.

`https://api-free.deepl.com/v2/` Is meant for API usage on the free plan, and the subscription is usage-based.

`https://api.deepl.com/v1/` Is meant for CAT tools and is usable with a per-user subscription.

Anterior, Weblate a fost clasificat ca un instrument CAT de către DeepL, așa că trebuia să utilizeze API v1, dar acum trebuie să utilizeze API v2. Prin urmare, este implicit v2 și puteți să o schimbați la v1 în cazul în care aveți un abonament CAT existent și doriți ca Weblate să îl folosească.

The easiest way to find out which one to use is to open an URL like the following in your browser:

https://api.deepl.com/v2/translate?text=Hello&target_lang=FR&auth_key=XXX

Replace the XXX with your auth_key. If you receive a JSON object which contains „Bonjour”, you have the correct URL; if not, try the other three.

Vezi și:

DeepL, *Traducere automată*, *Sugestii automate*

2.16.67 MT_DEEPL_KEY

Cheia API pentru DeepL API, vă puteți înregistra la https://www.deepl.com/pro.html

Vezi și:

DeepL, *Traducere automată*, *Sugestii automate*

2.16.68 MT_LIBRETRANSLATE_API_URL

Nou în versiunea 4.7.1.

API URL for the LibreTranslate instance to use.

`https://libretranslate.com/` (official public instance) Requires an API key to use outside of the website.

Mirrors are documented on the LibreTranslate GitHub repository, some of which can be used without authentication:

https://github.com/LibreTranslate/LibreTranslate#user-content-mirrors

Vezi și:

LibreTranslate, *Traducere automată*, *Sugestii automate*

2.16.69 MT_LIBRETRANSLATE_KEY

Nou în versiunea 4.7.1.

API key for the LibreTranslate instance specified in *MT_LIBRETRANSLATE_API_URL*.

Vezi și:

LibreTranslate, *Traducere automată*, *Sugestii automate*

2.16.70 MT_GOOGLE_KEY

Cheie API pentru Google Translate API v2, vă puteți înregistra la https://cloud.google.com/translate/docs

Vezi și:

Google Translate, *Traducere automată*, *Sugestii automate*

2.16.71 MT_GOOGLE_CREDENTIALS

Fișier de credențiale API v3 JSON obținut în consola Google cloud. Vă rugăm să furnizați o cale completă a sistemului de operare. Datele de acreditare sunt pentru fiecare cont de serviciu afiliat la un anumit proiect. Vă rugăm să consultați https://cloud.google.com/docs/authentication/getting-started pentru mai multe detalii.

2.16.72 MT_GOOGLE_PROJECT

ID-ul proiectului Google Cloud API v3 cu serviciul de traducere activat și facturarea activată. Vă rugăm să verificați https://cloud.google.com/appengine/docs/standard/nodejs/building-app/creating-project pentru mai multe detalii

2.16.73 MT_GOOGLE_LOCATION

API v3 Google Cloud App Engine poate fi specifică unei locații. Schimbați în consecință dacă soluția de rezervă implicită `global` nu funcționează pentru dvs.

Vă rugăm să consultați https://cloud.google.com/appengine/docs/locations pentru mai multe detalii

Vezi și:

Google Translate API V3 (Advanced)

2.16.74 MT_MICROSOFT_BASE_URL

Domeniul URL-ului de bază al regiunii, așa cum este definit în secțiunea „URL-uri de bază”.

Valoarea implicită este `api.cognitive.microsofttranslator.com` pentru Azure Global.

Pentru Azure China, vă rugăm să utilizați `api.translator.azure.cn`.

2.16.75 MT_MICROSOFT_COGNITIVE_KEY

Cheia de client pentru Microsoft Cognitive Services Translator API.

Vezi și:

Microsoft Cognitive Services Translator, *Traducere automată*, *Sugestii automate*, Cognitive Services - Text Translation API, Microsoft Azure Portal

2.16.76 MT_MICROSOFT_REGION

Prefixul regiunii, așa cum este definit în secțiunea „Autentificarea cu o resursă multiserviciu”.

2.16.77 MT_MICROSOFT_ENDPOINT_URL

Region endpoint URL domain for access token as defined in the „Authenticating with an access token” section.

Defaults to `api.cognitive.microsoft.com` for Azure Global.

For Azure China, please use your endpoint from the Azure Portal.

2.16.78 MT_MODERNMT_KEY

API key for the ModernMT machine translation engine.

Vezi și:

ModernMT `MT_MODERNMT_URL`

2.16.79 MT_MODERNMT_URL

URL of ModernMT. It defaults to `https://api.modernmt.com/` for the cloud service.

Vezi și:

ModernMT `MT_MODERNMT_KEY`

2.16.80 MT_MYMEMORY_EMAIL

MyMemory identification e-mail address. It permits 1000 requests per day.

Vezi și:

MyMemory, *Traducere automată*, *Sugestii automate*, MyMemory: API technical specifications

2.16.81 MT_MYMEMORY_KEY

MyMemory access key for private translation memory, use it with `MT_MYMEMORY_USER`.

Vezi și:

MyMemory, *Traducere automată*, *Sugestii automate*, MyMemory: API key generator

2.16.82 MT_MYMEMORY_USER

MyMemory user ID for private translation memory, use it with `MT_MYMEMORY_KEY`.

Vezi și:

MyMemory, *Traducere automată*, *Sugestii automate*, MyMemory: API key generator

2.16.83 MT_NETEASE_KEY

App key for NetEase Sight API, you can register at https://sight.youdao.com/

Vezi și:

NetEase Sight API machine translation, *Traducere automată*, *Sugestii automate*

2.16.84 MT_NETEASE_SECRET

App secret for the NetEase Sight API, you can register at https://sight.youdao.com/

Vezi și:

NetEase Sight API machine translation, *Traducere automată*, *Sugestii automate*

2.16.85 MT_TMSERVER

URL where tmserver is running.

Vezi și:

tmserver, *Traducere automată*, *Sugestii automate*, tmserver

2.16.86 MT_YANDEX_KEY

API key for the Yandex Translate API, you can register at https://yandex.com/dev/translate/

Vezi și:

Yandex Translate, *Traducere automată*, *Sugestii automate*

2.16.87 MT_YOUDAO_ID

Client ID for the Youdao Zhiyun API, you can register at https://ai.youdao.com/product-fanyi-text.s.

Vezi și:

Youdao Zhiyun API machine translation, *Traducere automată*, *Sugestii automate*

2.16.88 MT_YOUDAO_SECRET

Client secret for the Youdao Zhiyun API, you can register at https://ai.youdao.com/product-fanyi-text.s.

Vezi și:

Youdao Zhiyun API machine translation, *Traducere automată*, *Sugestii automate*

2.16.89 MT_SAP_BASE_URL

API URL to the SAP Translation Hub service.

Vezi și:

SAP Translation Hub, *Traducere automată*, *Sugestii automate*

2.16.90 MT_SAP_SANDBOX_APIKEY

API key for sandbox API usage

Vezi și:

SAP Translation Hub, *Traducere automată*, *Sugestii automate*

2.16.91 MT_SAP_USERNAME

Your SAP username

Vezi și:

SAP Translation Hub, *Traducere automată*, *Sugestii automate*

2.16.92 MT_SAP_PASSWORD

Your SAP password

Vezi și:

SAP Translation Hub, *Traducere automată*, *Sugestii automate*

2.16.93 MT_SAP_USE_MT

Whether to also use machine translation services, in addition to the term database. Possible values: `True` or `False`

Vezi și:

SAP Translation Hub, *Traducere automată*, *Sugestii automate*

2.16.94 NEARBY_MESSAGES

How many strings to show around the currently translated string. This is just a default value, users can adjust this in *Profilul utilizatorului*.

2.16.95 DEFAULT_PAGE_LIMIT

Nou în versiunea 4.7.

Default number of elements to display when pagination is active.

2.16.96 PAGURE_CREDENTIALS

Nou în versiunea 4.3.2.

List for credentials for Pagure servers.

Sugestie: Use this in case you want Weblate to interact with more of them, for single Pagure endpoint stick with `PAGURE_USERNAME` and `PAGURE_TOKEN`.

```
PAGURE_CREDENTIALS = {
    "pagure.io": {
        "username": "weblate",
        "token": "your-api-token",
    },
    "pagure.example.com": {
        "username": "weblate",
        "token": "another-api-token",
    },
}
```

2.16.97 PAGURE_USERNAME

Nou în versiunea 4.3.2.

Pagure username used to send merge requests for translation updates.

Vezi și:

`PAGURE_CREDENTIALS`, *Pagure*

2.16.98 PAGURE_TOKEN

Nou în versiunea 4.3.2.

Pagure personal access token used to make API calls for translation updates.

Vezi și:

`PAGURE_CREDENTIALS`, *Pagure*, Pagure API

2.16.99 PRIVACY_URL

Nou în versiunea 4.8.1.

URL where your Weblate instance shows its privacy policy.

Sugestie: Utile dacă vă găzduiți documentele juridice în afara Weblate pentru a le încorpora în Weblate, vă rugăm să verificați *Legal* pentru detalii.

Exemplu:

```
PRIVACY_URL = "https://weblate.org/terms/"
```

Vezi și:

`LEGAL_URL`

2.16.100 RATELIMIT_ATTEMPTS

Nou în versiunea 3.2.

Maximum number of authentication attempts before rate limiting is applied.

Defaults to 5.

Vezi și:

Rate limiting, `RATELIMIT_WINDOW`, `RATELIMIT_LOCKOUT`

2.16.101 RATELIMIT_WINDOW

Nou în versiunea 3.2.

How long authentication is accepted after rate limiting applies.

An amount of seconds defaulting to 300 (5 minutes).

Vezi și:

Rate limiting, `RATELIMIT_ATTEMPTS`, `RATELIMIT_LOCKOUT`

2.16.102 RATELIMIT_LOCKOUT

Nou în versiunea 3.2.

How long authentication is locked after rate limiting applies.

An amount of seconds defaulting to 600 (10 minutes).

Vezi și:

Rate limiting, `RATELIMIT_ATTEMPTS`, `RATELIMIT_WINDOW`

2.16.103 REGISTRATION_ALLOW_BACKENDS

Nou în versiunea 4.1.

List of authentication backends to allow registration from. This only limits new registrations, users can still authenticate and add authentication using all configured authentication backends.

It is recommended to keep `REGISTRATION_OPEN` enabled while limiting registration backends, otherwise users will be able to register, but Weblate will not show links to register in the user interface.

Exemplu:

```
REGISTRATION_ALLOW_BACKENDS = ["azuread-oauth2", "azuread-tenant-oauth2"]
```

Sugestie: The backend names match names used in URL for authentication.

Vezi și:

`REGISTRATION_OPEN`, *Autentificare*

2.16.104 REGISTRATION_CAPTCHA

A value of either `True` or `False` indicating whether registration of new accounts is protected by CAPTCHA. This setting is optional, and a default of `True` will be assumed if it is not supplied.

If turned on, a CAPTCHA is added to all pages where a users enters their e-mail address:

- New account registration.
- Password recovery.
- Adding e-mail to an account.
- Contact form for users that are not signed in.

2.16.105 REGISTRATION_EMAIL_MATCH

Nou în versiunea 2.17.

Allows you to filter which e-mail addresses can register.

Defaults to `.*`, which allows any e-mail address to be registered.

You can use it to restrict registration to a single e-mail domain:

```
REGISTRATION_EMAIL_MATCH = r"^.*@weblate\.org$"
```

2.16.106 REGISTRATION_OPEN

Whether registration of new accounts is currently permitted. This optional setting can remain the default `True`, or changed to `False`.

This setting affects built-in authentication by e-mail address or through the Python Social Auth (you can whitelist certain back-ends using *REGISTRATION_ALLOW_BACKENDS*).

Notă: If using third-party authentication methods such as *Autentificare LDAP*, it just hides the registration form, but new users might still be able to sign in and create accounts.

Vezi și:

REGISTRATION_ALLOW_BACKENDS, *REGISTRATION_EMAIL_MATCH*, *Autentificare*

2.16.107 REPOSITORY_ALERT_THRESHOLD

Nou în versiunea 4.0.2.

Threshold for triggering an alert for outdated repositories, or ones that contain too many changes. Defaults to 25.

Vezi și:

alerts

2.16.108 REQUIRE_LOGIN

Nou în versiunea 4.1.

This enables *LOGIN_REQUIRED_URLS* and configures REST framework to require authentication for all API endpoints.

Notă: This is implemented in the *Sample configuration*. For Docker, use *WEBLATE_REQUIRE_LOGIN*.

2.16.109 SENTRY_DSN

Nou în versiunea 3.9.

Sentry DSN to use for *Collecting error reports*.

Vezi și:

Django integration for Sentry

2.16.110 SESSION_COOKIE_AGE_AUTHENTICATED

Nou în versiunea 4.3.

Set session expiry for authenticated users. This complements `SESSION_COOKIE_AGE` which is used for unauthenticated users.

Vezi și:

`SESSION_COOKIE_AGE`

2.16.111 SIMPLIFY_LANGUAGES

Use simple language codes for default language/country combinations. For example an `fr_FR` translation will use the `fr` language code. This is usually the desired behavior, as it simplifies listing languages for these default combinations.

Turn this off if you want to different translations for each variant.

2.16.112 SITE_DOMAIN

Configures site domain. This is necessary to produce correct absolute links in many scopes (for example activation e-mails, notifications or RSS feeds).

In case Weblate is running on non-standard port, include it here as well.

Examples:

```
# Production site with domain name
SITE_DOMAIN = "weblate.example.com"

# Local development with IP address and port
SITE_DOMAIN = "127.0.0.1:8000"
```

Notă: This setting should only contain the domain name. For configuring protocol, (enabling and enforcing HTTPS) use *ENABLE_HTTPS* and for changing URL, use *URL_PREFIX*.

Sugestie: On a Docker container, the site domain is configured through *WEBLATE_ALLOWED_HOSTS*.

Vezi și:

Set correct site domain, *Allowed hosts setup*, *Correctly configure HTTPS* *WEBLATE_SITE_DOMAIN*, *ENABLE_HTTPS*

2.16.113 SITE_TITLE

Site title to be used for the website and sent e-mails.

2.16.114 SPECIAL_CHARS

Additional characters to include in the visual keyboard, *Visual keyboard*.

The default value is:

```
SPECIAL_CHARS = ("\t", "\n", "\u00a0", "…")
```

2.16.115 SINGLE_PROJECT

Nou în versiunea 3.8.

Redirects users directly to a project or component instead of showing the dashboard. You can either set it to `True` and in this case it only works in case there is actually only single project in Weblate. Alternatively set the project slug, and it will redirect unconditionally to this project.

Schimbat în versiunea 3.11: The setting now also accepts a project slug, to force displaying that single project.

Exemplu:

```
SINGLE_PROJECT = "test"
```

2.16.116 STATUS_URL

The URL where your Weblate instance reports its status.

2.16.117 SUGGESTION_CLEANUP_DAYS

Nou în versiunea 3.2.1.

Automatically deletes suggestions after a given number of days. Defaults to `None`, meaning no deletions.

2.16.118 UPDATE_LANGUAGES

Nou în versiunea 4.3.2.

Controls whether languages database should be updated when running database migration and is enabled by default. This setting has no effect on invocation of *`setuplang`*.

Vezi și:

Built-in language definitions

2.16.119 URL_PREFIX

This setting allows you to run Weblate under some path (otherwise it relies on being run from the webserver root).

Notă: To use this setting, you also need to configure your server to strip this prefix. For example with WSGI, this can be achieved by setting `WSGIScriptAlias`.

Sugestie: The prefix should start with a `/`.

Exemplu:

```
URL_PREFIX = "/translations"
```

Notă: This setting does not work with Django's built-in server, you would have to adjust `urls.py` to contain this prefix.

2.16.120 VCS_BACKENDS

Configuration of available VCS backends.

Notă: Weblate tries to use all supported back-ends you have the tools for.

Sugestie: You can limit choices or add custom VCS back-ends by using this.

```
VCS_BACKENDS = ("weblate.vcs.git.GitRepository",)
```

Vezi și:

Integrarea controlului versiunilor

2.16.121 VCS_CLONE_DEPTH

Nou în versiunea 3.10.2.

Configures how deep cloning of repositories Weblate should do.

Notă: Currently this is only supported in *Git*. By default Weblate does shallow clones of the repositories to make cloning faster and save disk space. Depending on your usage (for example when using custom *Add-ons*), you might want to increase the depth or turn off shallow clones completely by setting this to 0.

Sugestie: In case you get `fatal: protocol error: expected old/new/ref, got 'shallow <commit hash>'` error when pushing from Weblate, turn off shallow clones completely by setting:

```
VCS_CLONE_DEPTH = 0
```

2.16.122 WEBLATE_ADDONS

List of addons available for use. To use them, they have to be enabled for a given translation component. By default this includes all built-in addons, when extending the list you will probably want to keep existing ones enabled, for example:

```
WEBLATE_ADDONS = (
    # Built-in addons
    "weblate.addons.gettext.GenerateMoAddon",
    "weblate.addons.gettext.UpdateLinguasAddon",
    "weblate.addons.gettext.UpdateConfigureAddon",
    "weblate.addons.gettext.MsgmergeAddon",
    "weblate.addons.gettext.GettextCustomizeAddon",
    "weblate.addons.gettext.GettextAuthorComments",
    "weblate.addons.cleanup.CleanupAddon",
    "weblate.addons.consistency.LangaugeConsistencyAddon",
    "weblate.addons.discovery.DiscoveryAddon",
    "weblate.addons.flags.SourceEditAddon",
    "weblate.addons.flags.TargetEditAddon",
    "weblate.addons.flags.SameEditAddon",
    "weblate.addons.flags.BulkEditAddon",
    "weblate.addons.generate.GenerateFileAddon",
    "weblate.addons.json.JSONCustomizeAddon",
    "weblate.addons.properties.PropertiesSortAddon",
    "weblate.addons.git.GitSquashAddon",
    "weblate.addons.removal.RemoveComments",
    "weblate.addons.removal.RemoveSuggestions",
    "weblate.addons.resx.ResxUpdateAddon",
    "weblate.addons.autotranslate.AutoTranslateAddon",
    "weblate.addons.yaml.YAMLCustomizeAddon",
    "weblate.addons.cdn.CDNJSAddon",
    # Add-on you want to include
    "weblate.addons.example.ExampleAddon",
)
```

Notă: Removing the addon from the list does not uninstall it from the components. Weblate will crash in that case. Please uninstall addon from all components prior to removing it from this list.

Vezi și:

Add-ons, `DEFAULT_ADDONS`

2.16.123 WEBLATE_EXPORTERS

Nou în versiunea 4.2.

List of a available exporters offering downloading translations or glossaries in various file formats.

Vezi și:

Supported file formats

2.16.124 WEBLATE_FORMATS

Nou în versiunea 3.0.

List of file formats available for use.

Notă: The default list already has the common formats.

Vezi și:

Supported file formats

2.16.125 WEBLATE_GPG_IDENTITY

Nou în versiunea 3.1.

Identity used by Weblate to sign Git commits, for example:

```
WEBLATE_GPG_IDENTITY = "Weblate <weblate@example.com>"
```

The Weblate GPG keyring is searched for a matching key (`home/.gnupg` under `DATA_DIR`). If not found, a key is generated, please check *Signing Git commits with GnuPG* for more details.

Vezi și:

Signing Git commits with GnuPG

2.16.126 WEBSITE_REQUIRED

Defines whether *Website-ul proiectului* has to be specified when creating a project. Turned on by default as that suits public server setups.

2.17 Sample configuration

The following example is shipped as `weblate/settings_example.py` with Weblate:

```
#
# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#


import os
import platform
from logging.handlers import SysLogHandler

# Title of site to use
SITE_TITLE = "Weblate"

# Site domain
SITE_DOMAIN = ""

# Whether site uses https
ENABLE_HTTPS = False

#
# Django settings for Weblate project.
#

DEBUG = True

ADMINS = (
    # ("Your Name", "your_email@example.com"),
)

MANAGERS = ADMINS

DATABASES = {
    "default": {
        # Use "postgresql" or "mysql".
        "ENGINE": "django.db.backends.postgresql",
        # Database name.
        "NAME": "weblate",
        # Database user.
        "USER": "weblate",
        # Name of role to alter to set parameters in PostgreSQL,
        # use in case role name is different than user used for authentication.
        # "ALTER_ROLE": "weblate",
        # Database password.
        "PASSWORD": "",
```

(continues on next page)

(continuare din pagina precedentă)

```
        # Set to empty string for localhost.
        "HOST": "127.0.0.1",
        # Set to empty string for default.
        "PORT": "",
        # Customizations for databases.
        "OPTIONS": {
            # In case of using an older MySQL server,
            # which has MyISAM as a default storage
            # "init_command": "SET storage_engine=INNODB",
            # Uncomment for MySQL older than 5.7:
            # "init_command": "SET sql_mode='STRICT_TRANS_TABLES'",
            # Set emoji capable charset for MySQL:
            # "charset": "utf8mb4",
            # Change connection timeout in case you get MySQL gone away error:
            # "connect_timeout": 28800,
        },
        # Persistent connections
        "CONN_MAX_AGE": 0,
    }
}

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Data directory
DATA_DIR = os.path.join(BASE_DIR, "data")

# Local time zone for this installation. Choices can be found here:
# http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
# although not all choices may be available on all operating systems.
# In a Windows environment this must be set to your system time zone.
TIME_ZONE = "UTC"

# Language code for this installation. All choices can be found here:
# http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = "en-us"

LANGUAGES = (
    ("ar", "العربية"),
    ("az", "Azərbaycan"),
    ("be", "Беларуская"),
    ("be@latin", "Biełaruskaja"),
    ("bg", "Български"),
    ("br", "Brezhoneg"),
    ("ca", "Català"),
    ("cs", "Čeština"),
    ("da", "Dansk"),
    ("de", "Deutsch"),
    ("en", "English"),
    ("el", "Ελληνικά"),
    ("en-gb", "English (United Kingdom)"),
    ("es", "Español"),
    ("fi", "Suomi"),
    ("fr", "Français"),
    ("gl", "Galego"),
    ("he", "עברית"),
    ("hu", "Magyar"),
    ("hr", "Hrvatski"),
    ("id", "Indonesia"),
    ("is", "Íslenska"),
    ("it", "Italiano"),
    ("ja", "□□□"),
```

(continues on next page)

(continuare din pagina precedentă)

```
    ("kab", "Taqbaylit"),
    ("kk", "Қазақ тілі"),
    ("ko", "□□□"),
    ("nb", "Norsk bokmål"),
    ("nl", "Nederlands"),
    ("pl", "Polski"),
    ("pt", "Português"),
    ("pt-br", "Português brasileiro"),
    ("ro", "Română"),
    ("ru", "Русский"),
    ("sk", "Slovenčina"),
    ("sl", "Slovenščina"),
    ("sq", "Shqip"),
    ("sr", "Српски"),
    ("sr-latn", "Srpski"),
    ("sv", "Svenska"),
    ("th", "□□□"),
    ("tr", "Türkçe"),
    ("uk", "Українська"),
    ("zh-hans", "□□□"),
    ("zh-hant", "□□□"),
)

SITE_ID = 1

# If you set this to False, Django will make some optimizations so as not
# to load the internationalization machinery.
USE_I18N = True

# If you set this to False, Django will not format dates, numbers and
# calendars according to the current locale.
USE_L10N = True

# If you set this to False, Django will not use timezone-aware datetimes.
USE_TZ = True

# Type of automatic primary key, introduced in Django 3.2
DEFAULT_AUTO_FIELD = "django.db.models.AutoField"

# URL prefix to use, please see documentation for more details
URL_PREFIX = ""

# Absolute filesystem path to the directory that will hold user-uploaded files.
MEDIA_ROOT = os.path.join(DATA_DIR, "media")

# URL that handles the media served from MEDIA_ROOT. Make sure to use a
# trailing slash.
MEDIA_URL = f"{URL_PREFIX}/media/"

# Absolute path to the directory static files should be collected to.
# Don't put anything in this directory yourself; store your static files
# in apps' "static/" subdirectories and in STATICFILES_DIRS.
STATIC_ROOT = os.path.join(DATA_DIR, "static")

# URL prefix for static files.
STATIC_URL = f"{URL_PREFIX}/static/"

# Additional locations of static files
STATICFILES_DIRS = (
    # Put strings here, like "/home/html/static" or "C:/www/django/static".
    # Always use forward slashes, even on Windows.
```

(continues on next page)

(continuare din pagina precedentă)

```
    # Don't forget to use absolute paths, not relative paths.
)

# List of finder classes that know how to find static files in
# various locations.
STATICFILES_FINDERS = (
    "django.contrib.staticfiles.finders.FileSystemFinder",
    "django.contrib.staticfiles.finders.AppDirectoriesFinder",
    "compressor.finders.CompressorFinder",
)

# Make this unique, and don't share it with anybody.
# You can generate it using weblate/examples/generate-secret-key
SECRET_KEY = ""

_TEMPLATE_LOADERS = [
    "django.template.loaders.filesystem.Loader",
    "django.template.loaders.app_directories.Loader",
]
if not DEBUG:
    _TEMPLATE_LOADERS = [("django.template.loaders.cached.Loader", _TEMPLATE_
↪LOADERS)]
TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "OPTIONS": {
            "context_processors": [
                "django.contrib.auth.context_processors.auth",
                "django.template.context_processors.debug",
                "django.template.context_processors.i18n",
                "django.template.context_processors.request",
                "django.template.context_processors.csrf",
                "django.contrib.messages.context_processors.messages",
                "weblate.trans.context_processors.weblate_context",
            ],
            "loaders": _TEMPLATE_LOADERS,
        },
    }
]


# GitHub username and token for sending pull requests.
# Please see the documentation for more details.
GITHUB_USERNAME = None
GITHUB_TOKEN = None

# GitLab username and token for sending merge requests.
# Please see the documentation for more details.
GITLAB_USERNAME = None
GITLAB_TOKEN = None

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.email.EmailAuth",
    # "social_core.backends.google.GoogleOAuth2",
    # "social_core.backends.github.GithubOAuth2",
    # "social_core.backends.bitbucket.BitbucketOAuth",
    # "social_core.backends.suse.OpenSUSEOpenId",
    # "social_core.backends.ubuntu.UbuntuOpenId",
    # "social_core.backends.fedora.FedoraOpenId",
    # "social_core.backends.facebook.FacebookOAuth2",
```

(continues on next page)

(continuare din pagina precedentă)

```
    "weblate.accounts.auth.WeblateUserBackend",
)

# Custom user model
AUTH_USER_MODEL = "weblate_auth.User"

# Social auth backends setup
SOCIAL_AUTH_GITHUB_KEY = ""
SOCIAL_AUTH_GITHUB_SECRET = ""
SOCIAL_AUTH_GITHUB_SCOPE = ["user:email"]

SOCIAL_AUTH_BITBUCKET_KEY = ""
SOCIAL_AUTH_BITBUCKET_SECRET = ""
SOCIAL_AUTH_BITBUCKET_VERIFIED_EMAILS_ONLY = True

SOCIAL_AUTH_FACEBOOK_KEY = ""
SOCIAL_AUTH_FACEBOOK_SECRET = ""
SOCIAL_AUTH_FACEBOOK_SCOPE = ["email", "public_profile"]
SOCIAL_AUTH_FACEBOOK_PROFILE_EXTRA_PARAMS = {"fields": "id,name,email"}

SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = ""
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = ""

# Social auth settings
SOCIAL_AUTH_PIPELINE = (
    "social_core.pipeline.social_auth.social_details",
    "social_core.pipeline.social_auth.social_uid",
    "social_core.pipeline.social_auth.auth_allowed",
    "social_core.pipeline.social_auth.social_user",
    "weblate.accounts.pipeline.store_params",
    "weblate.accounts.pipeline.verify_open",
    "social_core.pipeline.user.get_username",
    "weblate.accounts.pipeline.require_email",
    "social_core.pipeline.mail.mail_validation",
    "weblate.accounts.pipeline.revoke_mail_code",
    "weblate.accounts.pipeline.ensure_valid",
    "weblate.accounts.pipeline.remove_account",
    "social_core.pipeline.social_auth.associate_by_email",
    "weblate.accounts.pipeline.reauthenticate",
    "weblate.accounts.pipeline.verify_username",
    "social_core.pipeline.user.create_user",
    "social_core.pipeline.social_auth.associate_user",
    "social_core.pipeline.social_auth.load_extra_data",
    "weblate.accounts.pipeline.cleanup_next",
    "weblate.accounts.pipeline.user_full_name",
    "weblate.accounts.pipeline.store_email",
    "weblate.accounts.pipeline.notify_connect",
    "weblate.accounts.pipeline.password_reset",
)
SOCIAL_AUTH_DISCONNECT_PIPELINE = (
    "social_core.pipeline.disconnect.allowed_to_disconnect",
    "social_core.pipeline.disconnect.get_entries",
    "social_core.pipeline.disconnect.revoke_tokens",
    "weblate.accounts.pipeline.cycle_session",
    "weblate.accounts.pipeline.adjust_primary_mail",
    "weblate.accounts.pipeline.notify_disconnect",
    "social_core.pipeline.disconnect.disconnect",
    "weblate.accounts.pipeline.cleanup_next",
)

# Custom authentication strategy
```

(continues on next page)

(continuare din pagina precedentă)

```
SOCIAL_AUTH_STRATEGY = "weblate.accounts.strategy.WeblateStrategy"

# Raise exceptions so that we can handle them later
SOCIAL_AUTH_RAISE_EXCEPTIONS = True

SOCIAL_AUTH_EMAIL_VALIDATION_FUNCTION = "weblate.accounts.pipeline.send_validation"
SOCIAL_AUTH_EMAIL_VALIDATION_URL = f"{URL_PREFIX}/accounts/email-sent/"
SOCIAL_AUTH_LOGIN_ERROR_URL = f"{URL_PREFIX}/accounts/login/"
SOCIAL_AUTH_EMAIL_FORM_URL = f"{URL_PREFIX}/accounts/email/"
SOCIAL_AUTH_NEW_ASSOCIATION_REDIRECT_URL = f"{URL_PREFIX}/accounts/profile/#account
↪"
SOCIAL_AUTH_PROTECTED_USER_FIELDS = ("email",)
SOCIAL_AUTH_SLUGIFY_USERNAMES = True
SOCIAL_AUTH_SLUGIFY_FUNCTION = "weblate.accounts.pipeline.slugify_username"

# Password validation configuration
AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME": "django.contrib.auth.password_validation.
↪UserAttributeSimilarityValidator"  # noqa: E501, pylint: disable=line-too-long
    },
    {
        "NAME": "django.contrib.auth.password_validation.MinimumLengthValidator",
        "OPTIONS": {"min_length": 10},
    },
    {"NAME": "django.contrib.auth.password_validation.CommonPasswordValidator"},
    {"NAME": "django.contrib.auth.password_validation.NumericPasswordValidator"},
    {"NAME": "weblate.accounts.password_validation.CharsPasswordValidator"},
    {"NAME": "weblate.accounts.password_validation.PastPasswordsValidator"},
    # Optional password strength validation by django-zxcvbn-password
    # {
    #     "NAME": "zxcvbn_password.ZXCVBNValidator",
    #     "OPTIONS": {
    #         "min_score": 3,
    #         "user_attributes": ("username", "email", "full_name")
    #     }
    # },
]

# Password hashing (prefer Argon)
PASSWORD_HASHERS = [
    "django.contrib.auth.hashers.Argon2PasswordHasher",
    "django.contrib.auth.hashers.PBKDF2PasswordHasher",
    "django.contrib.auth.hashers.PBKDF2SHA1PasswordHasher",
    "django.contrib.auth.hashers.BCryptSHA256PasswordHasher",
]

# Allow new user registrations
REGISTRATION_OPEN = True

# Shortcut for login required setting
REQUIRE_LOGIN = False

# Middleware
MIDDLEWARE = [
    "weblate.middleware.RedirectMiddleware",
    "weblate.middleware.ProxyMiddleware",
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "weblate.accounts.middleware.AuthenticationMiddleware",
```

(continues on next page)

(continuare din pagina precedentă)

```
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
    "social_django.middleware.SocialAuthExceptionMiddleware",
    "weblate.accounts.middleware.RequireLoginMiddleware",
    "weblate.api.middleware.ThrottlingMiddleware",
    "weblate.middleware.SecurityMiddleware",
    "weblate.wladmin.middleware.ManageMiddleware",
]

ROOT_URLCONF = "weblate.urls"

# Django and Weblate apps
INSTALLED_APPS = [
    # Weblate apps on top to override Django locales and templates
    "weblate.addons",
    "weblate.auth",
    "weblate.checks",
    "weblate.formats",
    "weblate.glossary",
    "weblate.machinery",
    "weblate.trans",
    "weblate.lang",
    "weblate_language_data",
    "weblate.memory",
    "weblate.screenshots",
    "weblate.fonts",
    "weblate.accounts",
    "weblate.configuration",
    "weblate.utils",
    "weblate.vcs",
    "weblate.wladmin",
    "weblate.metrics",
    "weblate",
    # Optional: Git exporter
    "weblate.gitexport",
    # Standard Django modules
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "django.contrib.admin.apps.SimpleAdminConfig",
    "django.contrib.admindocs",
    "django.contrib.sitemaps",
    "django.contrib.humanize",
    # Third party Django modules
    "social_django",
    "crispy_forms",
    "compressor",
    "rest_framework",
    "rest_framework.authtoken",
    "django_filters",
]

# Custom exception reporter to include some details
DEFAULT_EXCEPTION_REPORTER_FILTER = "weblate.trans.debug.
↪WeblateExceptionReporterFilter"

# Default logging of Weblate messages
# - to syslog in production (if available)
# - otherwise to console
```

(continues on next page)

(continuare din pagina precedentă)

```
# - you can also choose "logfile" to log into separate file
#   after configuring it below

# Detect if we can connect to syslog
HAVE_SYSLOG = False
if platform.system() != "Windows":
    try:
        handler = SysLogHandler(address="/dev/log", facility=SysLogHandler.LOG_
↪LOCAL2)
        handler.close()
        HAVE_SYSLOG = True
    except OSError:
        HAVE_SYSLOG = False

if DEBUG or not HAVE_SYSLOG:
    DEFAULT_LOG = "console"
else:
    DEFAULT_LOG = "syslog"
DEFAULT_LOGLEVEL = "DEBUG" if DEBUG else "INFO"

# A sample logging configuration. The only tangible logging
# performed by this configuration is to send an email to
# the site admins on every HTTP 500 error when DEBUG=False.
# See http://docs.djangoproject.com/en/stable/topics/logging for
# more details on how to customize your logging configuration.
LOGGING = {
    "version": 1,
    "disable_existing_loggers": True,
    "filters": {"require_debug_false": {"()": "django.utils.log.RequireDebugFalse"}
↪},
    "formatters": {
        "syslog": {"format": "weblate[%(process)d]: %(levelname)s %(message)s"},
        "simple": {"format": "[%(asctime)s: %(levelname)s/%(process)s] %(message)s
↪"},
        "logfile": {"format": "%(asctime)s %(levelname)s %(message)s"},
        "django.server": {
            "()": "django.utils.log.ServerFormatter",
            "format": "[%(server_time)s] %(message)s",
        },
    },
    "handlers": {
        "mail_admins": {
            "level": "ERROR",
            "filters": ["require_debug_false"],
            "class": "django.utils.log.AdminEmailHandler",
            "include_html": True,
        },
        "console": {
            "level": "DEBUG",
            "class": "logging.StreamHandler",
            "formatter": "simple",
        },
        "django.server": {
            "level": "INFO",
            "class": "logging.StreamHandler",
            "formatter": "django.server",
        },
        "syslog": {
            "level": "DEBUG",
            "class": "logging.handlers.SysLogHandler",
            "formatter": "syslog",
```

(continues on next page)

(continuare din pagina precedentă)

```
            "address": "/dev/log",
            "facility": SysLogHandler.LOG_LOCAL2,
        },
        # Logging to a file
        # "logfile": {
        #     "level":"DEBUG",
        #     "class":"logging.handlers.RotatingFileHandler",
        #     "filename": "/var/log/weblate/weblate.log",
        #     "maxBytes": 100000,
        #     "backupCount": 3,
        #     "formatter": "logfile",
        # },
    },
    "loggers": {
        "django.request": {
            "handlers": ["mail_admins", DEFAULT_LOG],
            "level": "ERROR",
            "propagate": True,
        },
        "django.server": {
            "handlers": ["django.server"],
            "level": "INFO",
            "propagate": False,
        },
        # Logging database queries
        # "django.db.backends": {
        #     "handlers": [DEFAULT_LOG],
        #     "level": "DEBUG",
        # },
        "weblate": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
        # Logging VCS operations
        "weblate.vcs": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
        # Python Social Auth
        "social": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
        # Django Authentication Using LDAP
        "django_auth_ldap": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
        # SAML IdP
        "djangosaml2idp": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
    },
}

# Remove syslog setup if it's not present
if not HAVE_SYSLOG:
    del LOGGING["handlers"]["syslog"]

# List of machine translations
MT_SERVICES = (
    #     "weblate.machinery.apertium.ApertiumAPYTranslation",
    #     "weblate.machinery.baidu.BaiduTranslation",
    #     "weblate.machinery.deepl.DeepLTranslation",
    #     "weblate.machinery.glosbe.GlosbeTranslation",
    #     "weblate.machinery.google.GoogleTranslation",
    #     "weblate.machinery.googlev3.GoogleV3Translation",
    #     "weblate.machinery.libretranslate.LibreTranslateTranslation",
    #     "weblate.machinery.microsoft.MicrosoftCognitiveTranslation",
    #     "weblate.machinery.microsoftterminology.MicrosoftTerminologyService",
    #     "weblate.machinery.modernmt.ModernMTTranslation",
    #     "weblate.machinery.mymemory.MyMemoryTranslation",
    #     "weblate.machinery.netease.NeteaseSightTranslation",
    #     "weblate.machinery.tmserver.AmagamaTranslation",
    #     "weblate.machinery.tmserver.TMServerTranslation",
```

(continues on next page)

(continuare din pagina precedentă)

```
    #     "weblate.machinery.yandex.YandexTranslation",
    #     "weblate.machinery.saptranslationhub.SAPTranslationHub",
    #     "weblate.machinery.youdao.YoudaoTranslation",
    "weblate.machinery.weblatetm.WeblateTranslation",
    "weblate.memory.machine.WeblateMemory",
)

# Machine translation API keys

# URL of the Apertium APy server
MT_APERTIUM_APY = None

# DeepL API key
MT_DEEPL_KEY = None

# LibreTranslate
MT_LIBRETRANSLATE_API_URL = None
MT_LIBRETRANSLATE_KEY = None

# Microsoft Cognitive Services Translator API, register at
# https://portal.azure.com/
MT_MICROSOFT_COGNITIVE_KEY = None
MT_MICROSOFT_REGION = None

# ModernMT
MT_MODERNMT_KEY = None

# MyMemory identification email, see
# https://mymemory.translated.net/doc/spec.php
MT_MYMEMORY_EMAIL = None

# Optional MyMemory credentials to access private translation memory
MT_MYMEMORY_USER = None
MT_MYMEMORY_KEY = None

# Google API key for Google Translate API v2
MT_GOOGLE_KEY = None

# Google Translate API3 credentials and project id
MT_GOOGLE_CREDENTIALS = None
MT_GOOGLE_PROJECT = None

# Baidu app key and secret
MT_BAIDU_ID = None
MT_BAIDU_SECRET = None

# Youdao Zhiyun app key and secret
MT_YOUDAO_ID = None
MT_YOUDAO_SECRET = None

# Netease Sight (Jianwai) app key and secret
MT_NETEASE_KEY = None
MT_NETEASE_SECRET = None

# API key for Yandex Translate API
MT_YANDEX_KEY = None

# tmserver URL
MT_TMSERVER = None

# SAP Translation Hub
```

(continues on next page)

(continuare din pagina precedentă)

```
MT_SAP_BASE_URL = None
MT_SAP_SANDBOX_APIKEY = None
MT_SAP_USERNAME = None
MT_SAP_PASSWORD = None
MT_SAP_USE_MT = True

# Use HTTPS when creating redirect URLs for social authentication, see
# documentation for more details:
# https://python-social-auth-docs.readthedocs.io/en/latest/configuration/settings.
↪html#processing-redirects-and-urlopen
SOCIAL_AUTH_REDIRECT_IS_HTTPS = ENABLE_HTTPS

# Make CSRF cookie HttpOnly, see documentation for more details:
# https://docs.djangoproject.com/en/1.11/ref/settings/#csrf-cookie-httponly
CSRF_COOKIE_HTTPONLY = True
CSRF_COOKIE_SECURE = ENABLE_HTTPS
# Store CSRF token in session
CSRF_USE_SESSIONS = True
# Customize CSRF failure view
CSRF_FAILURE_VIEW = "weblate.trans.views.error.csrf_failure"
SESSION_COOKIE_SECURE = ENABLE_HTTPS
SESSION_COOKIE_HTTPONLY = True
# SSL redirect
SECURE_SSL_REDIRECT = ENABLE_HTTPS
# Sent referrrer only for same origin links
SECURE_REFERRER_POLICY = "same-origin"
# SSL redirect URL exemption list
SECURE_REDIRECT_EXEMPT = (r"healthz/$",)  # Allowing HTTP access to health check
# Session cookie age (in seconds)
SESSION_COOKIE_AGE = 1000
SESSION_COOKIE_AGE_AUTHENTICATED = 1209600
SESSION_COOKIE_SAMESITE = "Lax"
# Increase allowed upload size
DATA_UPLOAD_MAX_MEMORY_SIZE = 50000000

# Apply session coookie settings to language cookie as ewll
LANGUAGE_COOKIE_SECURE = SESSION_COOKIE_SECURE
LANGUAGE_COOKIE_HTTPONLY = SESSION_COOKIE_HTTPONLY
LANGUAGE_COOKIE_AGE = SESSION_COOKIE_AGE_AUTHENTICATED * 10
LANGUAGE_COOKIE_SAMESITE = SESSION_COOKIE_SAMESITE

# Some security headers
SECURE_BROWSER_XSS_FILTER = True
X_FRAME_OPTIONS = "DENY"
SECURE_CONTENT_TYPE_NOSNIFF = True

# Optionally enable HSTS
SECURE_HSTS_SECONDS = 31536000 if ENABLE_HTTPS else 0
SECURE_HSTS_PRELOAD = ENABLE_HTTPS
SECURE_HSTS_INCLUDE_SUBDOMAINS = ENABLE_HTTPS

# HTTPS detection behind reverse proxy
SECURE_PROXY_SSL_HEADER = None

# URL of login
LOGIN_URL = f"{URL_PREFIX}/accounts/login/"

# URL of logout
LOGOUT_URL = f"{URL_PREFIX}/accounts/logout/"

# Default location for login
```

(continues on next page)

(continuare din pagina precedentă)

```
LOGIN_REDIRECT_URL = f"{URL_PREFIX}/"

# Anonymous user name
ANONYMOUS_USER_NAME = "anonymous"

# Reverse proxy settings
IP_PROXY_HEADER = "HTTP_X_FORWARDED_FOR"
IP_BEHIND_REVERSE_PROXY = False
IP_PROXY_OFFSET = 0

# Sending HTML in mails
EMAIL_SEND_HTML = True

# Subject of emails includes site title
EMAIL_SUBJECT_PREFIX = f"[{SITE_TITLE}] "

# Enable remote hooks
ENABLE_HOOKS = True

# By default the length of a given translation is limited to the length of
# the source string * 10 characters. Set this option to False to allow longer
# translations (up to 10.000 characters)
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH = True

# Use simple language codes for default language/country combinations
SIMPLIFY_LANGUAGES = True

# Render forms using bootstrap
CRISPY_TEMPLATE_PACK = "bootstrap3"

# List of quality checks
# CHECK_LIST = (
#     "weblate.checks.same.SameCheck",
#     "weblate.checks.chars.BeginNewlineCheck",
#     "weblate.checks.chars.EndNewlineCheck",
#     "weblate.checks.chars.BeginSpaceCheck",
#     "weblate.checks.chars.EndSpaceCheck",
#     "weblate.checks.chars.DoubleSpaceCheck",
#     "weblate.checks.chars.EndStopCheck",
#     "weblate.checks.chars.EndColonCheck",
#     "weblate.checks.chars.EndQuestionCheck",
#     "weblate.checks.chars.EndExclamationCheck",
#     "weblate.checks.chars.EndEllipsisCheck",
#     "weblate.checks.chars.EndSemicolonCheck",
#     "weblate.checks.chars.MaxLengthCheck",
#     "weblate.checks.chars.KashidaCheck",
#     "weblate.checks.chars.PunctuationSpacingCheck",
#     "weblate.checks.format.PythonFormatCheck",
#     "weblate.checks.format.PythonBraceFormatCheck",
#     "weblate.checks.format.PHPFormatCheck",
#     "weblate.checks.format.CFormatCheck",
#     "weblate.checks.format.PerlFormatCheck",
#     "weblate.checks.format.JavaScriptFormatCheck",
#     "weblate.checks.format.LuaFormatCheck",
#     "weblate.checks.format.ObjectPascalFormatCheck",
#     "weblate.checks.format.SchemeFormatCheck",
#     "weblate.checks.format.CSharpFormatCheck",
#     "weblate.checks.format.JavaFormatCheck",
#     "weblate.checks.format.JavaMessageFormatCheck",
#     "weblate.checks.format.PercentPlaceholdersCheck",
#     "weblate.checks.format.VueFormattingCheck",
```

(continues on next page)

(continuare din pagina precedentă)

```
#     "weblate.checks.format.I18NextInterpolationCheck",
#     "weblate.checks.format.ESTemplateLiteralsCheck",
#     "weblate.checks.angularjs.AngularJSInterpolationCheck",
#     "weblate.checks.qt.QtFormatCheck",
#     "weblate.checks.qt.QtPluralCheck",
#     "weblate.checks.ruby.RubyFormatCheck",
#     "weblate.checks.consistency.PluralsCheck",
#     "weblate.checks.consistency.SamePluralsCheck",
#     "weblate.checks.consistency.ConsistencyCheck",
#     "weblate.checks.consistency.TranslatedCheck",
#     "weblate.checks.chars.EscapedNewlineCountingCheck",
#     "weblate.checks.chars.NewLineCountCheck",
#     "weblate.checks.markup.BBCodeCheck",
#     "weblate.checks.chars.ZeroWidthSpaceCheck",
#     "weblate.checks.render.MaxSizeCheck",
#     "weblate.checks.markup.XMLValidityCheck",
#     "weblate.checks.markup.XMLTagsCheck",
#     "weblate.checks.markup.MarkdownRefLinkCheck",
#     "weblate.checks.markup.MarkdownLinkCheck",
#     "weblate.checks.markup.MarkdownSyntaxCheck",
#     "weblate.checks.markup.URLCheck",
#     "weblate.checks.markup.SafeHTMLCheck",
#     "weblate.checks.placeholders.PlaceholderCheck",
#     "weblate.checks.placeholders.RegexCheck",
#     "weblate.checks.duplicate.DuplicateCheck",
#     "weblate.checks.source.OptionalPluralCheck",
#     "weblate.checks.source.EllipsisCheck",
#     "weblate.checks.source.MultipleFailingCheck",
#     "weblate.checks.source.LongUntranslatedCheck",
#     "weblate.checks.format.MultipleUnnamedFormatsCheck",
#     "weblate.checks.glossary.GlossaryCheck",
# )

# List of automatic fixups
# AUTOFIX_LIST = (
#     "weblate.trans.autofixes.whitespace.SameBookendingWhitespace",
#     "weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis",
#     "weblate.trans.autofixes.chars.RemoveZeroSpace",
#     "weblate.trans.autofixes.chars.RemoveControlChars",
# )

# List of enabled addons
# WEBLATE_ADDONS = (
#     "weblate.addons.gettext.GenerateMoAddon",
#     "weblate.addons.gettext.UpdateLinguasAddon",
#     "weblate.addons.gettext.UpdateConfigureAddon",
#     "weblate.addons.gettext.MsgmergeAddon",
#     "weblate.addons.gettext.GettextCustomizeAddon",
#     "weblate.addons.gettext.GettextAuthorComments",
#     "weblate.addons.cleanup.CleanupAddon",
#     "weblate.addons.cleanup.RemoveBlankAddon",
#     "weblate.addons.consistency.LangaugeConsistencyAddon",
#     "weblate.addons.discovery.DiscoveryAddon",
#     "weblate.addons.autotranslate.AutoTranslateAddon",
#     "weblate.addons.flags.SourceEditAddon",
#     "weblate.addons.flags.TargetEditAddon",
#     "weblate.addons.flags.SameEditAddon",
#     "weblate.addons.flags.BulkEditAddon",
#     "weblate.addons.generate.GenerateFileAddon",
#     "weblate.addons.generate.PseudolocaleAddon",
#     "weblate.addons.json.JSONCustomizeAddon",
```

(continues on next page)

(continuare din pagina precedentă)

```
#     "weblate.addons.properties.PropertiesSortAddon",
#     "weblate.addons.git.GitSquashAddon",
#     "weblate.addons.removal.RemoveComments",
#     "weblate.addons.removal.RemoveSuggestions",
#     "weblate.addons.resx.ResxUpdateAddon",
#     "weblate.addons.yaml.YAMLCustomizeAddon",
#     "weblate.addons.cdn.CDNJSAddon",
# )

# E-mail address that error messages come from.
SERVER_EMAIL = "noreply@example.com"

# Default email address to use for various automated correspondence from
# the site managers. Used for registration emails.
DEFAULT_FROM_EMAIL = "noreply@example.com"

# List of URLs your site is supposed to serve
ALLOWED_HOSTS = ["*"]

# Configuration for caching
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/1",
        # If redis is running on same host as Weblate, you might
        # want to use unix sockets instead:
        # "LOCATION": "unix:///var/run/redis/redis.sock?db=1",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
            "PARSER_CLASS": "redis.connection.HiredisParser",
            # If you set password here, adjust CELERY_BROKER_URL as well
            "PASSWORD": None,
            "CONNECTION_POOL_KWARGS": {},
        },
        "KEY_PREFIX": "weblate",
    },
    "avatar": {
        "BACKEND": "django.core.cache.backends.filebased.FileBasedCache",
        "LOCATION": os.path.join(DATA_DIR, "avatar-cache"),
        "TIMEOUT": 86400,
        "OPTIONS": {"MAX_ENTRIES": 1000},
    },
}

# Store sessions in cache
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
# Store messages in session
MESSAGE_STORAGE = "django.contrib.messages.storage.session.SessionStorage"

# REST framework settings for API
REST_FRAMEWORK = {
    # Use Django's standard `django.contrib.auth` permissions,
    # or allow read-only access for unauthenticated users.
    "DEFAULT_PERMISSION_CLASSES": [
        # Require authentication for login required sites
        "rest_framework.permissions.IsAuthenticated"
        if REQUIRE_LOGIN
        else "rest_framework.permissions.IsAuthenticatedOrReadOnly"
    ],
    "DEFAULT_AUTHENTICATION_CLASSES": (
        "rest_framework.authentication.TokenAuthentication",
```

(continues on next page)

(continuare din pagina precedentă)

```
        "weblate.api.authentication.BearerAuthentication",
        "rest_framework.authentication.SessionAuthentication",
    ),
    "DEFAULT_THROTTLE_CLASSES": (
        "weblate.api.throttling.UserRateThrottle",
        "weblate.api.throttling.AnonRateThrottle",
    ),
    "DEFAULT_THROTTLE_RATES": {"anon": "100/day", "user": "5000/hour"},
    "DEFAULT_PAGINATION_CLASS": ("rest_framework.pagination.PageNumberPagination"),
    "PAGE_SIZE": 20,
    "VIEW_DESCRIPTION_FUNCTION": "weblate.api.views.get_view_description",
    "UNAUTHENTICATED_USER": "weblate.auth.models.get_anonymous",
}

# Fonts CDN URL
FONTS_CDN_URL = None

# Django compressor offline mode
COMPRESS_OFFLINE = False
COMPRESS_OFFLINE_CONTEXT = [
    {"fonts_cdn_url": FONTS_CDN_URL, "STATIC_URL": STATIC_URL, "LANGUAGE_BIDI":␣
↪True},
    {"fonts_cdn_url": FONTS_CDN_URL, "STATIC_URL": STATIC_URL, "LANGUAGE_BIDI":␣
↪False},
]

# Require login for all URLs
if REQUIRE_LOGIN:
    LOGIN_REQUIRED_URLS = (r"/(.*)$",)

# In such case you will want to include some of the exceptions
# LOGIN_REQUIRED_URLS_EXCEPTIONS = (
#     rf"{URL_PREFIX}/accounts/(.*)$",  # Required for login
#     rf"{URL_PREFIX}/admin/login/(.*)$",  # Required for admin login
#     rf"{URL_PREFIX}/static/(.*)$",  # Required for development mode
#     rf"{URL_PREFIX}/widgets/(.*)$",  # Allowing public access to widgets
#     rf"{URL_PREFIX}/data/(.*)$",  # Allowing public access to data exports
#     rf"{URL_PREFIX}/hooks/(.*)$",  # Allowing public access to notification hooks
#     rf"{URL_PREFIX}/healthz/$",  # Allowing public access to health check
#     rf"{URL_PREFIX}/api/(.*)$",  # Allowing access to API
#     rf"{URL_PREFIX}/js/i18n/$",  # JavaScript localization
#     rf"{URL_PREFIX}/contact/$",  # Optional for contact form
#     rf"{URL_PREFIX}/legal/(.*)$",  # Optional for legal app
# )

# Silence some of the Django system checks
SILENCED_SYSTEM_CHECKS = [
    # We have modified django.contrib.auth.middleware.AuthenticationMiddleware
    # as weblate.accounts.middleware.AuthenticationMiddleware
    "admin.E408"
]

# Celery worker configuration for testing
# CELERY_TASK_ALWAYS_EAGER = True
# CELERY_BROKER_URL = "memory://"
# CELERY_TASK_EAGER_PROPAGATES = True
# Celery worker configuration for production
CELERY_TASK_ALWAYS_EAGER = False
CELERY_BROKER_URL = "redis://localhost:6379"
CELERY_RESULT_BACKEND = CELERY_BROKER_URL
```

(continues on next page)

(continuare din pagina precedentă)

```
# Celery settings, it is not recommended to change these
CELERY_WORKER_MAX_MEMORY_PER_CHILD = 200000
CELERY_BEAT_SCHEDULE_FILENAME = os.path.join(DATA_DIR, "celery", "beat-schedule")
CELERY_TASK_ROUTES = {
    "weblate.trans.tasks.auto_translate*": {"queue": "translate"},
    "weblate.accounts.tasks.notify_*": {"queue": "notify"},
    "weblate.accounts.tasks.send_mails": {"queue": "notify"},
    "weblate.utils.tasks.settings_backup": {"queue": "backup"},
    "weblate.utils.tasks.database_backup": {"queue": "backup"},
    "weblate.wladmin.tasks.backup": {"queue": "backup"},
    "weblate.wladmin.tasks.backup_service": {"queue": "backup"},
    "weblate.memory.tasks.*": {"queue": "memory"},
}

# Enable plain database backups
DATABASE_BACKUP = "plain"

# Enable auto updating
AUTO_UPDATE = False

# PGP commits signing
WEBLATE_GPG_IDENTITY = None

# Third party services integration
MATOMO_SITE_ID = None
MATOMO_URL = None
GOOGLE_ANALYTICS_ID = None
SENTRY_DSN = None
SENTRY_ENVIRONMENT = SITE_DOMAIN
AKISMET_API_KEY = None
```

2.18 Management commands

Notă: Running management commands under a different user than the one running your webserver can result in files getting wrong permissions, please check *Filesystem permissions* for more details.

You will find basic management commands (available as `./manage.py` in the Django sources, or as an extended set in a script called **`weblate`** installable atop Weblate).

2.18.1 Invoking management commands

As mentioned before, invocation depends on how you installed Weblate.

If using virtualenv for Weblate, you can either specify the full path to **`weblate`**, or activate the virtualenv prior to invoking it:

```
# Direct invocation
~/weblate-env/bin/weblate

# Activating virtualenv adds it to search path
. ~/weblate-env/bin/activate
weblate
```

If you are using source code directly (either from a tarball or Git checkout), the management script is `./manage.py` available in the Weblate sources. To run it:

```
python ./manage.py list_versions
```

If you've installed Weblate using the pip or pip3 installer, or by using the `./setup.py` script, the **weblate** is installed to your path (or virtualenv path), from where you can use it to control Weblate:

```
weblate list_versions
```

For the Docker image, the script is installed like above, and you can run it using **docker exec**:

```
docker exec --user weblate <container> weblate list_versions
```

For **docker-compose** the process is similar, you just have to use **docker-compose exec**:

```
docker-compose exec --user weblate weblate weblate list_versions
```

In case you need to pass it a file, you can temporary add a volume:

```
docker-compose exec --user weblate /tmp:/tmp weblate weblate importusers /tmp/
↪users.json
```

Vezi și:

Installing using Docker, *Installing on Debian and Ubuntu*, *Installing on SUSE and openSUSE*, *Installing on RedHat, Fedora and CentOS*, *Installing from sources*

2.18.2 add_suggestions

weblate add_suggestions <project> <component> <language> <file>

Nou în versiunea 2.5.

Imports a translation from the file to use as a suggestion for the given translation. It skips duplicated translations; only different ones are added.

--author USER@EXAMPLE.COM
: E-mail of author for the suggestions. This user has to exist prior to importing (you can create one in the admin interface if needed).

Exemplu:

```
weblate --author michal@cihar.com add_suggestions weblate application cs /tmp/
↪suggestions-cs.po
```

2.18.3 auto_translate

weblate auto_translate <project> <component> <language>

Nou în versiunea 2.5.

Schimbat în versiunea 4.6: Added parameter for translation mode.

Performs automatic translation based on other component translations.

--source PROJECT/COMPONENT
: Specifies the component to use as source available for translation. If not specified all components in the project are used.

--user USERNAME
: Specify username listed as author of the translations. „Anonymous user" is used if not specified.

--overwrite
: Whether to overwrite existing translations.

--inconsistent
Whether to overwrite existing translations that are inconsistent (see *Inconsecvent*).

--add
Automatically add language if a given translation does not exist.

--mt MT
Use machine translation instead of other components as machine translations.

--threshold THRESHOLD
Similarity threshold for machine translation, defaults to 80.

--mode MODE
Specify translation mode, default is `translate` but `fuzzy` or `suggest` can be used.

Exemplu:

```
weblate auto_translate --user nijel --inconsistent --source weblate/application␣
↪weblate website cs
```

Vezi și:

Traducere automată

2.18.4 celery_queues

weblate celery_queues

Nou în versiunea 3.7.

Displays length of Celery task queues.

2.18.5 checkgit

weblate checkgit <project|project/component>

Prints current state of the back-end Git repository.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

2.18.6 commitgit

weblate commitgit <project|project/component>

Commits any possible pending changes to the back-end Git repository.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

2.18.7 commit_pending

weblate commit_pending <project|project/component>

Commits pending changes older than a given age.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

--age `HOURS`
Age in hours for committing. If not specified the value configured in *Component configuration* is used.

Notă: This is automatically performed in the background by Weblate, so there no real need to invoke this manually, besides forcing an earlier commit than specified by *Component configuration*.

Vezi și:

Running maintenance tasks, `COMMIT_PENDING_HOURS`

2.18.8 cleanuptrans

weblate cleanuptrans

Cleans up orphaned checks and translation suggestions. There is normally no need to run this manually, as the cleanups happen automatically in the background.

Vezi și:

Running maintenance tasks

2.18.9 createadmin

weblate createadmin

Creates an `admin` account with a random password, unless it is specified.

--password `PASSWORD`
Provides a password on the command-line, to not generate a random one.

--no-password
Do not set password, this can be useful with *–update*.

--username `USERNAME`
Use the given name instead of `admin`.

--email `USER@EXAMPLE.COM`
Specify the admin e-mail address.

--name
Specify the admin name (visible).

--update
Update the existing user (you can use this to change passwords).

Schimbat în versiunea 2.9: Added parameters `--username`, `--email`, `--name` and `--update`.

2.18.10 dump_memory

weblate dump_memory

Nou în versiunea 2.20.

Export a JSON file containing Weblate Translation Memory content.

Vezi și:

Memorie de traducere, *Weblate Translation Memory Schema*

2.18.11 dumpuserdata

weblate dumpuserdata <file.json>

Dumps userdata to a file for later use by `importuserdata`

Sugestie: This comes in handy when migrating or merging Weblate instances.

2.18.12 import_demo

weblate import_demo

Nou în versiunea 4.1.

Creates a demo project with components based on <https://github.com/WeblateOrg/demo>.

This can be useful when developing Weblate.

2.18.13 import_json

weblate import_json <json-file>

Nou în versiunea 2.7.

Batch import of components based on JSON data.

The imported JSON file structure pretty much corresponds to the component object (see `GET /api/components/(string:project)/(string:component)/`). You have to include the `name` and `filemask` fields.

--project `PROJECT`
Specifies where the components will be imported from.

--main-component `COMPONENT`
Use the given VCS repository from this component for all of them.

--ignore
Skip (already) imported components.

--update
Update (already) imported components.

Schimbat în versiunea 2.9: The parameters `--ignore` and `--update` are there to deal with already imported components.

Example of JSON file:

```
[
  {
    "slug": "po",
    "name": "Gettext PO",
    "file_format": "po",
    "filemask": "po/*.po",
    "new_lang": "none"
  },
  {
    "name": "Android",
    "filemask": "android/values-*/strings.xml",
    "template": "android/values/strings.xml",
    "repo": "weblate://test/test",
    "file_format": "aresource"
  }
]
```

Vezi și:

import_memory

2.18.14 import_memory

weblate import_memory <file>

Nou în versiunea 2.20.

Imports a TMX or JSON file into the Weblate translation memory.

--language-map LANGMAP

Allows mapping languages in the TMX to the Weblate translation memory. The language codes are mapped after normalization usually done by Weblate.

`--language-map en_US:en` will for example import all `en_US` strings as `en` ones.

This can be useful in case your TMX file locales happen not to match what you use in Weblate.

Vezi și:

Memorie de traducere, *Weblate Translation Memory Schema*

2.18.15 import_project

weblate import_project <project> <gitrepo> <branch> <filemask>

Schimbat în versiunea 3.0: The import_project command is now based on the *Descoperirea componentelor* add-on, leading to some changes in behavior and what parameters are accepted.

Batch imports components into project based on filemask.

<project> names an existing project, into which the components are to be imported.

The *<gitrepo>* defines the Git repository URL to use, and *<branch>* signifies the Git branch. To import additional translation components from an existing Weblate component, use a *weblate://<project>/<component>* URL for the *<gitrepo>*.

The *<filemask>* defines file discovery for the repository. It can be either be made simple using wildcards, or it can use the full power of regular expressions.

The simple matching uses `**` for component name and `*` for language, for example: `**/*.po`

The regular expression has to contain groups named *component* and *language*. For example: `(?P<language>[^/]*)/(?P<component>[^-/]*)\.po`

The import matches existing components based on files and adds the ones that do not exist. It does not change already existing ones.

--name-template TEMPLATE
Customize the name of a component using Django template syntax.

For example: `Documentation: {{ component }}`

--base-file-template TEMPLATE
Customize the base file for monolingual translations.

For example: `{{ component }}/res/values/string.xml`

--new-base-template TEMPLATE
Customize the base file for addition of new translations.

For example: `{{ component }}/ts/en.ts`

--file-format FORMAT
You can also specify the file format to use (see *Supported file formats*), the default is auto-detection.

--language-regex REGEX
You can specify language filtering (see *Component configuration*) with this parameter. It has to be a valid regular expression.

--main-component
You can specify which component will be chosen as the main one—the one actually containing the VCS repository.

--license NAME
Specify the overall, project or component translation license.

--license-url URL
Specify the URL where the translation license is to be found.

--vcs NAME
In case you need to specify which version control system to use, you can do it here. The default version control is Git.

To give you some examples, let's try importing two projects.

First The Debian Handbook translations, where each language has separate a folder with the translations of each chapter:

```
weblate import_project \
    debian-handbook \
    git://anonscm.debian.org/debian-handbook/debian-handbook.git \
    squeeze/master \
    '*/**.po'
```

Then the Tanaguru tool, where the file format needs be specified, along with the base file template, and how all components and translations are located in single folder:

```
weblate import_project \
    --file-format=properties \
    --base-file-template=web-app/tgol-web-app/src/main/resources/i18n/%s-I18N.
↪properties \
    tanaguru \
    https://github.com/Tanaguru/Tanaguru \
    master \
    web-app/tgol-web-app/src/main/resources/i18n/**-I18N_*.properties
```

More complex example of parsing of filenames to get the correct component and language out of a filename like `src/security/Numerous_security_holes_in_0.10.1.de.po`:

```
weblate import_project \
    tails \
    git://git.tails.boum.org/tails master \
    'wiki/src/security/(?P<component>.*)\.(?P<language>[^.]*)\.po$'
```

Filtering only translations in a chosen language:

```
./manage import_project \
    --language-regex '^(cs|sk)$' \
    weblate \
    https://github.com/WeblateOrg/weblate.git \
    'weblate/locale/*/LC_MESSAGES/**.po'
```

Importing Sphinx documentation split to multiple files:

```
$ weblate import_project --name-template 'Documentation: %s' \
    --file-format po \
    project https://github.com/project/docs.git master \
    'docs/locale/*/LC_MESSAGES/**.po'
```

Importing Sphinx documentation split to multiple files and directories:

```
$ weblate import_project --name-template 'Directory 1: %s' \
    --file-format po \
    project https://github.com/project/docs.git master \
    'docs/locale/*/LC_MESSAGES/dir1/**.po'
$ weblate import_project --name-template 'Directory 2: %s' \
    --file-format po \
    project https://github.com/project/docs.git master \
    'docs/locale/*/LC_MESSAGES/dir2/**.po'
```

Vezi și:

More detailed examples can be found in the starting chapter, alternatively you might want to use *import_json*.

2.18.16 importuserdata

weblate importuserdata <file.json>

Imports user data from a file created by *dumpuserdata*

2.18.17 importusers

weblate importusers --check <file.json>

Imports users from JSON dump of the Django auth_users database.

--check

With this option it will just check whether a given file can be imported and report possible conflicts arising from usernames or e-mails.

You can dump users from the existing Django installation using:

```
weblate dumpdata auth.User > users.json
```

2.18.18 install_addon

Nou în versiunea 3.2.

weblate install_addon --addon ADDON <project|project/component>

Installs an add-on to a set of components.

--addon ADDON
Name of the add-on to install. For example `weblate.gettext.customize`.

--configuration CONFIG
JSON encoded configuration of an add-on.

--update
Update the existing add-on configuration.

You can either define which project or component to install the add-on in (for example `weblate/application`), or use `--all` to include all existing components.

To install *Personalizați ieșirea gettext* for all components:

```
weblate install_addon --addon weblate.gettext.customize --config '{"width": -1}' --
↪update --all
```

Vezi și:

Add-ons

2.18.19 list_languages

weblate list_languages <locale>

Lists supported languages in MediaWiki markup - language codes, English names and localized names.

This is used to generate <https://wiki.l10n.cz/Slovn%C3%ADk_s_n%C3%A1zvy_jazyk%C5%AF>.

2.18.20 list_translators

weblate list_translators <project|project/component>

Lists translators by contributed language for the given project:

```
[French]
Jean Dupont <jean.dupont@example.com>
[English]
John Doe <jd@example.com>
```

--language-code
List names by language code instead of language name.

You can either define which project or component to use (for example `weblate/application`), or use `--all` to list translators from all existing components.

2.18.21 list_versions

weblate list_versions

Lists all Weblate dependencies and their versions.

2.18.22 loadpo

weblate loadpo <project|project/component>

Reloads translations from disk (for example in case you have done some updates in the VCS repository).

--force
Force update, even if the files should be up-to-date.

--lang LANGUAGE
Limit processing to a single language.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Notă: You seldom need to invoke this, Weblate will automatically load changed files for every VCS update. This is needed in case you manually changed an underlying Weblate VCS repository or in some special cases following an upgrade.

2.18.23 lock_translation

weblate lock_translation <project|project/component>

Prevents further translation of a component.

Sugestie: Useful in case you want to do some maintenance on the underlying repository.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Vezi și:

unlock_translation

2.18.24 move_language

weblate move_language source target

Nou în versiunea 3.0.

Allows you to merge language content. This is useful when updating to a new version which contains aliases for previously unknown languages that have been created with the *(generated)* suffix. It moves all content from the *source* language to the *target* one.

Exemplu:

```
weblate move_language cze cs
```

After moving the content, you should check whether there is anything left (this is subject to race conditions when somebody updates the repository meanwhile) and remove the *(generated)* language.

2.18.25 pushgit

weblate pushgit <project|project/component>

Pushes committed changes to the upstream VCS repository.

--force-commit
Force commits any pending changes, prior to pushing.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Notă: Weblate pushes changes automatically if *Împingeți pe comitere* in *Component configuration* is turned on, which is the default.

2.18.26 unlock_translation

weblate unlock_translation <project|project/component>

Unlocks a given component, making it available for translation.

Sugestie: Useful in case you want to do some maintenance on the underlying repository.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Vezi și:

lock_translation

2.18.27 setupgroups

weblate setupgroups

Configures default groups and optionally assigns all users to that default group.

--no-privs-update
Turns off automatic updating of existing groups (only adds new ones).

--no-projects-update
Prevents automatic updates of groups for existing projects. This allows adding newly added groups to existing projects, see *Control acces proiect*.

Vezi și:

List of privileges

2.18.28 setuplang

weblate setuplang

Updates list of defined languages in Weblate.

--no-update
Turns off automatic updates of existing languages (only adds new ones).

2.18.29 updatechecks

`weblate updatechecks <project|project/component>`

Updates all checks for all strings.

Sugestie: Useful for upgrades which do major changes to checks.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

2.18.30 updategit

`weblate updategit <project|project/component>`

Fetches remote VCS repositories and updates the internal cache.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Notă: Usually it is better to configure hooks in the repository to trigger *Cârlige de notificare*, instead of regular polling by `updategit`.

2.19 Anunțuri

Schimbat în versiunea 4.0: În versiunile anterioare, această funcție se numea mesaje de tablă albă.

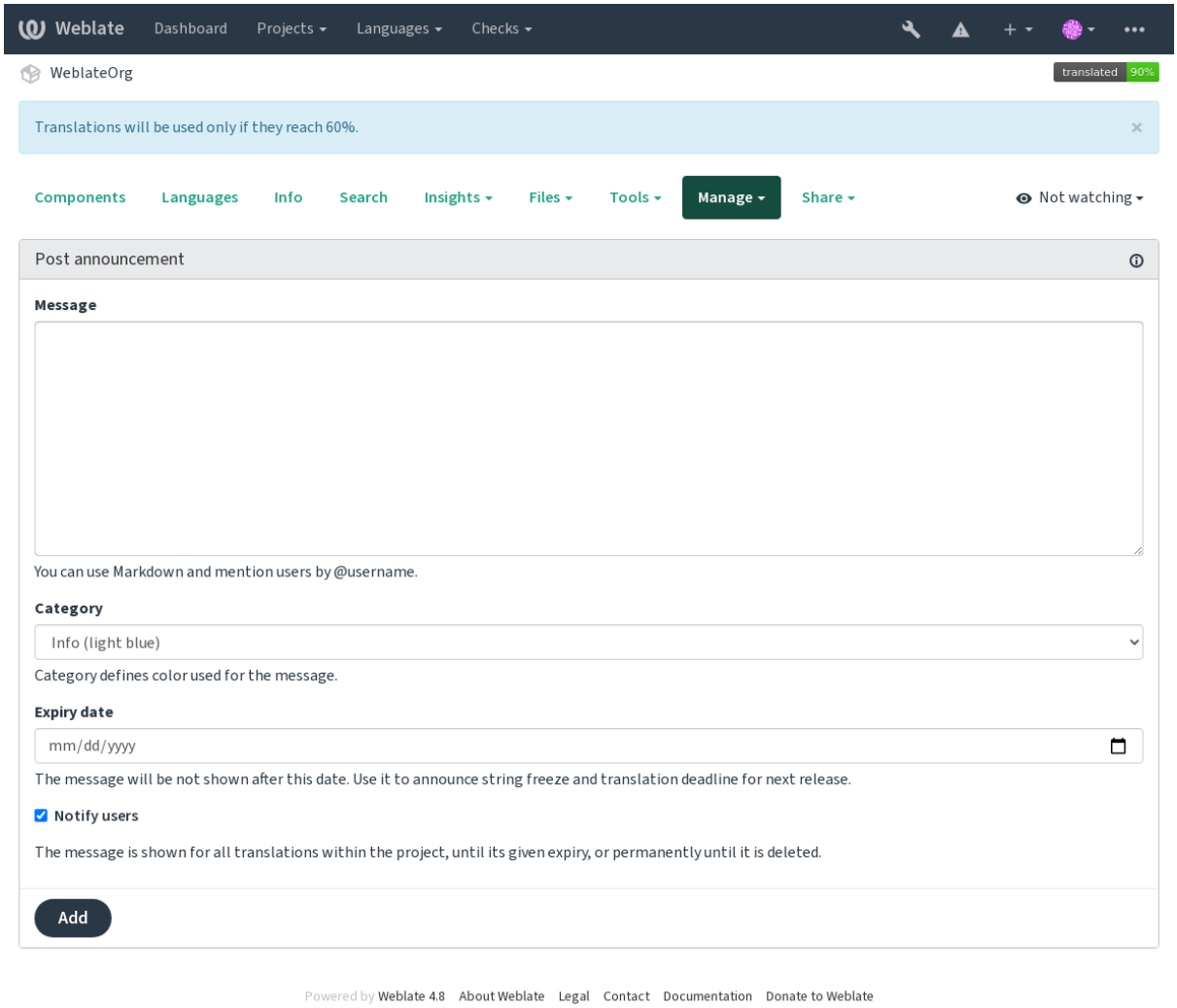
Furnizați informații traducătorilor dvs. prin postarea de anunțuri, la nivelul întregului site, pentru fiecare proiect, componentă sau limbă.

Anunțați scopul, termenele limită, statutul sau specificați obiectivele pentru traducere.

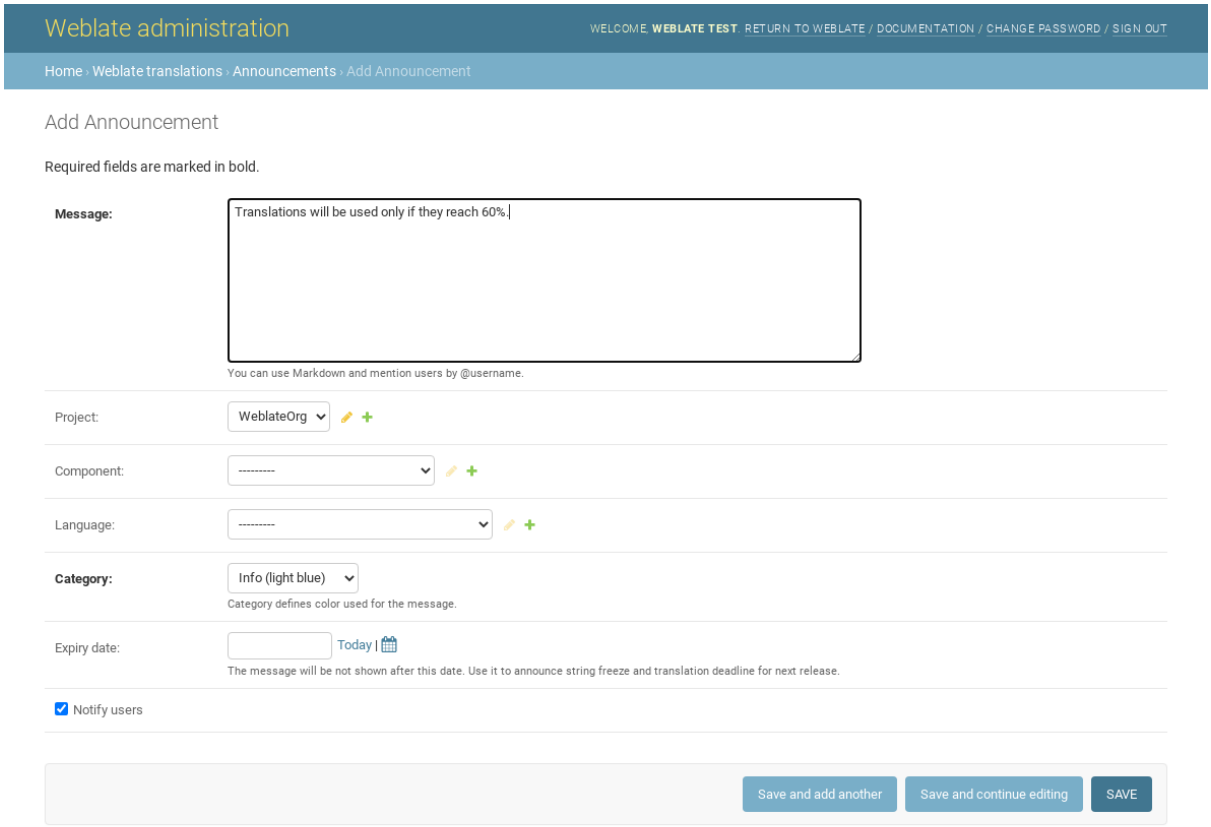
Utilizatorii vor primi notificări cu privire la anunțurile privind proiectele urmărite (cu excepția cazului în care renunță).

Acest lucru poate fi util pentru diverse lucruri, de la anunțarea scopului site-ului până la specificarea țintelor pentru traduceri.

Anunțurile pot fi postate pe fiecare nivel în meniul *Manage*, folosind *Post announcement*:



Acesta poate fi adăugat și prin intermediul interfeței de administrare:



Anunțurile sunt apoi afișate în funcție de contextul specificat:

Niciun context specificat

Afișat pe tabloul de bord (pagina de destinație).

Proiect specificat

Afișat în cadrul proiectului, inclusiv toate componentele și traducerile acestuia.

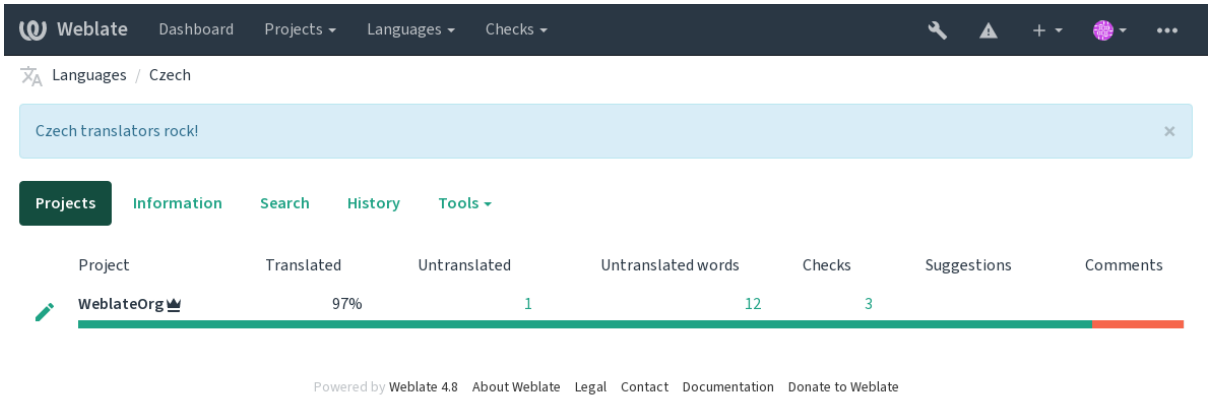
Componenta specificată

Afișat pentru o anumită componentă și pentru toate traducerile acesteia.

Limba specificată

Afișat în prezentarea generală a limbii și a tuturor traducerilor în limba respectivă.

Iată cum arată pe pagina de prezentare generală a limbilor:



2.20 Liste de componente

Specificați mai multe liste de componente care să apară ca opțiuni pe tabloul de bord al utilizatorului, din care utilizatorii pot alege una ca vizualizare implicită. Consultați *Tabloul de bord* pentru a afla mai multe.

Schimbat în versiunea 2.20: Pentru fiecare listă de componente prezentată în tabloul de bord se va prezenta o stare.

Numele și conținutul listelor de componente pot fi specificate în interfața de administrare, în secțiunea *Liste de componente*. Fiecare listă de componente trebuie să aibă un nume care să fie afișat utilizatorului și un slug care să o reprezinte în URL.

Schimbat în versiunea 2.13: Modificați setările tabloului de bord pentru utilizatorii anonimi din interfața de administrare, modificând ce tablou de bord este prezentat utilizatorilor neautentificați.

2.20.1 Liste automate de componente

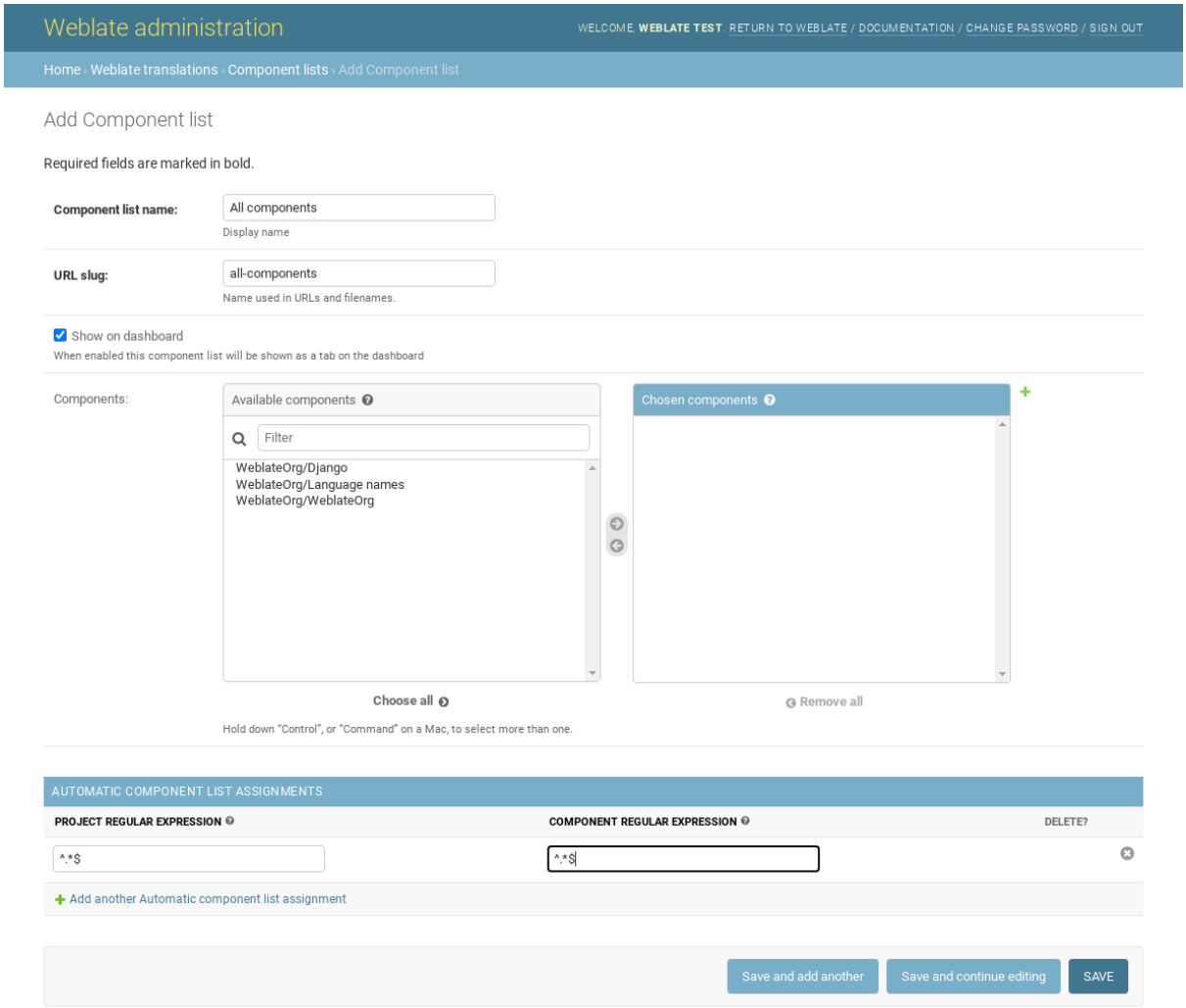
Nou în versiunea 2.13.

Adăugați componentele în listă în mod automat pe baza slug-ului lor prin crearea de reguli *Automatic component list assignment*.

- Utile pentru menținerea listelor de componente pentru instalații mari sau în cazul în care doriți să aveți o singură listă de componente cu toate componentele din instalația dumneavoastră Weblate.

Sugestie: Faceți o listă de componente care să conțină toate componentele instalației Weblate.

1. Define *Automatic component list assignment* with `^.*$` as regular expression in both the project and the component fields, as shown on this image:



2.21 Optional Weblate modules

Several optional modules are available for your setup.

2.21.1 Git exporter

Nou în versiunea 2.10.

Provides you read-only access to the underlying Git repository using HTTP(S).

Installation

1. Add `weblate.gitexport` to installed apps in `settings.py`:

```
INSTALLED_APPS += ("weblate.gitexport",)
```

2. Export existing repositories by migrating your database after installation:

```
weblate migrate
```

Usage

The module automatically hooks into Weblate and sets the exported repository URL in the *Component configuration*. The repositories are accessible under the `/git/` part of the Weblate URL, for example `https://example.org/git/weblate/main/`.

Repositories for publicly available projects can be cloned without authentication:

```
git clone 'https://example.org/git/weblate/main/'
```

Access to browse the repositories with restricted access (with *Private access control* or when `REQUIRE_LOGIN` is enabled) requires an API token which can be obtained in your *user profile*:

```
git clone 'https://user:KEY@example.org/git/weblate/main/'
```

Sugestie: By default members or *Users* group and anonymous user have access to the repositories for public projects via *Access repository* and *Power user* roles.

2.21.2 Facturare

Nou în versiunea 2.4.

This is used on Hosted Weblate to define billing plans, track invoices and usage limits.

Installation

1. Add `weblate.billing` to installed apps in `settings.py`:

```
INSTALLED_APPS += ("weblate.billing",)
```

2. Run the database migration to optionally install additional database structures for the module:

```
weblate migrate
```

Usage

After installation you can control billing in the admin interface. Users with billing enabled will get new *Billing* tab in their *Profilul utilizatorului*.

The billing module additionally allows project admins to create new projects and components without being superusers (see *Adding translation projects and components*). This is possible when following conditions are met:

- The billing is in its configured limits (any overusage results in blocking of project/component creation) and paid (if its price is non zero)
- The user is admin of existing project with billing or user is owner of billing (the latter is necessary when creating new billing for users to be able to import new projects).

Upon project creation user is able to choose which billing should be charged for the project in case he has access to more of them.

2.21.3 Legal

Nou în versiunea 2.15.

This is used on Hosted Weblate to provide required legal documents. It comes provided with blank documents, and you are expected to fill out the following templates in the documents:

`legal/documents/tos.html` Terms of service document

`legal/documents/privacy.html` Privacy policy document

`legal/documents/summary.html` Short overview of the terms of service and privacy policy

---

Notă: Legal documents for the Hosted Weblate service are available in this Git repository <https://github.com/WeblateOrg/wllegal/tree/main/wllegal/templates/legal/documents>.

Most likely these will not be directly usable to you, but might come in handy as a starting point if adjusted to meet your needs.

Installation

1. Add `weblate.legal` to installed apps in `settings.py`:

```
INSTALLED_APPS += ("weblate.legal",)

# Optional:

# Social auth pipeline to confirm TOS upon registration/subsequent sign in
SOCIAL_AUTH_PIPELINE += ("weblate.legal.pipeline.tos_confirm",)

# Middleware to enforce TOS confirmation of signed in users
MIDDLEWARE += [
    "weblate.legal.middleware.RequireTOSMiddleware",
]
```

2. Run the database migration to optionally install additional database structures for the module:

```
weblate migrate
```

3. Edit the legal documents in the `weblate/legal/templates/legal/` folder to match your service.

Usage

After installation and editing, the legal documents are shown in the Weblate UI.

2.21.4 Avatars

Avatars are downloaded and cached server-side to reduce information leaks to the sites serving them by default. The built-in support for fetching avatars from e-mails addresses configured for it can be turned off using `ENABLE_AVATARS`.

Weblate currently supports:

- Gravatar
- Libravatar

Vezi și:

Avatar caching, `AVATAR_URL_PREFIX`, `ENABLE_AVATARS`

2.21.5 Spam protection

You can protect against spamming by users by using the Akismet service.

1. Install the *akismet* Python module (this is already included in the official Docker image).
2. Obtain the Akismet API key.
3. Store it as `AKISMET_API_KEY` or `WEBLATE_AKISMET_API_KEY` in Docker.

Following content is sent to Akismet for checking:

- Suggestions from unauthenticated users
- Project and component descriptions and links

Notă: This (among other things) relies on IP address of the client, please see *Running behind reverse proxy* for properly configuring that.

Vezi și:

Running behind reverse proxy, `AKISMET_API_KEY`, `WEBLATE_AKISMET_API_KEY`

2.21.6 Signing Git commits with GnuPG

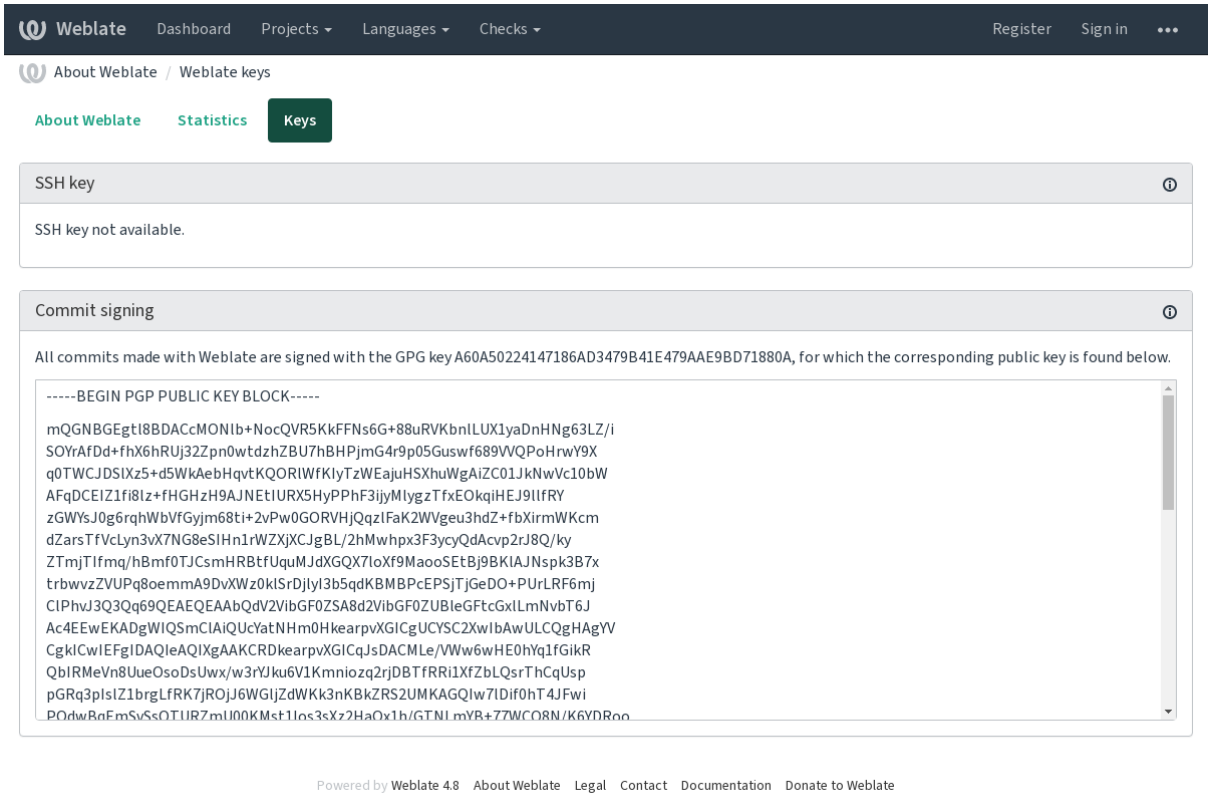
Nou în versiunea 3.1.

All commits can be signed by the GnuPG key of the Weblate instance.

1. Turn on `WEBLATE_GPG_IDENTITY`. (Weblate will generate a GnuPG key when needed and will use it to sign all translation commits.)

This feature needs GnuPG 2.1 or newer installed.

You can find the key in the `DATA_DIR` and the public key is shown on the „About” page:



2. Alternatively you can also import existing keys into Weblate, just set `HOME=$DATA_DIR/home` when invoking gpg.

Vezi și:

WEBLATE_GPG_IDENTITY

2.21.7 Rate limiting

Schimbat în versiunea 3.2: The rate limiting now accepts more fine-grained configuration.

Schimbat în versiunea 4.6: The rate limiting no longer applies to superusers.

Several operations in Weblate are rate limited. At most *RATELIMIT_ATTEMPTS* attempts are allowed within *RATELIMIT_WINDOW* seconds. The user is then blocked for *RATELIMIT_LOCKOUT*. There are also settings specific to scopes, for example `RATELIMIT_CONTACT_ATTEMPTS` or `RATELIMIT_TRANSLATE_ATTEMPTS`. The table below is a full list of available scopes.

The following operations are subject to rate limiting:

Nume	Domeniul de aplicare	Allowed attempts	Ratelimit window	Lockout period
Înregistrare	`REGISTRATION`	5	300	600
Sending message to admins	`MESSAGE`	5	300	600
Password authentication on sign in	`LOGIN`	5	300	600
Sitewide search	`SEARCH`	6	60	60
Translating	`TRANSLATE`	30	60	600
Adding to glossary	`GLOSSARY`	30	60	600
Starting translation into a new language	`LANGUAGE`	2	300	600
Creating new project	`PROJECT`	5	600	600

If a user fails to log in *AUTH_LOCK_ATTEMPTS* times, password authentication will be turned off on the account until having gone through the process of having its password reset.

The settings can be also applied in the Docker container by adding `WEBLATE_` prefix to the setting name, for example *RATELIMIT_ATTEMPTS* becomes *WEBLATE_RATELIMIT_ATTEMPTS*.

The API has separate rate limiting settings, see *API rate limiting*.

Vezi și:

Rate limiting, *Running behind reverse proxy*, *API rate limiting*

2.21.8 Fedora Messaging integration

Fedora Messaging is AMQP-based publisher for all changes happening in Weblate. You can hook additional services on changes happening in Weblate using this.

The Fedora Messaging integration is available as a separate Python module `weblate-fedora-messaging`. Please see <https://github.com/WeblateOrg/fedora_messaging/> for setup instructions.

2.22 Customizing Weblate

Extend and customize using Django and Python. Contribute your changes upstream so that everybody can benefit. This reduces your maintenance costs; code in Weblate is taken care of when changing internal interfaces or refactoring the code.

Atenționare: Neither internal interfaces nor templates are considered a stable API. Please review your own customizations for every upgrade, the interfaces or their semantics might change without notice.

Vezi și:

Contributing to Weblate

2.22.1 Creating a Python module

If you are not familiar with Python, you might want to look into Python For Beginners, explaining the basics and pointing to further tutorials.

To write some custom Python code (called a module), a place to store it is needed, either in the system path (usually something like `/usr/lib/python3.7/site-packages/`) or in the Weblate directory, which is also added to the interpreter search path.

Better yet, turn your customization into a proper Python package:

1. Create a folder for your package (we will use *weblate_customization*).
2. Within it, create a `setup.py` file to describe the package:

```
from setuptools import setup

setup(
    name="weblate_customization",
    version="0.0.1",
    author="Your name",
    author_email="yourname@example.com",
    description="Sample Custom check for Weblate.",
    license="GPLv3+",
    keywords="Weblate check example",
```

(continues on next page)

(continuare din pagina precedentă)

```
    packages=["weblate_customization"],
)
```

3. Create a folder for the Python module (also called `weblate_customization`) for the customization code.
4. Within it, create a `__init__.py` file to make sure Python can import the module.
5. This package can now be installed using **`pip install -e`**. More info to be found in "Editable" Installs.
6. Once installed, the module can be used in the Weblate configuration (for example `weblate_customization.checks.FooCheck`).

Your module structure should look like this:

```
weblate_customization
├── setup.py
└── weblate_customization
    ├── __init__.py
    ├── addons.py
    └── checks.py
```

You can find an example of customizing Weblate at <https://github.com/WeblateOrg/customize-example>, it covers all the topics described below.

2.22.2 Changing the logo

1. Create a simple Django app containing the static files you want to overwrite (see *Creating a Python module*).

   Branding appears in the following files:

   `icons/weblate.svg` Logo shown in the navigation bar.

   `logo-*.png` Web icons depending on screen resolution and web-browser.

   `favicon.ico` Web icon used by legacy browsers.

   `weblate-*.png` Avatars for bots or anonymous users. Some web-browsers use these as shortcut icons.

   `email-logo.png` Used in notifications e-mails.

2. Add it to `INSTALLED_APPS`:

   ```
   INSTALLED_APPS = (
       # Add your customization as first
       "weblate_customization",
       # Weblate apps are here…
   )
   ```

3. Run `weblate collectstatic --noinput`, to collect static files served to clients.

Vezi și:

Managing static files (e.g. images, JavaScript, CSS), *Serving static files*

2.22.3 Custom quality checks, addons and auto-fixes

To install your code for *Reparații automate personalizate*, *Întocmirea propriilor cecuri* or *Writing add-on* in Weblate:

1. Place the files into your Python module containing the Weblate customization (see *Creating a Python module*).
2. Add its fully-qualified path to the Python class in the dedicated settings (`WEBLATE_ADDONS`, `CHECK_LIST` or `AUTOFIX_LIST`):

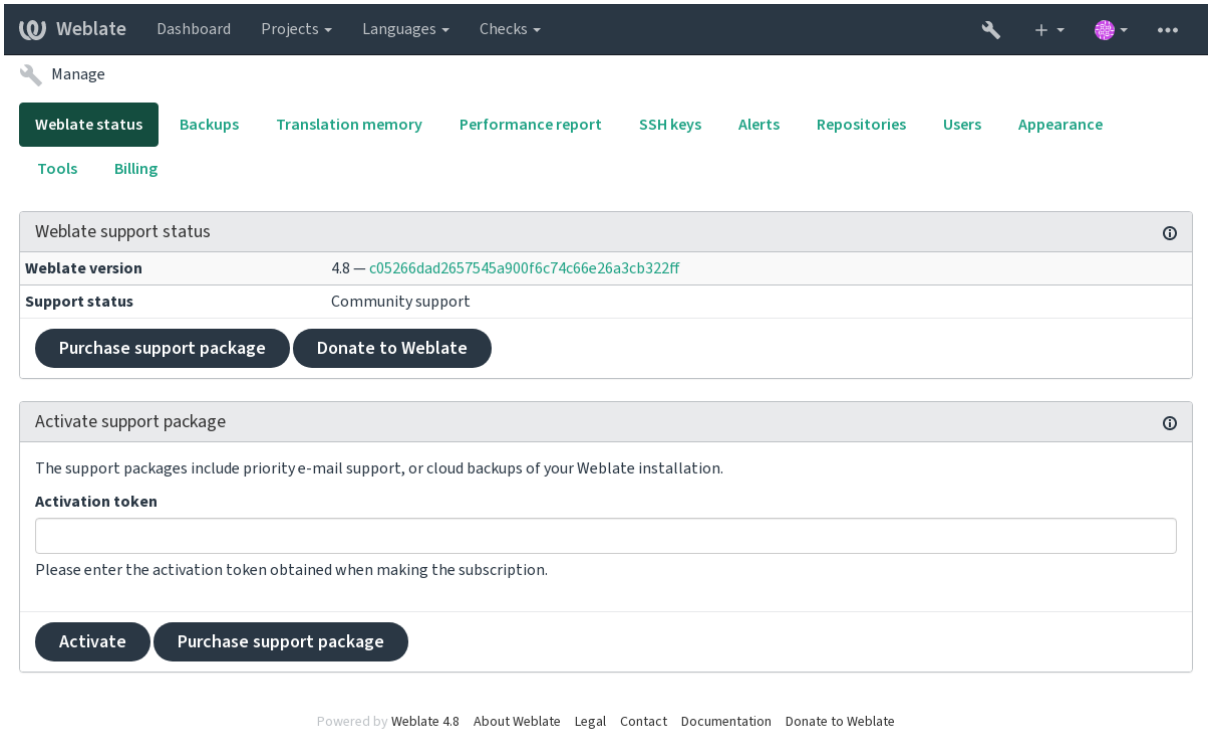
```
# Checks
CHECK_LIST += ("weblate_customization.checks.FooCheck",)

# Autofixes
AUTOFIX_LIST += ("weblate_customization.autofix.FooFixer",)

# Add-ons
WEBLATE_ADDONS += ("weblate_customization.addons.ExamplePreAddon",)
```

Vezi și:

Reparații automate personalizate, *Întocmirea propriilor cecuri*, *Writing add-on*, *Executing scripts from add-on*

2.23 Interfața de gestionare

Interfața de gestionare oferă setări de administrare în cadrul URL-ului `/manage/`. Aceasta este disponibilă pentru utilizatorii conectați cu privilegii de administrator, accesibilă cu ajutorul pictogramei cu cheie franceză din dreapta sus:



Acesta include o prezentare generală de bază a Weblate-ului dumneavoastră:

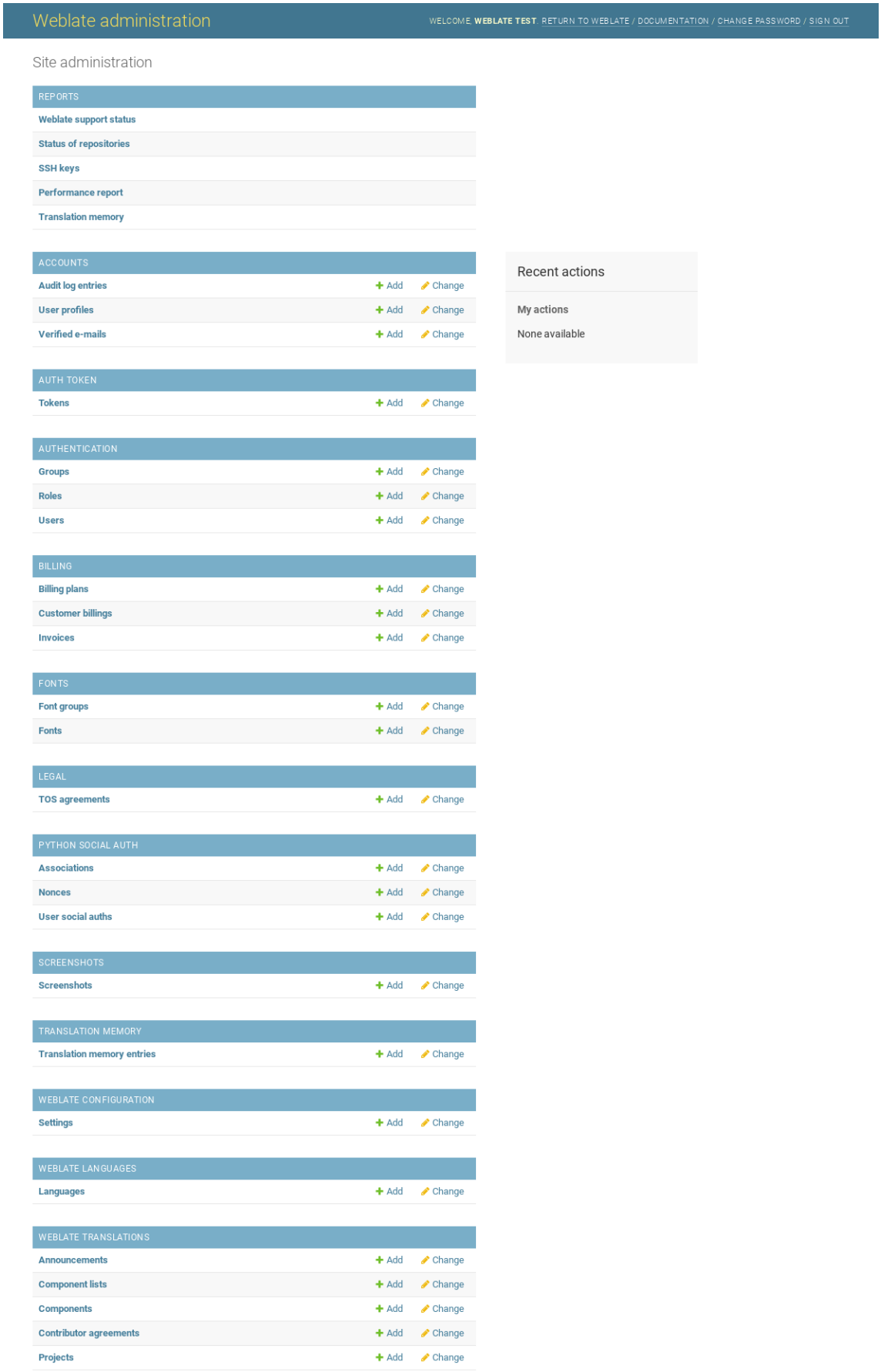
- Starea suportului, vezi *Getting support for Weblate*
- Copii de rezervă, vezi *Copierea de rezervă și mutarea Weblate*
- Shared translation memory, see *Memorie de traducere*
- Raport de performanță pentru a analiza starea de sănătate a Weblate și lungimea cozilor de așteptare Celery

- Gestionarea cheilor SSH, vezi *SSH repositories*
- Prezentare generală a alertelor pentru toate componentele, vezi alerts

2.23.1 Interfața de administrare Django

Atenționare: Va fi eliminat în viitor, deoarece utilizarea sa este descurajată - majoritatea funcțiilor pot fi gestionate direct în Weblate.

Aici puteți gestiona obiectele stocate în baza de date, cum ar fi utilizatorii, traducerile și alte setări:


Weblate administration
WELCOME, WEBLATE TEST. RETURN TO WEBLATE / DOCUMENTATION / CHANGE PASSWORD / SIGN OUT
Site administration
REPORTS
Weblate support status
Status of repositories
SSH keys
Performance report
Translation memory
ACCOUNTS
Audit log entries + Add Change
User profiles + Add Change
Verified e-mails + Add Change
AUTH TOKEN
Tokens + Add Change
AUTHENTICATION
Groups + Add Change
Roles + Add Change
Users + Add Change
BILLING
Billing plans + Add Change
Customer billings + Add Change
Invoices + Add Change
FONTS
Font groups + Add Change
Fonts + Add Change
LEGAL
TOS agreements + Add Change
PYTHON SOCIAL AUTH
Associations + Add Change
Nonces + Add Change
User social auths + Add Change
SCREENSHOTS
Screenshots + Add Change
TRANSLATION MEMORY
Translation memory entries + Add Change
WEBLATE CONFIGURATION
Settings + Add Change
WEBLATE LANGUAGES
Languages + Add Change
WEBLATE TRANSLATIONS
Announcements + Add Change
Component lists + Add Change
Components + Add Change
Contributor agreements + Add Change
Projects + Add Change
Recent actions
My actions
None available

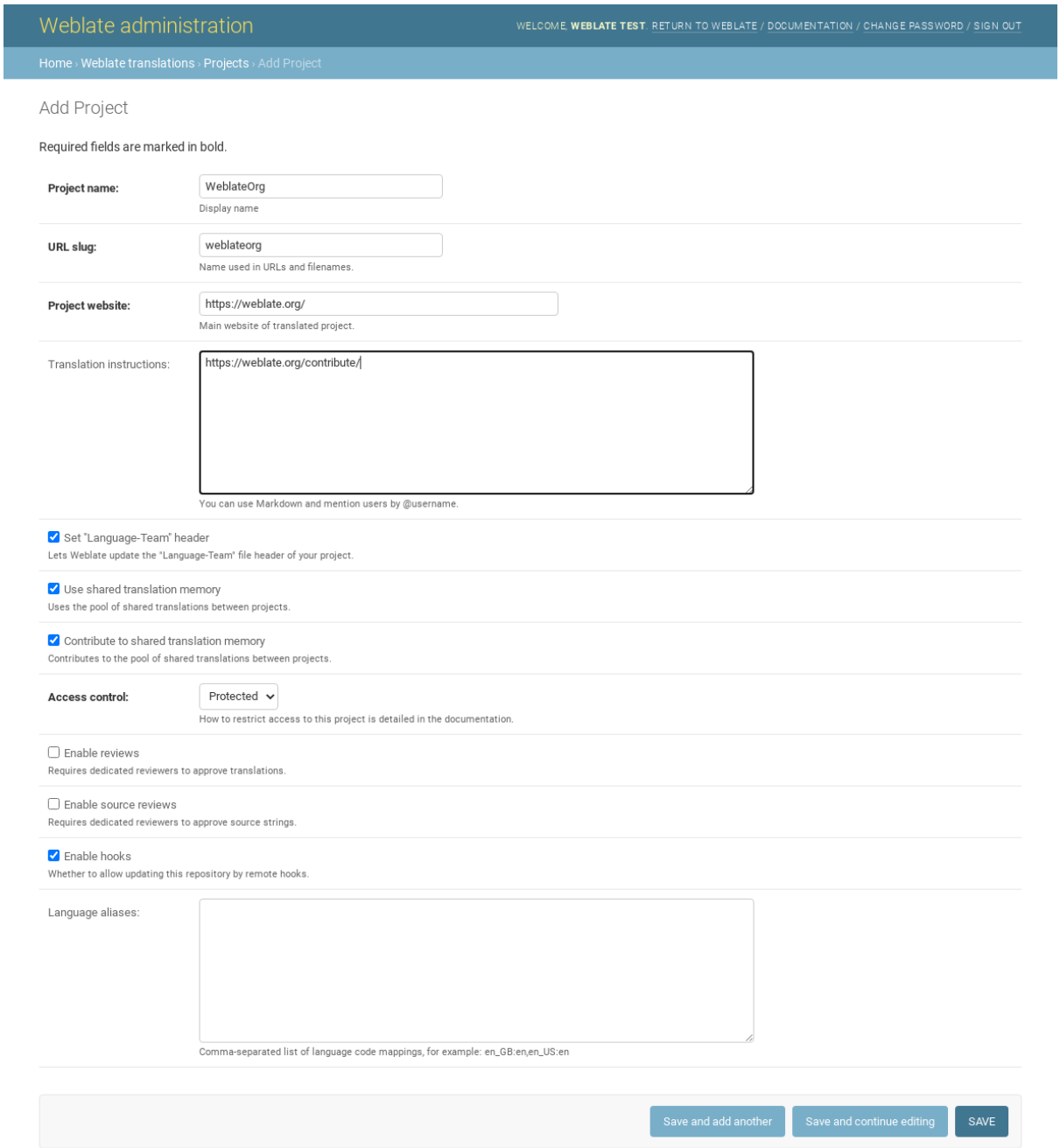
În secțiunea *Reports*, puteți verifica starea site-ului dumneavoastră, îl puteți ajusta pentru producție sau puteți gestiona cheile SSH utilizate pentru a accesa *Accessing repositories*.

Gestionați obiectele bazei de date în oricare dintre secțiuni. Cea mai interesantă este probabil *Weblate translations*, unde puteți gestiona proiecte traductibile, vezi *Project configuration* și *Component configuration*.

Weblate languages conține definițiile limbilor, explicate mai detaliat în *Language definitions*.

Adăugarea unui proiect

Adăugarea unui proiect servește drept container pentru toate componentele. De obicei, creați un proiect pentru o singură piesă de software sau o singură carte (See *Project configuration* for info on individual parameters):



Vezi și:

Project configuration

Componente bilingve

După ce ați adăugat un proiect, i se pot adăuga componente de traducere. (See *Component configuration* for info regarding individual parameters):

Weblate administration
WELCOME, WEBLATE TEST. RETURN TO WEBLATE / DOCUMENTATION / CHANGE PASSWORD / SIGN OUT
Home · Weblate translations · Components · Add Component
Add Component
IMPORT SPEED DOCUMENTATION
Required fields are marked in bold.
Component name:
Language names
Display name
URL slug:
language-names
Name used in URLs and filenames.
Project:
WeblateOrg
Version control system:
Git
Version control system to use to access your repository containing translations. You can also choose additional integration with third party providers to submit merge requests.
Source code repository:
https://github.com/WeblateOrg/demo.git
URL of a repository use weblate://project/component to share it with other component.
Repository push URL:
URL of a push repository pushing is turned off if empty.
Repository browser:
https://github.com/WeblateOrg/demo/blob/{{branch}}/{{filename}}#L{
Link to repository browser, use {{branch}} for branch, {{filename}} and {{line}} as filename and line placeholders. You might want to strip leading directory by using {{filename|parentdir}}.
Exported repository URL:
URL of repository where users can fetch changes from Weblate
Source string bug reporting address:
E-mail address for reports on errors in source strings. Leave empty for no e-mails.
Repository branch:
Repository branch to translate
Push branch:
Branch for pushing changes, leave empty to use repository branch
Filemask:
weblate/lang/data/locale/*/LC_MESSAGES/{{ij
Path of files to translate relative to repository root, use * instead of language code, for example: po/*.po or locale/*/LC_MESSAGES/{{ijango.po.
Monolingual base language file:
Filename of translation base file, containing all strings and their source, it is recommended for monolingual translation formats.
Edit base file
Whether users will be able to edit the base file for monolingual translations.
Intermediate language file:
Filename of intermediate translation file. In most cases this is a translation file provided by developers and is used when creating actual source strings.
Template for new translations:
weblate/lang/data/locale/{{ijango.pot
Filename of file used for creating new translations. For gettext choose .pot file.
File format:
gettext PO file
Locked
Locked component will not get any translation updates.
Allow translation propagation
Whether translation updates in other components will cause automatic translation in this one
Turn on suggestions
Whether to allow translation suggestions at all.
Suggestion voting
Users can only vote for suggestions and can't make direct translations.
Autoaccept suggestions:
0
Automatically accept suggestions with this number of votes, use 0 to turn it off.
Translation flags:
Additional comma-separated flags to influence Weblate behavior.
Enforced checks:
List of checks which can not be ignored.
Translation license:
GNU General Public License v3.0 or later
Contributor agreement:
User agreement which needs to be approved before a user can translate this component.
Adding new translation:
Create new language file
How to handle requests for creating new translations.
Language code style:
Default based on the file format
Customize language code used to generate the filename for translations created by Weblate.
Manage strings
Enables adding and removing strings straight from Weblate. If your strings are extracted from the source code or managed externally you probably want to keep it disabled.
Merge style:
Rebase
Define whether Weblate should merge the upstream repository or rebase changes onto it.
Commit message when translating:
Translated using Weblate ({{ language_name }})
Currently translated at {{ stats.translated_percent }}% ({{ stats.translated }} of {{ stats.all }} strings)
Translation: {{ project_name }}/{{ component_name }}
Translate-URL: {{ url }}
You can use template language for various info, please consult the documentation for more details.
Commit message when adding translation:
Added translation using Weblate ({{ language_name }})
You can use template language for various info, please consult the documentation for more details.
Commit message when removing translation:
Deleted translation using Weblate ({{ language_name }})
You can use template language for various info, please consult the documentation for more details.
Commit message when merging translation:
Merge branch '{{ component_remote_branch }}' into Weblate.
You can use template language for various info, please consult the documentation for more details.
Commit message when addon makes a change:
Update translation files
Updated by "{{ addon_name }}" hook in Weblate.
Translation: {{ project_name }}/{{ component_name }}
Translate-URL: {{ url }}
You can use template language for various info, please consult the documentation for more details.
Push on commit
Whether the repository should be pushed upstream on every commit.
Age of changes to commit:
24
Time in hours after which any pending changes will be committed to the VCS.
Lock on error
Whether the component should be locked on repository errors.
Source language:
English
Language used for source strings in all components
Language filter:
^(cs|hs|hr|)$
Regular expression used to filter translation files when scanning for filemask.
Variants regular expression:
Regular expression used to determine variants of a string.
Priority:
Medium
Components with higher priority are offered first to translators.
Restricted component
Restrict access to the component to only those explicitly given permission.
Share in projects:
WeblateOrg
Choose additional projects where this component will be listed. Hold down "Control", or "Command" on a Mac, to select more than one.
Use as a glossary
Glossary color:
Silver
Remote revision:
Local revision:
Save and add another
Save and continue editing
SAVE

Vezi și:

Component configuration, *Bilingual and monolingual formats*

Componente monolingve

Pentru o traducere mai ușoară a acestora, furnizați un fișier șablon care să conțină corespondența dintre ID-urile mesajelor și limba sursă respectivă (usually English). (See *Component configuration* for info regarding individual parameters):

Weblate administration

WELCOME, WEBLATE TEST. RETURN TO WEBLATE / DOCUMENTATION / CHANGE PASSWORD / SIGN OUT

Home › Weblate translations › Components › Add Component

Add Component

IMPORT SPHINX DOCUMENTATION

Required fields are marked in bold.

Component name: Android
Display name

URL slug: android
Name used in URLs and filenames.

Project: WeblateOrg

Version control system: Git
Version control system to use to access your repository containing translations. You can also choose additional integration with third party providers to submit merge requests.

Source code repository: weblate://weblateorg/language-names
URL of a repository, use weblate://project/component to share it with other component.

Repository push URL:
URL of a push repository, pushing is turned off if empty.

Repository browser:
Link to repository browser, use {{branch}} for branch, {{filename}} and {{line}} as filename and line placeholders. You might want to strip leading directory by using {{filename|parent+0}}.

Exported repository URL:
URL of repository where users can fetch changes from Weblate.

Source string bug reporting address:
E-mail address for reports on errors in source strings. Leave empty for no e-mails.

Repository branch:
Repository branch to translate

Push branch:
Branch for pushing changes, leave empty to use repository branch

Filemask: app/src/main/res/values-*/strings.xml
Path of files to translate relative to repository root, use * instead of language code, for example: po/* po or locale/*/LC_MESSAGES/*.po.

Monolingual base language file: app/src/main/res/values/strings.xml
Filename of translation base file, containing all strings and their source; it is recommended for monolingual translation formats.

☑ Edit base file
Whether users will be able to edit the base file for monolingual translations.

Intermediate language file:
Filename of intermediate translation file. In most cases this is a translation file provided by developers and is used when creating actual source strings.

Template for new translations:
Filename of file used for creating new translations. For gettext choose .pot file.

File format: Android String Resource

☐ Locked
Locked component will not get any translation updates.

☑ Allow translation propagation
Whether translation updates in other components will cause automatic translation in this one

☑ Turn on suggestions
Whether to allow translation suggestions at all.

☐ Suggestion voting
Users can only vote for suggestions and can't make direct translations.

Autoaccept suggestions: 0
Automatically accept suggestions with this number of votes; use 0 to turn it off.

Translation flags:
Additional comma-separated flags to influence Weblate behavior.

Enforced checks: []
List of checks which can not be ignored.

Translation license: MIT License

Contributor agreement:
User agreement which needs to be approved before a user can translate this component.

Adding new translation: Create new language file
How to handle requests for creating new translations.

Language code style: Default based on the file format
Customize language code used to generate the filename for translations created by Weblate.

☐ Manage strings
Enables adding and removing strings straight from Weblate. If your strings are extracted from the source code or managed externally you probably want to keep it disabled.

Merge style: Rebase
Define whether Weblate should merge the upstream repository or rebase changes onto it.

Commit message when translating:
Translated using Weblate ({{ language_name }})

Currently translated at {{ stats.translated_percent }}% ({{ stats.translated }} of {{ stats.all }} strings)

Translation: {{ project_name }}/{{ component_name }}
Translate-URL: {{ url }}

You can use template language for various info; please consult the documentation for more details.

Commit message when adding translation:
Added translation using Weblate ({{ language_name }})

You can use template language for various info; please consult the documentation for more details.

Commit message when removing translation:
Deleted translation using Weblate ({{ language_name }})

You can use template language for various info; please consult the documentation for more details.

Commit message when merging translation:
Merge branch '{{ component_remote_branch }}' into Weblate.

You can use template language for various info; please consult the documentation for more details.

Commit message when addon makes a change:
Update translation files

Updated by "{{ addon_name }}" hook in Weblate.

Translation: {{ project_name }}/{{ component_name }}
Translate-URL: {{ url }}

You can use template language for various info; please consult the documentation for more details.

☑ Push on commit
Whether the repository should be pushed upstream on every commit.

Age of changes to commit: 24
Time in hours after which any pending changes will be committed to the VCS.

☑ Lock on error
Whether the component should be locked on repository errors.

Source language: English
Language used for source strings in all components

Language filter: ^[^x]
Regular expression used to filter translation files when scanning for filemask.

Variants regular expression:
Regular expression used to determine variants of a string.

Priority: Medium
Components with higher priority are offered first to translators.

☐ Restricted component
Restrict access to the component to only those explicitly given permission.

Share in projects: WeblateOrg
Choose additional projects where this component will be listed. Hold down "Control", or "Command" on a Mac, to select more than one.

☐ Use as a glossary

Glossary color: Silver

Remote revision:

Local revision:

Save and add another | Save and continue editing | SAVE

Vezi și:

Component configuration, *Bilingual and monolingual formats*

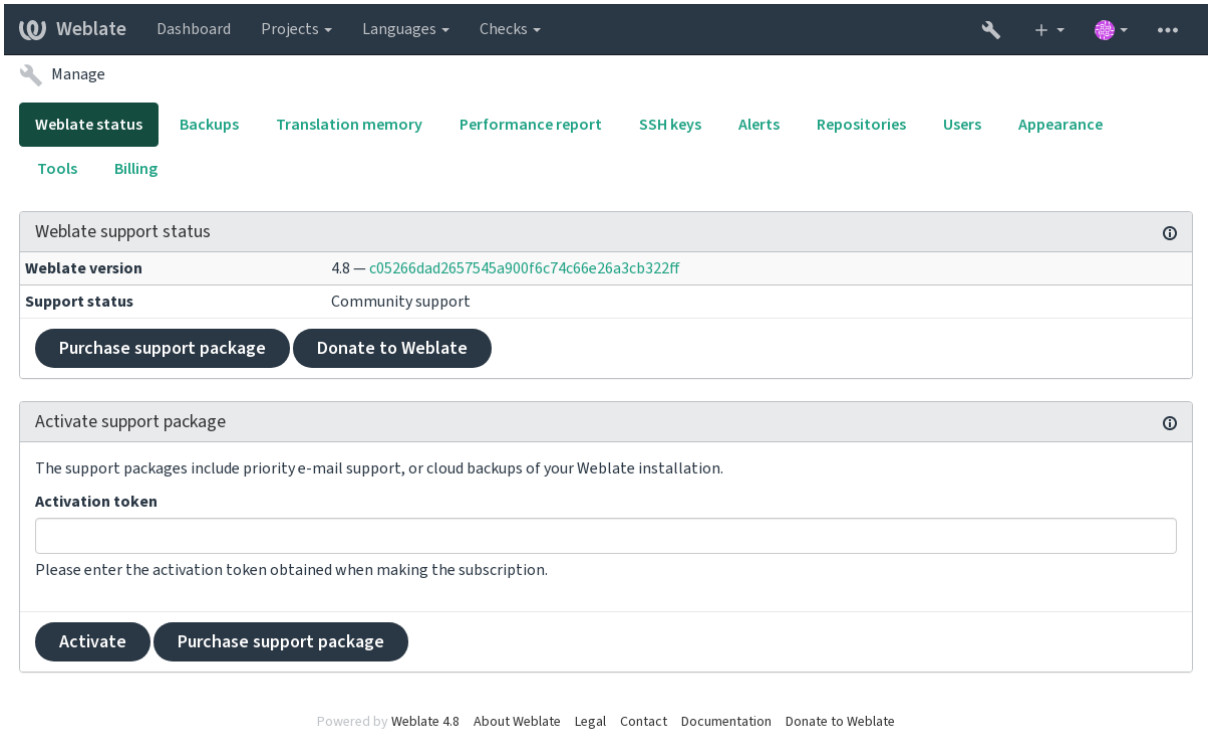
2.24 Getting support for Weblate

Weblate is copylefted libre software with community support. Subscribers receive priority support at no extra charge. Prepaid help packages are available for everyone. You can find more info about current support offerings at <https://weblate.org/support/>.

2.24.1 Integrating support

Nou în versiunea 3.8.

Purchased support packages can optionally be integrated into your Weblate subscription management interface, from where you will find a link to it. Basic instance details about your installation are also reported back to Weblate this way.



2.24.2 Data submitted to the Weblate

- URL where your Weblate instance is configured
- Your site title
- The Weblate version you are running
- Tallies of some objects in your Weblate database (projects, components, languages, source strings and users)
- The public SSH key of your instance

Additionally, when *Descoperiți Weblate* is turned on:

- List of public projects (name, URL and website)

No other data is submitted.

2.24.3 Integration services

- See if your support package is still valid
- *Stocarea de backup provizionată Weblate*
- *Descoperiți Weblate*

Sugestie: Purchased support packages are already activated upon purchase, and can be used without integrating them.

2.24.4 Descoperiți Weblate

Nou în versiunea 4.5.2.

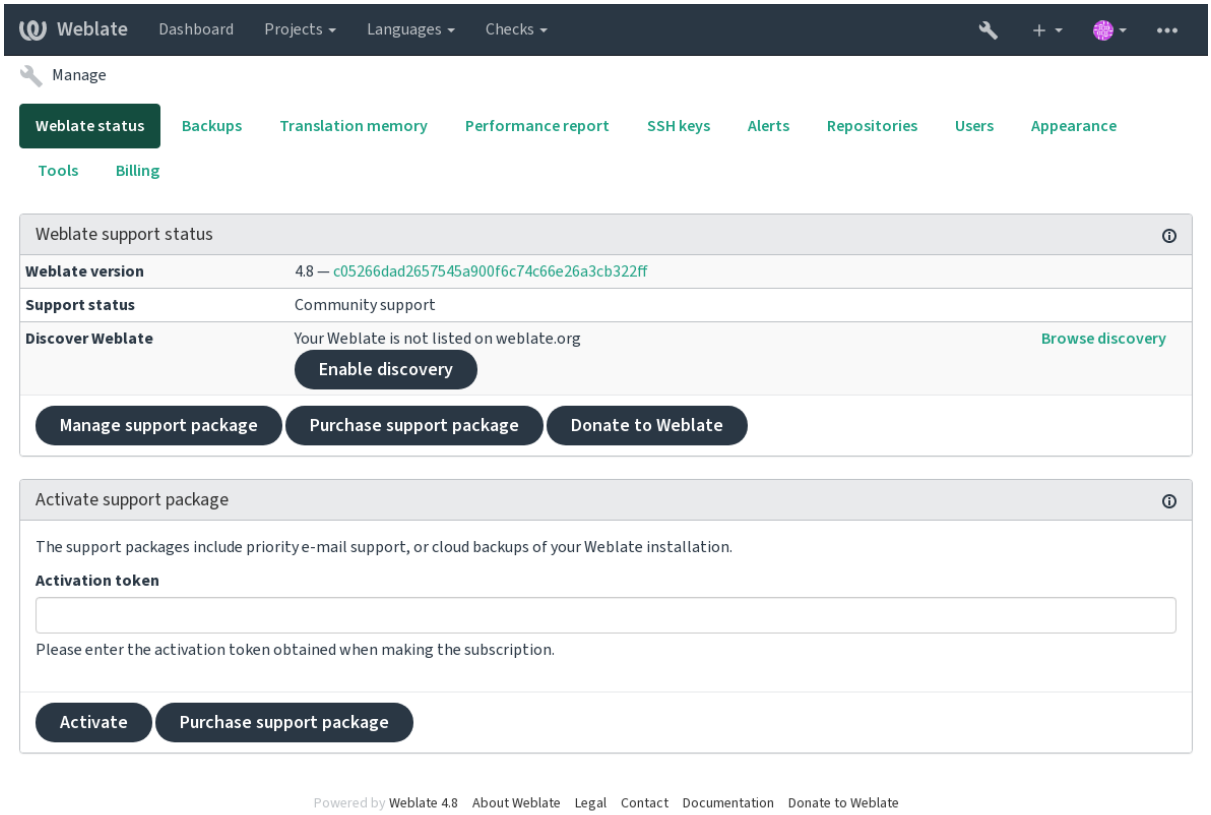
Notă: This feature is currently in early beta.

Discover Weblate is an opt-in service that makes it easier for users to find Weblate servers and communities. Users can browse registered services on <https://weblate.org/discover/>, and find there projects to contribute.

Getting listed

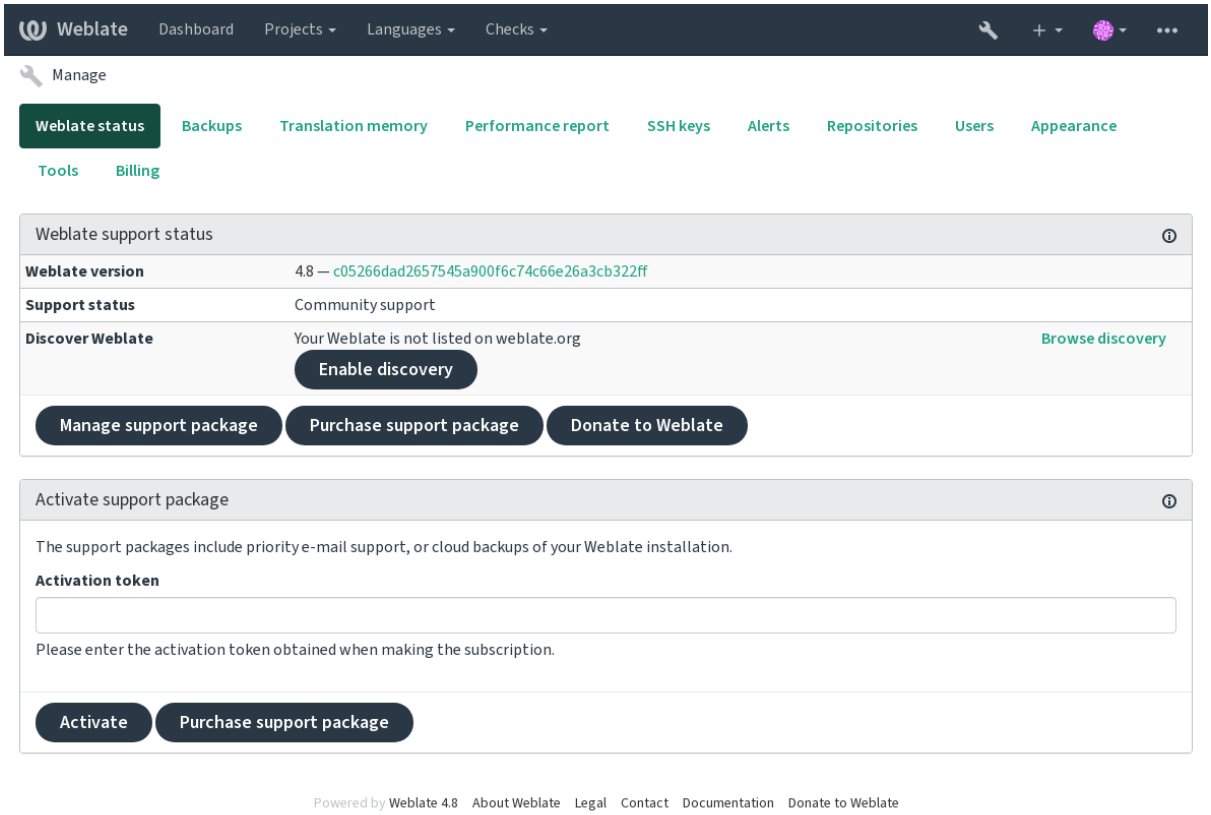
Sugestie: Participating in Discover Weblate makes Weblate submit some information about your server, please see *Data submitted to the Weblate*.

To list your server with an active support subscription (see *Integrating support*) in Discover Weblate all you need to do is turn this on in the management panel:



Listing your server without a support subscription in Discover Weblate:

1. Register yourself at <https://weblate.org/user/>
2. Register your Weblate server in the discovery database at <https://weblate.org/subscription/discovery/>
3. Confirm the service activation in your Weblate and turn on the discovery listing in your Weblate management page using *Enable discovery* button:



Customizing listing

You can customize the listing by providing a text and image (570 x 260 pixels) at <https://weblate.org/user/>.

2.25 Legal documents

Notă: Herein you will find various legal information you might need to operate Weblate in certain legal jurisdictions. It is provided as a means of guidance, without any warranty of accuracy or correctness. It is ultimately your responsibility to ensure that your use of Weblate complies with all applicable laws and regulations.

2.25.1 ITAR and other export controls

Weblate can be run within your own datacenter or virtual private cloud. As such, it can be used to store ITAR or other export-controlled information, however, end users are responsible for ensuring such compliance.

The Hosted Weblate service has not been audited for compliance with ITAR or other export controls, and does not currently offer the ability to restrict translations access by country.

2.25.2 US encryption controls

Weblate does not contain any cryptographic code, but might be subject export controls as it uses third party components utilizing cryptography for authentication, data-integrity and -confidentiality.

Most likely Weblate would be classified as ECCN 5D002 or 5D992 and, as publicly available libre software, it should not be subject to EAR (see Encryption items NOT Subject to the EAR).

Software components used by Weblate (listing only components related to cryptographic function):

Python See https://wiki.python.org/moin/PythonSoftwareFoundationLicenseFaq#Is_Python_subject_to_export_laws.3F

GnuPG Optionally used by Weblate

Git Optionally used by Weblate

curl Used by Git

OpenSSL Used by Python and cURL

The strength of encryption keys depends on the configuration of Weblate and the third party components it interacts with, but in any decent setup it will include all export restricted cryptographic functions:

- In excess of 56 bits for a symmetric algorithm
- Factorisation of integers in excess of 512 bits for an asymmetric algorithm
- Computation of discrete logarithms in a multiplicative group of a finite field of size greater than 512 bits for an asymmetric algorithm
- Discrete logarithms in a group different than above in excess of 112 bits for an asymmetric algorithm

Weblate doesn't have any cryptographic activation feature, but it can be configured in a way where no cryptography code would be involved. The cryptographic features include:

- Accessing remote servers using secure protocols (HTTPS)
- Generating signatures for code commits (PGP)

Vezi și:

Export Controls (EAR) on Open Source Software

CAPITOLUL 3

Contributor docs

3.1 Contributing to Weblate

There are dozens of ways to improve Weblate. You can choose the one you feel comfortable with, be it coding, graphics design, documentation, sponsorship, or an idea:

- *Reporting issues in Weblate*
- *Starting contributing code to Weblate*
- *Translating Weblate*
- *Contribute to Weblate documentation*
- *Weblate discussions*
- *Funding Weblate development*

3.1.1 Translating Weblate

Weblate is continually being translated using Weblate itself. Feel free to take your part in the effort of making Weblate available in as many human languages as possible. It brings Weblate closer to its users!

If you find a possible mistake in the source string, you can mark it with a comment in the Weblate editor. This way, it can be discussed and corrected. If you're certain, you can also click on the link in the *Source string location* section and submit a PR with your correction.

3.1.2 Contribute to Weblate documentation

You are welcome to improve the documentation page of your choice. Do it easily by clicking the *Edit on GitHub* button in the top-right corner of the page.

Please respect these guidelines while writing:

1. Don't remove part of the documentation if it's valid.
2. Use clear and easily-understandable language. You are writing tech docs, not a poem. Not all docs readers are native speakers, be thoughtful.

3. Don't be affraid to ask if you are not certain. If you have to ask about some feature while editing, don't change its docs before you have the answer. This means: You change or ask. Don't do both at the same time.
4. Verify your changes by performing described actions while following the docs.
5. Send PR with changes in small chunks to make it easier and quicker to review and merge.
6. If you want to rewrite and change the structure of a big article, do it in two steps:
    1. Rewrite
    2. Once the rewrite is reviewed, polished, and merged, change the structure of the paragraphs in another PR.

Sugestie: You can translate the docs.

3.1.3 Extending built-in language definitions

The language definitions are in the weblate-language-data repository.

You are welcome to add missing language definitions to `languages.csv`, other files are generated from that file.

3.1.4 Weblate discussions

If you have an idea and not sure if it's suitable for an issue, don't worry. You can join the community in GitHub discussions.

3.1.5 Funding Weblate development

You can boost Weblate's development on the donate page. Funds collected there are used to enable gratis hosting for libre software projects and further development of Weblate. Please check the donate page for options, such as funding goals and the rewards you get as a proud funder.

Backers who have funded Weblate

List of Weblate supporters:

- Yashiro Ccs
- Cheng-Chia Tseng
- Timon Reinhard
- Cassidy James
- Loic Dachary
- Marozed
- https://freedombox.org/
- GNU Solidario (GNU Health)
- BallotReady
- Richard Nespithal
- MyExpenses.Mobi

Do you want to be in the list? Please see options on the Donate to Weblate.

3.2 Starting contributing code to Weblate

Understand the Weblate source code by going through *Weblate source code*, *Weblate frontend* and *Weblate internals*.

3.2.1 Starting with the codebase

Familiarize yourself with the Weblate codebase, by having a go at the bugs labelled good first issue.

3.2.2 Running Weblate locally

The most comfortable approach to get started with Weblate development is to follow *Installing from sources*. It will get you a virtualenv with editable Weblate sources.

1. Clone the Weblate source code:

```
git clone https://github.com/WeblateOrg/weblate.git
cd weblate
```

2. Create a virtualenv:

```
virtualenv .venv
.venv/bin/activate
```

3. Install Weblate (for this you need some system dependencies, see *Installing from sources*):

```
pip install -e .
```

3. Install all dependencies useful for development:

```
pip install -r requirements-dev.txt
```

4. Start a development server:

```
weblate runserver
```

5. Depending on your configuration, you might also want to start Celery workers:

```
./weblate/examples/celery start
```

6. To run a test (see *Local testing* for more details):

```
. scripts/test-database
./manage.py test
```

Vezi și:

Installing from sources

3.2.3 Running Weblate locally in Docker

If you have Docker and docker-compose installed, you can spin up the development environment by simply running:

```
./rundev.sh
```

It will create a development Docker image and start it. Weblate is running on <http://127.0.0.1:8080/> and you can sign in as the user `admin` using `admin` as the password. The new installation is empty, so you might want to continue with *Adding translation projects and components*.

The `Dockerfile` and `docker-compose.yml` for this are located in the `dev-docker` directory.

The script also accepts some parameters, to execute tests, run it with the `test` parameter and then specify any `test` parameters, for example running only tests in the `weblate.machine` module:

```
./rundev.sh test --failfast weblate.machine
```

Notă: Be careful that your Docker containers are up and running before running the tests. You can check that by running the `docker ps` command.

To display the logs:

```
./rundev.sh logs
```

To stop the background containers, run:

```
./rundev.sh stop
```

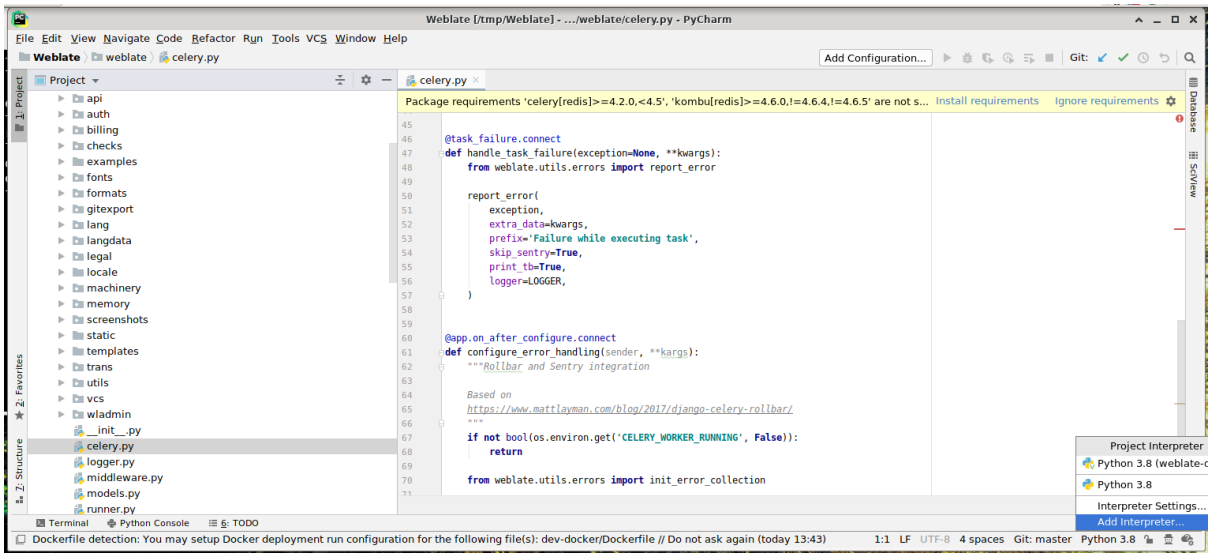
Running the script without arguments will re-create the Docker container and restart it.

Notă: This is not a suitable setup for production, as it includes several hacks which are insecure, but they make development easier.

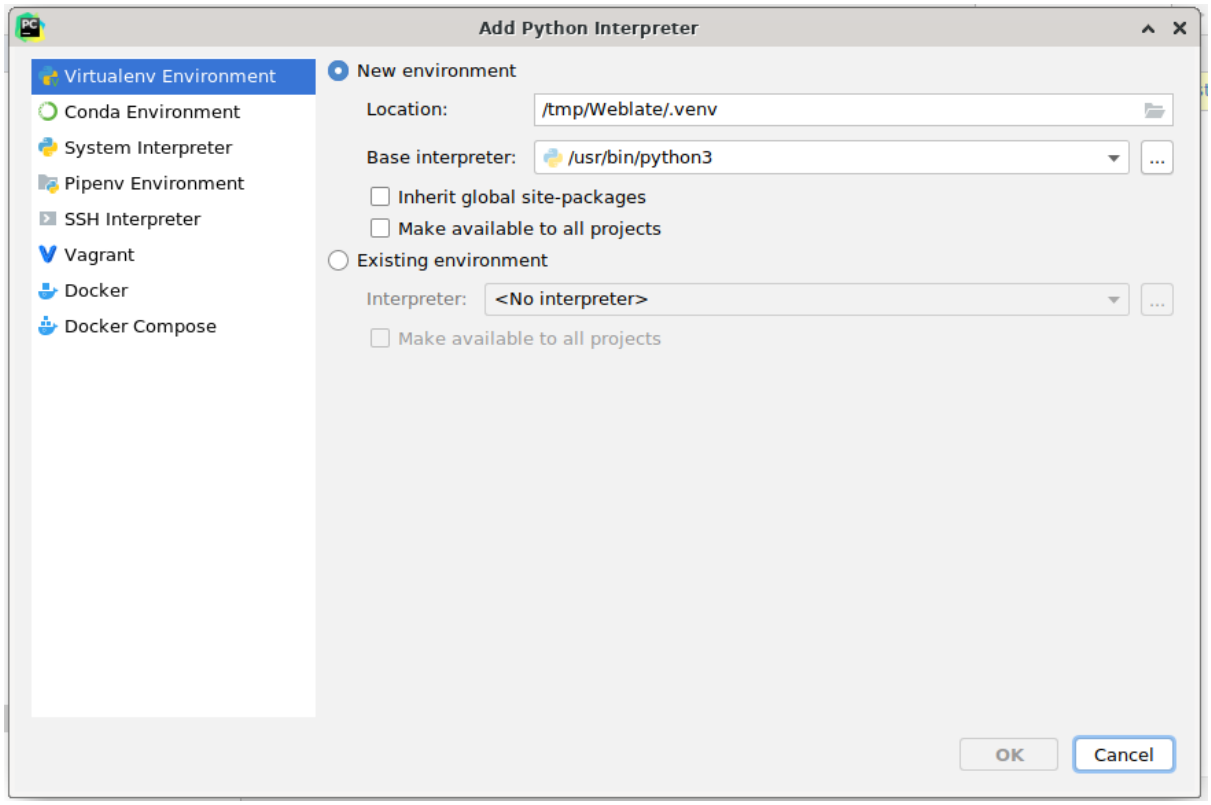
3.2.4 Coding Weblate with PyCharm

PyCharm is a known IDE for Python, here are some guidelines to help you set up your Weblate project in it.

Considering you have just cloned the GitHub repository to a folder, just open it with PyCharm. Once the IDE is open, the first step is to specify the interpreter you want to use:

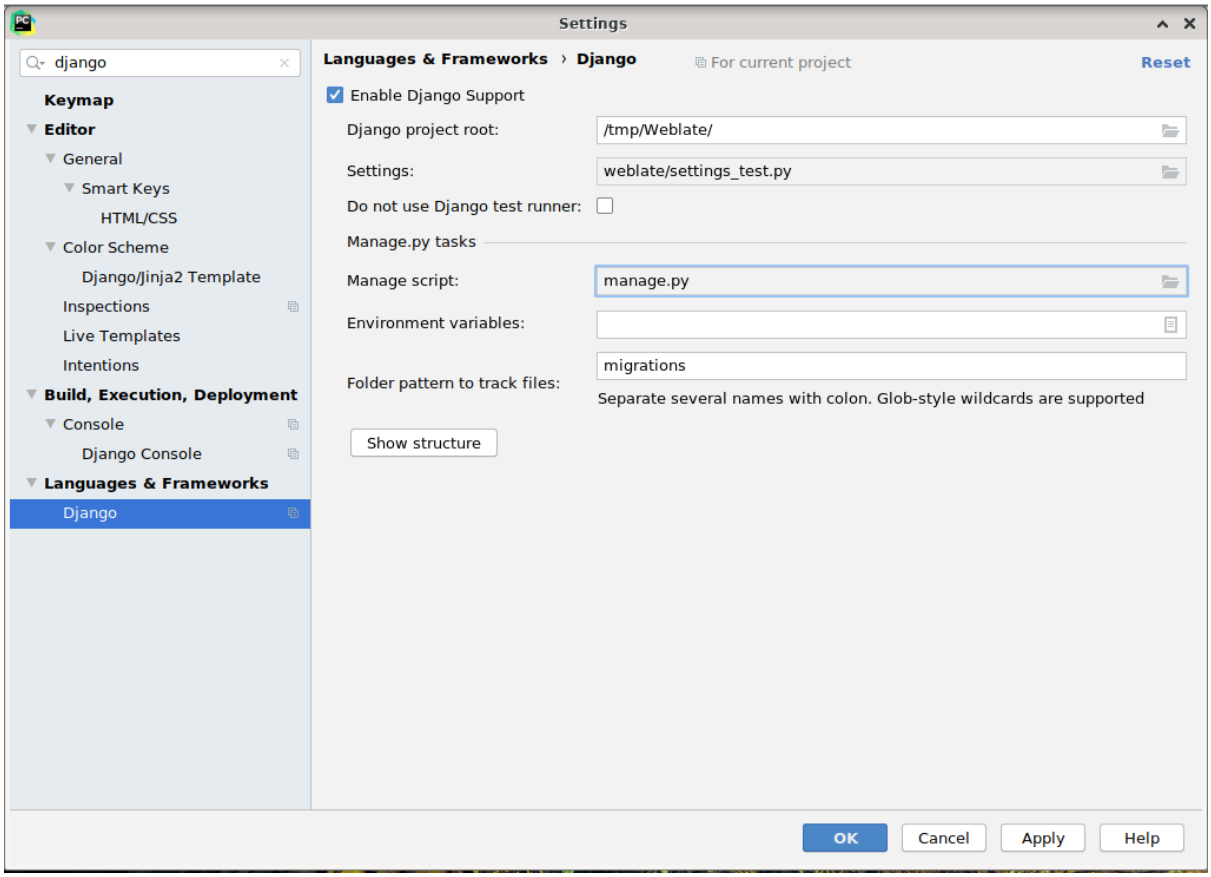


You can either choose to let PyCharm create the virtualenv for you, or select an already existing one:



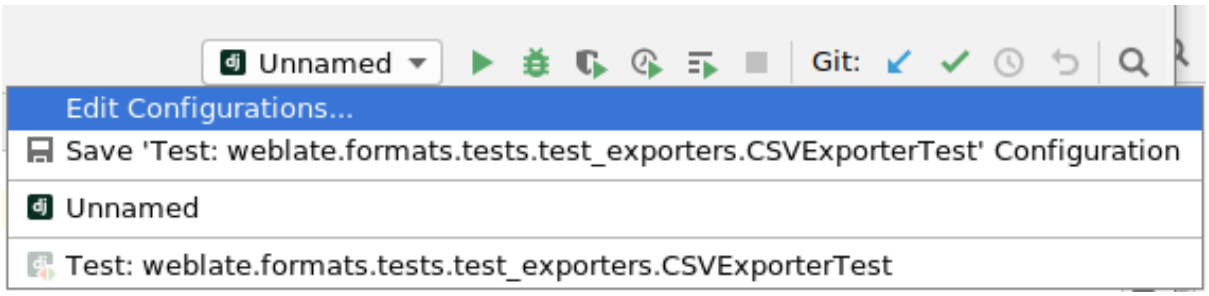
Don't forget to install the dependencies once the interpreter is set: Either through the console (the console from the IDE will directly use your virtualenv by default), or through the interface when you get a warning about missing dependencies.

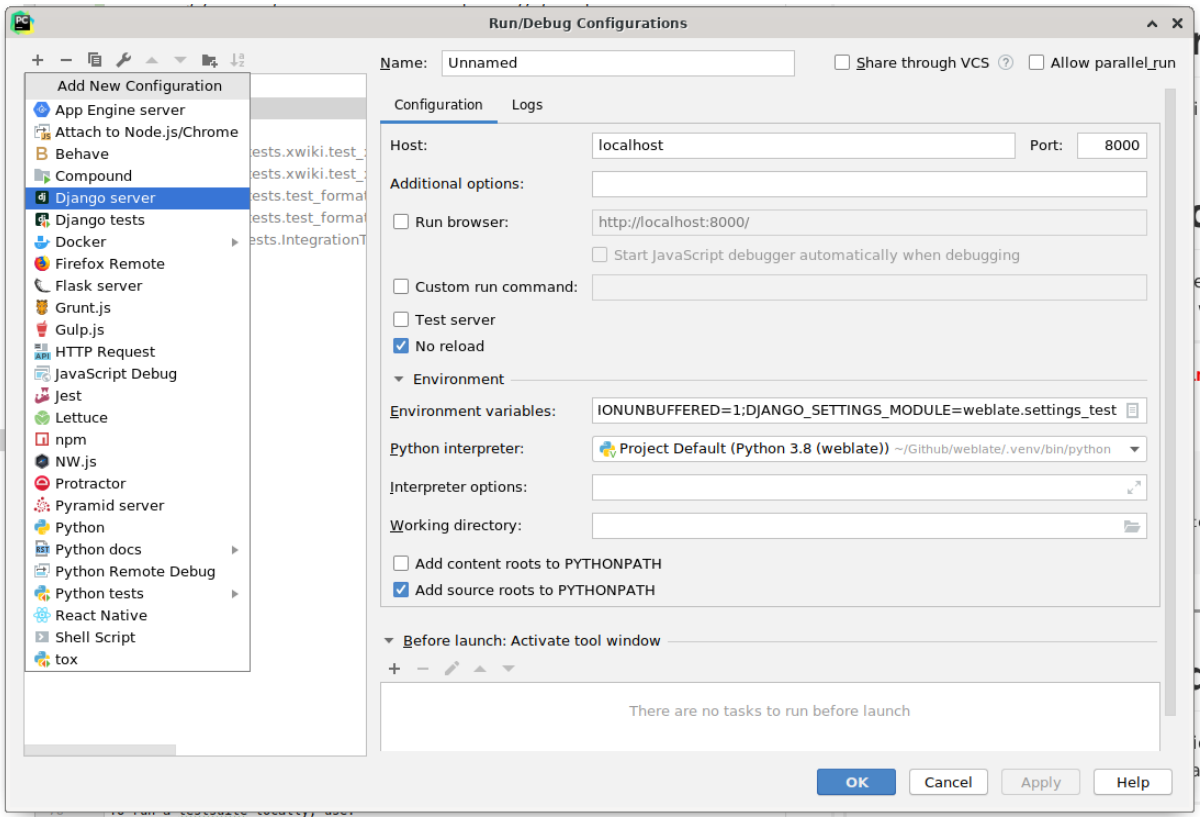
The second step is to set the right info to use Django natively inside PyCharm: The idea is to be able to immediately trigger the unit tests in the IDE. For that you need to specify the root path of the Django project and the path to its settings:



Be careful, the *Django project root* is the actual root of the repository, not the Weblate sub-directory. About the settings, you could use the `weblate/settings_test.py` from the repository, but you could create your own setting and set it there.

The last step is to run the server and to put breakpoints in the code to be able to debug it. This is done by creating a new *Django Server* configuration:





Sugestie: Be careful with the property called *No reload*: It prevents the server from being reloaded live if you modify files. This allows the existing debugger breakpoints to persist, when they normally would be discarded upon reloading the server.

3.2.5 Bootstrapping your devel instance

You might want to use `import_demo` to create demo translations and `createadmin` to make an admin user.

3.3 Weblate source code

Weblate is developed on GitHub. You are welcome to fork the code and open pull requests. Patches in any other form are welcome too.

Vezi și:

Check out *Weblate internals* to see how Weblate looks from inside.

3.3.1 Security by Design Principles

Any code for Weblate should be written with Security by Design Principles in mind.

3.3.2 Coding standard

The code should follow PEP-8 coding guidelines and should be formatted using **black** code formatter.

To check the code quality, you can use **flake8**, the recommended plugins are listed in `.pre-commit-config.yaml` and its configuration is placed in `setup.cfg`.

The easiest approach to enforce all this is to install pre-commit. Weblate repository contains configuration for it to verify the committed files are sane. After installing it (it is already included in the `requirements-lint.txt`) turn it on by running `pre-commit install` in Weblate checkout. This way all your changes will be automatically checked.

You can also trigger check manually, to check all files run:

```
pre-commit run --all
```

3.4 Debugging Weblate

Bugs can behave as application crashes or as misbehavior. You are welcome to collect info on any such issue and submit it to the issue tracker.

3.4.1 Debug mode

Turning on debug mode will make the exceptions show in the browser. This is useful to debug issues in the web interface, but not suitable for production environment as it has performance consequences and might leak private data.

Vezi și:

Disable debug mode

3.4.2 Weblate logs

Weblate can produce detailed logs of what is going in the background. In the default configuration it uses syslog and that makes the log appear either in `/var/log/messages` or `/var/log/syslog` (depending on your syslog daemon configuration).

The Celery process (see *Background tasks using Celery*) usually produces own logs as well. The example system-wide setups log to several files under `/var/log/celery/`.

Docker containers log to their output (as usual in the Docker world), so you can look at the logs using `docker-compose logs`.

Vezi și:

Sample configuration contains `LOGGING` configuration.

3.4.3 Not processing background tasks

Lot of things happen in background Celery workers. In case things like sending out e-mails or component removal does not work, there might be some issue with it.

Things to check in that case:

- Check Celery process is running, see *Background tasks using Celery*
- Check Celery queue status either in *Interfața de gestionare* or using `celery_queues`
- Look into Celery logs for errors (see *Weblate logs*)

3.4.4 Not receiving e-mails from Weblate

You can verify whether outgoing e-mail is working correctly by using the `sendtestemail` management command (see *Invoking management commands* for instructions on how to invoke it in different environments) or using *Interfața de gestionare* under the *Tools* tab.

These send e-mail directly, so this verifies that your SMTP configuration is correct (see *Configuring outgoing e-mail*). Most of the e-mails from Weblate are however sent in the background and there might be some issues with Celery involved as well, please see *Not processing background tasks* for debugging that.

3.4.5 Analyzing application crashes

In case the application crashes, it is useful to collect as much info about the crash as possible. The easiest way to achieve this is by using third-party services which can collect such info automatically. You can find info on how to set this up in *Collecting error reports*.

3.4.6 Silent failures

Lots of tasks are offloaded to Celery for background processing. Failures are not shown in the user interface, but appear in the Celery logs. Configuring *Collecting error reports* helps you to notice such failures easier.

3.4.7 Performance issues

In case Weblate performs badly in some situation, please collect the relevant logs showing the issue, and anything that might help figuring out where the code might be improved.

In case some requests take too long without any indication, you might want to install dogslow along with *Collecting error reports* and get pinpointed and detailed tracebacks in the error collection tool.

3.5 Weblate internals

Notă: This chapter will give you basic overview of Weblate internals.

Weblate derives most of its code structure from, and is based on Django.

3.5.1 Directory structure

Quick overview of directory structure of Weblate main repository:

`docs` Source code for this documentation, which can be built using Sphinx.

`dev-docker` Docker code to run development server, see *Running Weblate locally in Docker*.

`weblate` Source code of Weblate as a Django application, see *Weblate internals*.

`weblate/static` Client files (CSS, Javascript and images), see *Weblate frontend*.

3.5.2 Modules

Weblate consists of several Django applications (some optional, see *Optional Weblate modules*):

`accounts`

User account, profiles and notifications.

`addons`

Add-ons to tweak Weblate behavior, see *Add-ons*.

`api`

API based on Django REST framework.

`auth`

Authentication and permissions.

`billing`

The optional *Facturare* module.

`checks`

Translation string *Quality checks* module.

`fonts`

Font rendering checks module.

`formats`

File format abstraction layer based on translate-toolkit.

`gitexport`

The optional *Git exporter* module.

`lang`

Module defining language and plural models.

`legal`

The optional *Legal* module.

`machinery`

Integration of machine translation services.

`memory`

Built-in translation memory, see *Memorie de traducere*.

`screenshots`

Screenshots management and OCR module.

`trans`

Main module handling translations.

`utils`

Various helper utilities.

`vcs`

Version control system abstraction.

`wladmin`

Django admin interface customization.

3.6 Developing addons

Add-ons are way to customize localization workflow in Weblate.

class `weblate.addons.base.`**BaseAddon**(*storage=None*)

classmethod can_install(*component*, *user*)
Check whether add-on is compatible with given component.

configure(*settings*)
Save configuration.

daily(*component*)
Hook triggered daily.

classmethod get_add_form(*user*, *component*, ***kwargs*)
Return configuration form for adding new add-on.

get_settings_form(*user*, ***kwargs*)
Return configuration form for this add-on.

post_add(*translation*)
Hook triggered after new translation is added.

post_commit(*component*)
Hook triggered after changes are committed to the repository.

post_push(*component*)
Hook triggered after repository is pushed upstream.

post_update(*component*, *previous_head: str*, *skip_push: bool*)
Hook triggered after repository is updated from upstream.

Parametrii

- **previous_head** (*str*) – HEAD of the repository prior to update, can be blank on initial clone.
- **skip_push** (*bool*) – Whether the add-on operation should skip pushing changes upstream. Usually you can pass this to underlying methods as `commit_and_push` or `commit_pending`.

pre_commit(*translation*, *author*)
Hook triggered before changes are committed to the repository.

pre_push(*component*)
Hook triggered before repository is pushed upstream.

pre_update(*component*)
Hook triggered before repository is updated from upstream.

save_state()
Save add-on state information.

stay_on_create = False
Base class for Weblate add-ons.

store_post_load(*translation*, *store*)
Hook triggered after a file is parsed.

It receives an instance of a file format class as a argument.

This is useful to modify file format class parameters, for example adjust how the file will be saved.

unit_pre_create(*unit*)
Hook triggered before new unit is created.

Here is an example addon:

```
#
# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#


from django.utils.translation import gettext_lazy as _

from weblate.addons.base import BaseAddon
from weblate.addons.events import EVENT_PRE_COMMIT


class ExampleAddon(BaseAddon):
    # Filter for compatible components, every key is
    # matched against property of component
    compat = {"file_format": {"po", "po-mono"}}
    # List of events add-on should receive
    events = (EVENT_PRE_COMMIT,)
    # Add-on unique identifier
    name = "weblate.example.example"
    # Verbose name shown in the user interface
    verbose = _("Example add-on")
    # Detailed add-on description
    description = _("This add-on does nothing it is just an example.")

    # Callback to implement custom behavior
    def pre_commit(self, translation, author):
        return
```

3.7 Weblate frontend

The frontend is currently built using Bootstrap, jQuery and few third party libraries.

3.7.1 Supported browsers

Weblate supports the latest, stable releases of all major browsers and platforms.

Alternative browsers which use the latest version of WebKit, Blink, or Gecko, whether directly or via the platform's web view API, are not explicitly supported. However, Weblate should (in most cases) display and function correctly in these browsers as well.

Older browsers might work, but some features might be limited.

3.7.2 Dependency management

The yarn package manager is used to update third party libraries. The configuration lives in `scripts/yarn` and there is a wrapper script `scripts/yarn-update` to upgrade the libraries, build them and copy to correct locations in `weblate/static/vendor`, where all third partly frontend code is located. The Weblate specific code should be placed directly in `weblate/static` or feature specific subdirectories (for example `weblate/static/editor`).

Adding new third-party library typically consists of:

```
# Add a yarn package
yarn --cwd scripts/yarn add PACKAGE
# Edit the script to copy package to the static folder
edit scripts/yarn-update
# Run the update script
./scripts/yarn-update
# Add files to git
git add .
```

3.7.3 Coding style

Weblate relies on Prettier for the code formatting for both JavaScript and CSS files.

We also use ESLint to check the JavaScript code.

3.7.4 Localizare

Should you need any user visible text in the frontend code, it should be localizable. In most cases all you need is to wrap your text inside `gettext` function, but there are more complex features available:

```
document.write(gettext('this is to be translated'));

var object_count = 1 // or 0, or 2, or 3, ...
s = ngettext('literal for the singular case',
        'literal for the plural case', object_count);

fmts = ngettext('There is %s object. Remaining: %s',
        'There are %s objects. Remaining: %s', 11);
s = interpolate(fmts, [11, 20]);
// s is 'There are 11 objects. Remaining: 20'
```

Vezi și:

Translation topic in the Django documentation

3.7.5 Icons

Weblate currently uses material design icons. In case you are looking for new symbol, check Material Design Icons or Material Design Resources.

Additionally, there is `scripts/optimize-svg` to reduce size of the SVG as most of the icons are embedded inside the HTML to allow styling of the paths.

3.8 Reporting issues in Weblate

Weblate issue tracker is hosted at GitHub.

Feel welcome to report any issues you have, or suggest improvement for Weblate there. There are various templates prepared to comfortably guide you through the issue report.

If what you have found is a security issue in Weblate, please consult the *Security issues* section below.

If you are not sure about your bug report or feature request, you can try *Weblate discussions*.

3.8.1 Security issues

In order to give the community time to respond and upgrade, you are strongly urged to report all security issues privately. HackerOne is used to handle security issues, and can be reported directly at HackerOne. Once you submit it there, community has limited but enough time to solve the incident.

Alternatively, report to security@weblate.org, which ends up on HackerOne as well.

If you don't want to use HackerOne, for whatever reason, you can send the report by e-mail to michal@cihar.com. You can choose to encrypt it using this PGP key *3CB 1DF1 EF12 CF2A C0EE 5A32 9C27 B313 42B7 511D*. You can also get the PGP key from Keybase.

Notă: Weblate depends on third-party components for many things. In case you find a vulnerability affecting one of those components in general, please report it directly to the respective project.

Some of these are:

- Django
- Django REST framework
- Python Social Auth

3.9 Weblate testsuite and continuous integration

Testsuites exist for most of the current code, increase coverage by adding testcases for any new functionality, and verify that it works.

3.9.1 Continuous integration

Current test results can be found on GitHub Actions and coverage is reported on Codecov.

There are several jobs to verify different aspects:

- Unit tests
- Documentation build and external links
- Migration testing from all supported releases
- Code linting
- Setup verification (ensures that generated dist files do not miss anything and can be tested)

The configuration for the CI is in `.github/workflows` directory. It heavily uses helper scripts stored in `ci` directory. The scripts can be also executed manually, but they require several environment variables, mostly defining Django settings file to use and database connection. The example definition of that is in `scripts/test-database`:

```
# Simple way to configure test database from environment

# Database backend to use postgresql / mysql / mariadb
export CI_DATABASE=${1:-postgresql}

# Database server configuration
export CI_DB_USER=weblate
export CI_DB_PASSWORD=weblate
export CI_DB_HOST=127.0.0.1

# Django settings module to use
export DJANGO_SETTINGS_MODULE=weblate.settings_test
```

The simple execution can look like:

```
. scripts/test-database
./ci/run-migrate
./ci/run-test
./ci/run-docs
```

3.9.2 Local testing

To run a testsuite locally, use:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py test
```

Sugestie: You will need a database (PostgreSQL) server to be used for tests. By default Django creates separate database to run tests with `test_` prefix, so in case your settings is configured to use `weblate`, the tests will use `test_weblate` database. See *Database setup for Weblate* for setup instructions.

The `weblate/settings_test.py` is used in CI environment as well (see *Continuous integration*) and can be tuned using environment variables:

```
# Simple way to configure test database from environment

# Database backend to use postgresql / mysql / mariadb
export CI_DATABASE=${1:-postgresql}

# Database server configuration
export CI_DB_USER=weblate
export CI_DB_PASSWORD=weblate
export CI_DB_HOST=127.0.0.1

# Django settings module to use
export DJANGO_SETTINGS_MODULE=weblate.settings_test
```

Prior to running tests you should collect static files as some tests rely on them being present:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py collectstatic
```

You can also specify individual tests to run:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py test weblate.gitexport
```

Sugestie: The tests can also be executed inside developer docker container, see *Running Weblate locally in Docker*.

Vezi şi:

See Testing in Django for more info on running and writing tests for Django.

3.10 Data schemas

Weblate uses JSON Schema to define layout of external JSON files.

3.10.1 Weblate Translation Memory Schema

https://weblate.org/schemas/weblate-memory.schema.json			
type	*array*		
items	*The Translation Memory Item*		
	type	*object*	
	properties		
	• **category**	*The String Category*	
		1 is global, 2 is shared, 10000000+ are project specific, 20000000+ are user specific	
		type	*integer*
		examples	1
		minimum	0
		default	1
	• **origin**	*The String Origin*	
		Filename or component name	
		type	*string*
		examples	test.tmx
			project/component
		default	
	• **source**	*The Source String*	
		type	*string*
		examples	Hello
		minLength	1
		default	
	• **source_language**	*The Source Language*	
		ISO 639-1 / ISO 639-2 / IETF BCP 47	
		type	*string*
		examples	en
		pattern	^[^]+$
		default	
	• **target**	*The Target String*	
		type	*string*
		examples	Ahoj
		minLength	1
		default	
	• **target_language**	*The Target Language*	
		ISO 639-1 / ISO 639-2 / IETF BCP 47	
		type	*string*
		examples	cs
		pattern	^[^]+$
		default	
	additionalProperties	False	
definitions			

Vezi și:

Memorie de traducere, `dump_memory`, `import_memory`

3.10.2 Weblate user data export

https://weblate.org/schemas/weblate-userdata.schema.json			
type	*object*		
properties			
• **basic**	*Basic*		
	type	*object*	
	properties		
	• **username**	*Username*	
		type	*string*
		examples	admin
		default	
	• **full_name**	*Full name*	
		type	*string*
		examples	Weblate Admin
		default	
	• **email**	*E-mail*	
		type	*string*
		examples	noreply@example.com
		default	
	• **date_joined**	*Date joined*	
		type	*string*
		examples	2019-11-18T18:53:54.862Z
		default	
• **profile**	*Profile*		
	type	*object*	
	properties		
	• **language**	*Language*	
		type	*string*
		examples	cs
		pattern	^.*$
		default	
	• **suggested**	*Number of suggested strings*	
		type	*integer*
		examples	1
		default	0
	• **translated**	*Number of translated strings*	
		type	*integer*
		examples	24
		default	0
	• **uploaded**	*Number of uploaded screenshots*	
		type	*integer*
		examples	1
		default	0
	• **hide_completed**	*Hide completed translations on the dashboard*	
		type	*boolean*
		examples	False
		default	True
	• **secondary_in_zen**	*Show secondary translations in the Zen mode*	
		type	*boolean*
		examples	True
		default	True

continues on next page

Tabelul 2 – continuare din pagina precedentă

	• **hide_source_secondary**	*Hide source if a secondary translation exists*		
		type	*boolean*	
		examples	False	
		default	True	
	• **editor_link**	*Editor link*		
		type	*string*	
		examples		
		pattern	^.*$	
		default		
	• **translate_mode**	*Translation editor mode*		
		type	*integer*	
		examples	0	
		default	0	
	• **zen_mode**	*Zen editor mode*		
		type	*integer*	
		examples	0	
		default	0	
	• **special_chars**	*Special characters*		
		type	*string*	
		examples		
		pattern	^.*$	
		default		
	• **dashboard_view**	*Default dashboard view*		
		type	*integer*	
		examples	1	
		default	0	
	• **dashboard_component_list**	*Default component list*		
		default	null	
		anyOf	type	*null*
			type	*integer*
	• **languages**	*Translated languages*		
		type	*array*	
		default		
		items	*Language code*	
			type	*string*
			examples	cs
			pattern	^.*$
			default	
	• **secondary_languages**	*Secondary languages*		
		type	*array*	
		default		
		items	*Language code*	
			type	*string*
			examples	sk
			pattern	^.*$
			default	
	• **watched**	*Watched projects*		
		type	*array*	
		default		
		items	*Project slug*	
			type	*string*
			examples	weblate
			pattern	^.*$
			default	
• **auditlog**	*Audit log*			
	type	*array*		

continues on next page

Tabelul 2 – continuare din pagina precedentă

	default			
	items	*Items*		
		type	*object*	
		properties		
		• **address**	*IP address*	
			type	*string*
			examples	127.0.0.1
			pattern	^.*$
			default	
		• **user_agent**	*User agent*	
			type	*string*
			examples	PC / Linux / Firefox 70.0
			pattern	^.*$
			default	
		• **timestamp**	*Timestamp*	
			type	*string*
			examples	2019-11-18T18:58:30.845Z
			pattern	^.*$
			default	
		• **activity**	*Activity*	
			type	*string*
			examples	login
			pattern	^.*$
			default	
definitions				

Vezi și:

Profilul utilizatorului, `dumpuserdata`

3.11 Releasing Weblate

3.11.1 Releasing schedule

Weblate has two month release cycle for releases (x.y). These are usually followed by a bunch of bugfix releases to fix issues which slip into them (x.y.z).

The change in the major version indicates that the upgrade process can not skip this version - you always have to upgrade to x.0 before upgrading to higher x.y releases.

Vezi și:

Upgrading Weblate

3.11.2 Release planning

The features for upcoming releases are collected using GitHub milestones, you can see our roadmap at <https://github.com/WeblateOrg/weblate/milestones>.

3.11.3 Release process

Things to check prior to release:

1. Check newly translated languages by **./scripts/list-translated-languages**.
2. Set final version by **./scripts/prepare-release**.
3. Make sure screenshots are up to date **make -C docs update-screenshots**.
4. Merge any possibly pending translations **wlc push; git remote update; git merge origin/weblate**

Perform the release:

5. Create a release **./scripts/create-release --tag** (see below for requirements).

Post release manual steps:

6. Update Docker image.
7. Close GitHub milestone.
8. Once the Docker image is tested, add a tag and push it.
9. Update Helm chart to new version.
10. Include new version in `.github/workflows/migrations.yml` to cover it in migration testing.
11. Increase version in the website download links.
12. Increase version in the repository by **./scripts/set-version**.

To create tags using the **./scripts/create-release** script you will need following:

- GnuPG with private key used to sign the release
- Push access to Weblate git repositories (it pushes tags)
- Configured **hub** tool and access to create releases on the Weblate repo
- SSH access to Weblate download server (the Website downloads are copied there)

3.12 Security and privacy

Sfat: La Weblate, securitatea menține un mediu care prețuiește confidențialitatea utilizatorilor noștri.

Development of Weblate adheres to the Best Practices of the Linux Foundation's Core Infrastructure Initiative.

Vezi și:

Security issues

3.12.1 Tracking dependencies for vulnerabilities

Security issues in our dependencies are monitored using Dependabot. This covers the Python and JavaScript libraries, and the latest stable release has its dependencies updated to avoid vulnerabilities.

Sugestie: There might be vulnerabilities in third-party libraries which do not affect Weblate, so those are not addressed by releasing bugfix versions of Weblate.

3.12.2 Docker container security

The Docker containers are scanned using Anchore and Trivy.

This allows us to detect vulnerabilities early and release improvements quickly.

You can get the results of these scans at GitHub — they are stored as artifacts on our CI in the SARIF format (Static Analysis Results Interchange Format).

Vezi și:

Continuous integration

3.13 Despre Weblate

3.13.1 Project goals

Web-based continuous localization tool with tight *Integrarea controlului versiunilor* supporting a wide range of *Supported file formats*, making it easy for translators to contribute.

3.13.2 Denumire proiect

„Weblate” is a portmanteau of the words „web” and „translate”.

3.13.3 Website-ul proiectului

The landing page is https://weblate.org and there is a cloud-hosted service at https://hosted.weblate.org. The documentation can be read at https://docs.weblate.org.

3.13.4 Project logos

The project logos and other graphics are available in https://github.com/WeblateOrg/graphics.

3.13.5 Leadership

This project is maintained by Michal Čihař, who can be reached at michal@cihar.com.

3.13.6 Authors

Weblate was started by Michal Čihař. Since its inception in 2012, thousands of people have contributed.

3.14 Licență

Copyright (C) 2012 - 2021 Michal Čihař <michal@cihar.com>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

CAPITOLUL 4

Change History

4.1 Weblate 4.8.1

Released on September 10th 2021.

- Fixed user removal in Django admin interface.
- Document add-on parameters in more detail.
- Fixed JavaScript error on glossary.
- Add limit to number of matches in consistency check.
- Improve placeholders handling in machine translations.
- Fixed creating add-ons using API.
- Added `PRIVACY_URL` setting to add privacy policy link to the footer.
- Hide member e-mail addresses from project admins.
- Improved gettext PO merging in case of conflicts.
- Improved glossary highlighting.
- Improved `safe-html` flag behavior with XML checks.
- Fixed commit messages on linked components.

All changes in detail.

4.2 Weblate 4.8

Released on August 21th 2021.

- Added support for Apple stringsdict format.
- The exact search operator is now case-sensitive on PostgreSQL.
- Fixed saving glossary explanations in some cases.
- Documentation improvements.
- Performance improvements.

- Improved squash add-on compatibility with Gerrit.
- Fixed adding strings to monolingual glossary components.
- Improved performance in handling variants.
- Fixed squash add-on sometimes skipping parsing upstream changes.
- Preserve file extension for downloads.
- Added support for the Fluent format.
- Added support for using tabs to indent JSON formats.

All changes in detail.

4.3 Weblate 4.7.2

Released on July 15th 2021.

- Support more language aliases to be configured on a project.
- Fixed search string validation in API.
- Fixed Git exporter URLs after a domain change.
- Fixed cleanup add-on for Windows RC files.
- Fixed possible crash in XLIFF updating.

All changes in detail.

4.4 Weblate 4.7.1

Released on June 30th 2021.

- Improved popup for adding terms to glossary.
- Added support for LibreTranslate machine translation service.
- Added rate limiting on creating new projects.
- Improved performance of file updates.

All changes in detail.

4.5 Weblate 4.7

Released on June 17th 2021.

- Improved configuration health check.
- Added support for `object-pascal-format` used in gettext PO, see *Object Pascal format*.
- Renamed *Nearby keys* to *Similar keys* to better describe the purpose.
- Added support for *mi18n lang files*.
- Improved SAML authentication integration.
- Fixed *Gerrit* integration to better handle corner cases.
- Weblate now requires Django 3.2.
- Fixed inviting users when e-mail authentication is disabled.

- Improved language definitions.
- Added support for blocking users from contributing to a project.
- Fixed automatic creation of glossary languages.
- Extended documentation about add-ons.
- Performance improvements for components with linked repositories.
- Added support for free DeepL API.
- The user management no longer needs Django admin interface.

All changes in detail.

4.6 Weblate 4.6.2

Released on May 8th 2021.

- Fixed crash after moving shared component between projects.
- Fixed adding new strings to empty properties files.
- Fixed copy icon alignment in RTL languages.
- Extended string statistics on the Info tab.
- Fixed handling of translation files ignored in Git.
- Improved metrics performance.
- Fixed possible bug in saving glossaries.
- Fixed consistency check behavior on languages with different plural rules.

All changes in detail.

4.7 Weblate 4.6.1

Released on May 2nd 2021.

- Remove obsolete spam protection code.
- Improve source plural check accuracy.
- Update list of user interface languages in Docker.
- Improved error messages when creating pull requests.
- Fixed creating pull requests on Pagure.
- Fixed triggering automatically installed add-ons.
- Fixed possible caching issues on upgrade.
- Fixed adding new units to monolingual translations using upload.

All changes in detail.

4.8 Weblate 4.6

Released on April 19th 2021.

- The auto_translate management command has now a parameter for specifying translation mode.
- Added support for *Text files*.
- Added trends and metrics for all objects.
- Added support for direct copying text from secondary languages.
- Added date filtering when browsing changes.
- Improved activity charts.
- Sender for contact form e-mails can now be configured.
- Improved parameters validation in component creation API.
- The rate limiting no longer applies to superusers.
- Improved automatic translation add-on performance and reliability.
- The rate limiting now can be customized in the Docker container.
- API for creating components now automatically uses *Weblate internal URLs*.
- Simplified state indication while listing strings.
- Password hashing now uses Argon2 by default.
- Simplified progress bars indicating translation status.
- Renamed *Add missing languages* to clarify the purpose.
- Fixed saving string state to XLIFF.
- Added language-wide search.
- Initial support for *Scaling horizontally* the Docker deployment.

All changes in detail.

4.9 Weblate 4.5.3

Released on April 1st 2021.

- Fixed metrics collection.
- Fixed possible crash when adding strings.
- Improved search query examples.
- Fixed possible loss of newly added strings on replace upload.

4.10 Weblate 4.5.2

Released on March 26th 2021.

- Configurable schedule for automatic translation.
- Added Lua format check.
- Ignore format strings in the *Cuvinte consecutive duplicate* check.
- Allow uploading screenshot from a translate page.

- Added forced file synchronization to the repository maintenance.
- Fixed automatic suggestions for languages with a longer code.
- Improved performance when adding new strings.
- Several bug fixes in quality checks.
- Several performance improvements.
- Added integration with *Descoperiți Weblate*.
- Fixed checks behavior with read-only strings.

All changes in detail.

4.11 Weblate 4.5.1

Released on March 5th 2021.

- Fixed editing of glossary flags in some corner cases.
- Extend metrics usage to improve performance of several pages.
- Store correct source language in TMX files.
- Better handling for uploads of monolingual PO using API.
- Improved alerts behavior on glossary components.
- Improved Markdown link checks.
- Indicate glossary and source language in breadcrumbs.
- Paginated component listing of huge projects.
- Improved performance of translation, component or project removal.
- Improved bulk edit performance.
- Fixed preserving „Needs editing" and „Approved" states for ODF files.
- Improved interface for customizing translation-file downloads

All changes in detail.

4.12 Weblate 4.5

Released on February 19th 2021.

- Added support for `lua-format` used in gettext PO.
- Added support for sharing a component between projects.
- Fixed multiple unnamed variables check behavior with multiple format flags.
- Dropped mailing list field on the project in favor of generic instructions for translators.
- Added pseudolocale generation add-on.
- Added support for TermBase eXchange files.
- Added support for manually defining string variants using a flag.
- Improved performance of consistency checks.
- Improved performance of translation memory for long strings.
- Added support for searching in explanations.

- Strings can now be added and removed in bilingual formats as well.
- Extend list of supported languages in Amazon Translate machine translation.
- Automatically enable Java MessageFormat checks for Java Properties.
- Added a new upload method to add new strings to a translation.
- Added a simple interface to browse translation.
- Glossaries are now stored as regular components.
- Dropped specific API for glossaries as component API is used now.
- Added simplified interface to toggle some of the flags.
- Added support for non-translatable or forbidden terms in the glossary.
- Added support for defining terminology in a glossary.
- Moved text direction toggle to get more space for the visual keyboard.
- Added option to automatically watch projects user-contributed to.
- Added check whether translation matches the glossary.
- Added support for customizing navigation text color.

All changes in detail.

4.13 Weblate 4.4.2

Released on January 14th 2021.

- Fixed corruption of one distributed MO file.

4.14 Weblate 4.4.1

Released on January 13th 2021.

- Fixed reverting plural changes.
- Fixed displaying help for project settings.
- Improved administration of users.
- Improved handling of context in monolingual PO files.
- Fixed cleanup add-on behavior with HTML, ODF, IDML and Windows RC formats.
- Fixed parsing of location from CSV files.
- Use content compression for file downloads.
- Improved user experience on importing from ZIP file.
- Improved detection of file format for uploads.
- Avoid duplicate pull requests on Pagure.
- Improved performance when displaying ghost translations.
- Reimplemented translation editor to use native browser textarea.
- Fixed cleanup add-on breaking adding new strings.
- Added API for add-ons.

All changes in detail.

4.15 Weblate 4.4

Released on December 15th 2020.

- Improved validation when creating a component.
- Weblate now requires Django 3.1.
- Added support for appearance customization in the management interface.
- Fixed read-only state handling in bulk edit.
- Improved CodeMirror integration.
- Added add-on to remove blank strings from translation files.
- The CodeMirror editor is now used for translations.
- Syntax highlighting in translation editor for XML, HTML, Markdown and reStructuredText.
- Highlight placeables in translation editor.
- Improved support for non-standard language codes.
- Added alert when using ambiguous language codes.
- The user is now presented with a filtered list of languages when adding a new translation.
- Extended search capabilities for changes in history.
- Improved billing detail pages and libre hosting workflow.
- Extended translation statistics API.
- Improved „other translations" tab while translating.
- Added tasks API.
- Improved performance of file upload.
- Improved display of user defined special characters.
- Improved performance of auto-translation.
- Several minor improvements in the user interface.
- Improved naming of ZIP downloads.
- Added option for getting notifications on unwatched projects.

  All changes in detail.

4.16 Weblate 4.3.2

Released on November 4th 2020.

- Fixed crash on certain component filemasks.
- Improved accuracy of the consecutive duplicated words check.
- Added support for Pagure pull requests.
- Improved error messages for failed registrations.
- Reverted rendering developer comments as Markdown.
- Simplified setup of Git repositories with different default branch than „master".
- Newly created internal repositories now use main as the default branch.
- Reduced false positives rate of unchanged translation while translating reStructuredText.

- Fixed CodeMirror display issues in some situations.
- Renamed Template group to „Sources” to clarify its meaning.
- Fixed GitLab pull requests on repositories with longer paths.

All changes in detail.

4.17 Weblate 4.3.1

Released on October 21st 2020.

- Improved auto-translation performance.
- Fixed session expiry for authenticated users.
- Add support for hiding version information.
- Improve hooks compatibility with Bitbucket Server.
- Improved performance of translation memory updates.
- Reduced memory usage.
- Improved performance of Matrix view.
- Added confirmation before removing a user from a project.

  All changes in detail.

4.18 Weblate 4.3

Released on October 15th 2020.

- Include user stats in the API.
- Fixed component ordering on paginated pages.
- Define source language for a glossary.
- Rewritten support for GitHub and GitLab pull requests.
- Fixed stats counts after removing suggestion.
- Extended public user profile.
- Fixed configuration of enforced checks.
- Improve documentation about built-in backups.
- Moved source language attribute from project to a component.
- Add Vue I18n formatting check.
- Generic placeholders check now supports regular expressions.
- Improved look of Matrix mode.
- Machinery is now called automatic suggestions.
- Added support for interacting with multiple GitLab or GitHub instances.
- Extended API to cover project updates, unit updates and removals and glossaries.
- Unit API now properly handles plural strings.
- Component creation can now handle ZIP file or document upload.
- Consolidated API response status codes.

- Support Markdown in contributor agreement.
- Improved source strings tracking.
- Improved JSON, YAML and CSV formats compatibility.
- Added support for removing strings.
- Improved performance of file downloads.
- Improved repository management view.
- Automatically enable java-format for Android.
- Added support for localized screenshots.
- Added support for Python 3.9.
- Fixed translating HTML files under certain conditions.

All changes in detail.

4.19 Weblate 4.2.2

Released on September 2nd 2020.

- Fixed matching of source strings for JSON formats.
- Fixed login redirect for some authentication configurations.
- Fixed LDAP authentication with group sync.
- Fixed crash in reporting automatic translation progress.
- Fixed Git commit squashing with trailers enabled.
- Fixed creating local VCS components using API.

4.20 Weblate 4.2.1

Released on August 21st 2020.

- Fixed saving plurals for some locales in Android resources.
- Fixed crash in the cleanup add-on for some XLIFF files.
- Allow setting up localization CDN in Docker image.

4.21 Weblate 4.2

Released on August 18th 2020.

- Improved user pages and added listing of users.
- Dropped support for migrating from 3.x releases, migrate through 4.1 or 4.0.
- Added exports into several monolingual formats.
- Improved activity charts.
- Number of displayed nearby strings can be configured.
- Added support for locking components experiencing repository errors.
- Simplified main navigation (replaced buttons with icons).

- Improved language code handling in Google Translate integration.
- The Git squash add-on can generate `Co-authored-by:` trailers.
- Improved query search parser.
- Improved user feedback from format strings checks.
- Improved performance of bulk state changes.
- Added compatibility redirects after project or component renaming.
- Added notifications for strings approval, component locking and license change.
- Added support for ModernMT.
- Allow to avoid overwriting approved translations on file upload.
- Dropped support for some compatibility URL redirects.
- Added check for ECMAScript template literals.
- Added option to watch a component.
- Removed leading dot from JSON unit keys.
- Removed separate Celery queue for translation memory.
- Allow translating all components a language at once.
- Allow to configure `Content-Security-Policy` HTTP headers.
- Added support for aliasing languages at project level.
- New add-on to help with HTML or JavaScript localization, see *CDN de localizare JavaScript*.
- The Weblate domain is now configured in the settings, see `SITE_DOMAIN`.
- Add support for searching by component and project.

4.22 Weblate 4.1.1

Released on June 19th 2020.

- Fixed changing autofix or add-ons configuration in Docker.
- Fixed possible crash in „About” page.
- Improved installation of byte-compiled locale files.
- Fixed adding words to glossary.
- Fixed keyboard shortcuts for machinery.
- Removed debugging output causing discarding log events in some setups.
- Fixed lock indication on project listing.
- Fixed listing GPG keys in some setups.
- Added option for which DeepL API version to use.
- Added support for acting as SAML Service Provider, see *Autentificare SAML*.

4.23 Weblate 4.1

Released on June 15th 2020.

- Added support for creating new translations with included country code.
- Added support for searching source strings with screenshot.
- Extended info available in the stats insights.
- Improved search editing on „Translate” pages.
- Improve handling of concurrent repository updates.
- Include source language in project creation form.
- Include changes count in credits.
- Fixed UI language selection in some cases.
- Allow to whitelist registration methods with registrations closed.
- Improved lookup of related terms in glossary.
- Improved translation memory matches.
- Group same machinery results.
- Add direct link to edit screenshot from translate page.
- Improved removal confirmation dialog.
- Include templates in ZIP download.
- Add support for Markdown and notification configuration in announcements.
- Extended details in check listings.
- Added support for new file formats: *Laravel PHP șiruri de caractere*, *HTML files*, *OpenDocument Format*, *IDML Format*, *Windows RC files*, *INI translations*, *Inno Setup INI translations*, *Proprietăți GWT*, *go-i18n JSON files*, *ARB File*.
- Consistently use dismissed as state of dismissed checks.
- Add support for configuring default add-ons to enable.
- Fixed editor keyboard shortcut to dismiss checks.
- Improved machine translation of strings with placeholders.
- Show ghost translation for user languages to ease starting them.
- Improved language code parsing.
- Show translations in user language first in the list.
- Renamed shapings to more generic name variants.
- Added new quality checks: *Variabile multiple fără nume*, *Lungă netradusă*, *Cuvinte consecutive duplicate*.
- Reintroduced support for wiping translation memory.
- Fixed option to ignore source checks.
- Added support for configuring different branch for pushing changes.
- API now reports rate limiting status in the HTTP headers.
- Added support for Google Translate V3 API (Advanced).
- Added ability to restrict access on component level.
- Added support for whitespace and other special chars in translation flags, see *Personalizarea comportamentului cu ajutorul steguleților*.

- Always show rendered text check if enabled.
- API now supports filtering of changes.
- Added support for sharing glossaries between projects.

4.24 Weblate 4.0.4

Released on May 7th 2020.

- Fixed testsuite execution on some Python 3.8 environments.
- Typo fixes in the documentation.
- Fixed creating components using API in some cases.
- Fixed JavaScript errors breaking mobile navigation.
- Fixed crash on displaying some checks.
- Fixed screenshots listing.
- Fixed monthly digest notifications.
- Fixed intermediate translation behavior with units non existing in translation.

4.25 Weblate 4.0.3

Released on May 2nd 2020.

- Fixed possible crash in reports.
- User mentions in comments are now case insensitive.
- Fixed PostgreSQL migration for non superusers.
- Fixed changing the repository URL while creating component.
- Fixed crash when upstream repository is gone.

4.26 Weblate 4.0.2

Released on April 27th 2020.

- Improved performance of translation stats.
- Improved performance of changing labels.
- Improved bulk edit performance.
- Improved translation memory performance.
- Fixed possible crash on component deletion.
- Fixed displaying of translation changes in some corner cases.
- Improved warning about too long celery queue.
- Fixed possible false positives in the consistency check.
- Fixed deadlock when changing linked component repository.
- Included edit distance in changes listing and CSV and reports.
- Avoid false positives of punctuation spacing check for Canadian French.

- Fixed XLIFF export with placeholders.
- Fixed false positive with zero width check.
- Improved reporting of configuration errors.
- Fixed bilingual source upload.
- Automatically detect supported languages for DeepL machine translation.
- Fixed progress bar display in some corner cases.
- Fixed some checks triggering on non translated strings.

4.27 Weblate 4.0.1

Released on April 16th 2020.

- Fixed package installation from PyPI.

4.28 Weblate 4.0

Released on April 16th 2020.

- Weblate now requires Python 3.6 or newer.
- Added management overview of component alerts.
- Added component alert for broken repository browser URLs.
- Improved sign in and registration pages.
- Project access control and workflow configuration integrated to project settings.
- Added check and highlighter for i18next interpolation and nesting.
- Added check and highlighter for percent placeholders.
- Display suggestions failing checks.
- Record source string changes in history.
- Upgraded Microsoft Translator to version 3 API.
- Reimplemented translation memory backend.
- Added support for several `is:` lookups in *Searching*.
- Allow to make *Traducere neschimbată* avoid internal blacklist.
- Improved comments extraction from monolingual po files.
- Renamed whiteboard messages to announcements.
- Fixed occasional problems with registration mails.
- Improved LINGUAS update add-on to handle more syntax variants.
- Fixed editing monolingual XLIFF source file.
- Added support for exact matching in *Searching*.
- Extended API to cover screenshots, users, groups, componentlists and extended creating projects.
- Add support for source upload on bilingual translations.
- Added support for intermediate language from developers.
- Added support for source strings review.

- Extended download options for platform wide translation memory.

4.29 Weblate 3.x series

4.29.1 Weblate 3.11.3

Released on March 11th 2020.

- Fixed searching for fields with certain priority.
- Fixed predefined query for recently added strings.
- Fixed searching returning duplicate matches.
- Fixed notifications rendering in Gmail.
- Fixed reverting changes from the history.
- Added links to events in digest notifications.
- Fixed email for account removal confirmation.
- Added support for Slack authentication in Docker container.
- Avoid sending notifications for not subscribed languages.
- Include Celery queues in performance overview.
- Fixed documentation links for add-ons.
- Reduced false negatives for unchanged translation check.
- Raised bleach dependency to address CVE-2020-6802.
- Fixed listing project level changes in history.
- Fixed stats invalidation in some corner cases.
- Fixed searching for certain string states.
- Improved format string checks behavior on missing percent.
- Fixed authentication using some third party providers.

4.29.2 Weblate 3.11.2

Released on February 22nd 2020.

- Fixed rendering of suggestions.
- Fixed some strings wrongly reported as having no words.

4.29.3 Weblate 3.11.1

Released on February 20th 2020.

- Documented Celery setup changes.
- Improved filename validation on component creation.
- Fixed minimal versions of some dependencies.
- Fixed adding groups with certain Django versions.
- Fixed manual pushing to upstream repository.
- Improved glossary matching.

4.29.4 Weblate 3.11

Released on February 17th 2020.

- Allow using VCS push URL during component creation via API.
- Rendered width check now shows image with the render.
- Fixed links in notifications e-mails.
- Improved look of plaintext e-mails.
- Display ignored checks and allow to make them active again.
- Display nearby keys on monolingual translations.
- Added support for grouping string shapings.
- Recommend upgrade to new Weblate versions in the system checks.
- Provide more detailed analysis for duplicate language alert.
- Include more detailed license info on the project pages.
- Automatically unshallow local copies if needed.
- Fixed download of strings needing action.
- New alert to warn about using the same filemask twice.
- Improve XML placeables extraction.
- The `SINGLE_PROJECT` can now enforce redirection to chosen project.
- Added option to resolve comments.
- Added bulk editing of flags.
- Added support for labels.
- Added bulk edit add-on.
- Added option for *Executarea controalelor*.
- Increased default validity of confirmation links.
- Improved Matomo integration.
- Fixed *A fost tradus* to correctly handle source string change.
- Extended automatic updates configuration by `AUTO_UPDATE`.
- LINGUAS add-ons now do full sync of translations in Weblate.

4.29.5 Weblate 3.10.3

Released on January 18th 2020.

- Support for translate-toolkit 2.5.0.

4.29.6 Weblate 3.10.2

Released on January 18th 2020.

- Add lock indication to projects.
- Fixed CSS bug causing flickering in some web browsers.
- Fixed searching on systems with non-English locales.
- Improved repository matching for GitHub and Bitbucket hooks.
- Fixed data migration on some Python 2.7 installations.
- Allow configuration of Git shallow cloning.
- Improved background notification processing.
- Fixed broken form submission when navigating back in web browser.
- New add-on to configure YAML formatting.
- Fixed same plurals check to not fire on single plural form languages.
- Fixed regex search on some fields.

4.29.7 Weblate 3.10.1

Released on January 9th 2020.

- Extended API with translation creation.
- Fixed several corner cases in data migrations.
- Compatibility with Django 3.0.
- Improved data clean-up performance.
- Added support for customizable security.txt.
- Improved breadcrumbs in changelog.
- Improved translations listing on dashboard.
- Improved HTTP responses for webhooks.
- Added support for GitLab merge requests in Docker container.

4.29.8 Weblate 3.10

Released on December 20th 2019.

- Improved application user interface.
- Added doublespace check.
- Fixed creating new languages.
- Avoid sending auditlog notifications to deleted e-mails.
- Added support for read only strings.
- Added support for Markdown in comments.
- Allow placing translation instruction text in project info.
- Add copy to clipboard for secondary languages.
- Improved support for Mercurial.
- Improved Git repository fetching performance.

- Add search lookup for age of string.
- Show source language for all translations.
- Show context for nearby strings.
- Added support for notifications on repository operations.
- Improved translation listings.
- Extended search capabilities.
- Added support for automatic translation strings marked for editing.
- Avoid sending duplicate notifications for linked component alerts.
- Improve default merge request message.
- Better indicate string state in Zen mode.
- Added support for more languages in Yandex Translate.
- Improved look of notification e-mails.
- Provide choice for translation license.

4.29.9 Weblate 3.9.1

Released on October 28th 2019.

- Remove some unneeded files from backups.
- Fixed potential crash in reports.
- Fixed cross database migration failure.
- Added support for force pushing Git repositories.
- Reduced risk of registration token invalidation.
- Fixed account removal hitting rate limiter.
- Added search based on priority.
- Fixed possible crash on adding strings to JSON file.
- Safe HTML check and fixup now honor source string markup.
- Avoid sending notifications to invited and deleted users.
- Fix SSL connection to redis in Celery in Docker container.

4.29.10 Weblate 3.9

Released on October 15th 2019.

- Include Weblate metadata in downloaded files.
- Improved UI for failing checks.
- Indicate missing strings in format checks.
- Separate check for French punctuation spacing.
- Add support for fixing some of quality checks errors.
- Add separate permission to create new projects.
- Extend stats for char counts.
- Improve support for Java style language codes.

- Added new generic check for placeholders.
- Added support for WebExtension JSON placeholders.
- Added support for flat XML format.
- Extended API with project, component and translation removal and creation.
- Added support for Gitea and Gitee webhooks.
- Added new custom regex based check.
- Allow to configure contributing to shared translation memory.
- Added ZIP download for more translation files.
- Make XLIFF standard compliant parsing of maxwidth and font.
- Added new check and fixer for safe HTML markup for translating web applications.
- Add component alert on unsupported configuration.
- Added automatic translation add-on to bootstrap translations.
- Extend automatic translation to add suggestions.
- Display add-on parameters on overview.
- Sentry is now supported through modern Sentry SDK instead of Raven.
- Changed example settings to be better fit for production environment.
- Added automated backups using BorgBackup.
- Split cleanup add-on for RESX to avoid unwanted file updates.
- Added advanced search capabilities.
- Allow users to download their own reports.
- Added localization guide to help configuring components.
- Added support for GitLab merge requests.
- Improved display of repository status.
- Perform automated translation in the background.

4.29.11 Weblate 3.8

Released on August 15th 2019.

- Added support for simplified creating of similar components.
- Added support for parsing translation flags from the XML based file formats.
- Log exceptions into Celery log.
- Improve performance of repository scoped add-ons.
- Improved look of notification e-mails.
- Fixed password reset behavior.
- Improved performance on most of translation pages.
- Fixed listing of languages not known to Weblate.
- Add support for cloning add-ons to discovered components.
- Add support for replacing file content with uploaded.
- Add support for translating non VCS based content.
- Added OpenGraph widget image to use on social networks.

- Added support for animated screenshots.
- Improved handling of monolingual XLIFF files.
- Avoid sending multiple notifications for single event.
- Add support for filtering changes.
- Extended predefined periods for reporting.
- Added webhook support for Azure Repos.
- New opt-in notifications on pending suggestions or untranslated strings.
- Add one click unsubscribe link to notification e-mails.
- Fixed false positives with Has been translated check.
- New management interface for admins.
- String priority can now be specified using flags.
- Added language management views.
- Add checks for Qt library and Ruby format strings.
- Added configuration to better fit single project installations.
- Notify about new string on source string change on monolingual translations.
- Added separate view for translation memory with search capability.

4.29.12 Weblate 3.7.1

Released on June 28th 2019.

- Documentation updates.
- Fixed some requirements constraints.
- Updated language database.
- Localization updates.
- Various user interface tweaks.
- Improved handling of unsupported but discovered translation files.
- More verbosely report missing file format requirements.

4.29.13 Weblate 3.7

Released on June 21st 2019.

- Added separate Celery queue for notifications.
- Use consistent look with application for API browsing.
- Include approved stats in the reports.
- Report progress when updating translation component.
- Allow to abort running background component update.
- Extend template language for filename manipulations.
- Use templates for editor link and repository browser URL.
- Indicate max length and current characters count when editing translation.
- Improved handling of abbreviations in unchanged translation check.

- Refreshed landing page for new contributors.
- Add support for configuring msgmerge add-on.
- Delay opening SMTP connection when sending notifications.
- Improved error logging.
- Allow custom location in MO generating add-on.
- Added add-ons to cleanup old suggestions or comments.
- Added option to enable horizontal mode in the Zen editor.
- Improved import performance with many linked components.
- Fixed examples installation in some cases.
- Improved rendering of alerts in changes.
- Added new horizontal stats widget.
- Improved format strings check on plurals.
- Added font management tool.
- New check for rendered text dimensions.
- Added support for subtitle formats.
- Include overall completion stats for languages.
- Added reporting at project and global scope.
- Improved user interface when showing translation status.
- New Weblate logo and color scheme.
- New look of bitmap badges.

4.29.14 Weblate 3.6.1

Released on April 26th 2019.

- Improved handling of monolingual XLIFF files.
- Fixed digest notifications in some corner cases.
- Fixed add-on script error alert.
- Fixed generating MO file for monolingual PO files.
- Fixed display of uninstalled checks.
- Indicate administered projects on project listing.
- Allow update to recover from missing VCS repository.

4.29.15 Weblate 3.6

Released on April 20th 2019.

- Add support for downloading user data.
- Add-ons are now automatically triggered upon installation.
- Improved instructions for resolving merge conflicts.
- Cleanup add-on is now compatible with app store metadata translations.
- Configurable language code syntax when adding new translations.

- Warn about using Python 2 with planned termination of support in April 2020.
- Extract special characters from the source string for visual keyboard.
- Extended contributor stats to reflect both source and target counts.
- Admins and consistency add-ons can now add translations even if disabled for users.
- Fixed description of toggle disabling `Language-Team` header manipulation.
- Notify users mentioned in comments.
- Removed file format autodetection from component setup.
- Fixed generating MO file for monolingual PO files.
- Added digest notifications.
- Added support for muting component notifications.
- Added notifications for new alerts, whiteboard messages or components.
- Notifications for administered projects can now be configured.
- Improved handling of three letter language codes.

4.29.16 Weblate 3.5.1

Released on March 10th 2019.

- Fixed Celery systemd unit example.
- Fixed notifications from HTTP repositories with login.
- Fixed race condition in editing source string for monolingual translations.
- Include output of failed add-on execution in the logs.
- Improved validation of choices for adding new language.
- Allow to edit file format in component settings.
- Update installation instructions to prefer Python 3.
- Performance and consistency improvements for loading translations.
- Make Microsoft Terminology service compatible with current Zeep releases.
- Localization updates.

4.29.17 Weblate 3.5

Released on March 3rd 2019.

- Improved performance of built-in translation memory.
- Added interface to manage global translation memory.
- Improved alerting on bad component state.
- Added user interface to manage whiteboard messages.
- Add-on commit message now can be configured.
- Reduce number of commits when updating upstream repository.
- Fixed possible metadata loss when moving component between projects.
- Improved navigation in the Zen mode.
- Added several new quality checks (Markdown related and URL).

- Added support for app store metadata files.
- Added support for toggling GitHub or Gerrit integration.
- Added check for Kashida letters.
- Added option to squash commits based on authors.
- Improved support for XLSX file format.
- Compatibility with Tesseract 4.0.
- Billing add-on now removes projects for unpaid billings after 45 days.

4.29.18 Weblate 3.4

Released on January 22nd 2019.

- Added support for XLIFF placeholders.
- Celery can now utilize multiple task queues.
- Added support for renaming and moving projects and components.
- Include characters counts in reports.
- Added guided adding of translation components with automatic detection of translation files.
- Customizable merge commit messages for Git.
- Added visual indication of component alerts in navigation.
- Improved performance of loading translation files.
- New add-on to squash commits prior to push.
- Improved displaying of translation changes.
- Changed default merge style to rebase and made that configurable.
- Better handle private use subtags in language code.
- Improved performance of fulltext index updates.
- Extended file upload API to support more parameters.

4.29.19 Weblate 3.3

Released on November 30th 2018.

- Added support for component and project removal.
- Improved performance for some monolingual translations.
- Added translation component alerts to highlight problems with a translation.
- Expose XLIFF string resname as context when available.
- Added support for XLIFF states.
- Added check for non writable files in DATA_DIR.
- Improved CSV export for changes.

4.29.20 Weblate 3.2.2

Released on October 20th 2018.

- Remove no longer needed Babel dependency.
- Updated language definitions.
- Improve documentation for add-ons, LDAP and Celery.
- Fixed enabling new dos-eol and auto-java-messageformat flags.
- Fixed running setup.py test from PyPI package.
- Improved plurals handling.
- Fixed translation upload API failure in some corner cases.
- Fixed updating Git configuration in case it was changed manually.

4.29.21 Weblate 3.2.1

Released on October 10th 2018.

- Document dependency on backports.csv on Python 2.7.
- Fix running tests under root.
- Improved error handling in gitexport module.
- Fixed progress reporting for newly added languages.
- Correctly report Celery worker errors to Sentry.
- Fixed creating new translations with Qt Linguist.
- Fixed occasional fulltext index update failures.
- Improved validation when creating new components.
- Added support for cleanup of old suggestions.

4.29.22 Weblate 3.2

Released on October 6th 2018.

- Add install_addon management command for automated add-on installation.
- Allow more fine grained ratelimit settings.
- Added support for export and import of Excel files.
- Improve component cleanup in case of multiple component discovery add-ons.
- Rewritten Microsoft Terminology machine translation backend.
- Weblate now uses Celery to offload some processing.
- Improved search capabilities and added regular expression search.
- Added support for Youdao Zhiyun API machine translation.
- Added support for Baidu API machine translation.
- Integrated maintenance and cleanup tasks using Celery.
- Improved performance of loading translations by almost 25%.
- Removed support for merging headers on upload.
- Removed support for custom commit messages.

- Configurable editing mode (zen/full).
- Added support for error reporting to Sentry.
- Added support for automated daily update of repositories.
- Added support for creating projects and components by users.
- Built-in translation memory now automatically stores translations done.
- Users and projects can import their existing translation memories.
- Better management of related strings for screenshots.
- Added support for checking Java MessageFormat.

See 3.2 milestone on GitHub for detailed list of addressed issues.

4.29.23 Weblate 3.1.1

Released on July 27th 2018.

- Fix testsuite failure on some setups.

4.29.24 Weblate 3.1

Released on July 27th 2018.

- Upgrades from older version than 3.0.1 are not supported.
- Allow to override default commit messages from settings.
- Improve webhooks compatibility with self hosted environments.
- Added support for Amazon Translate.
- Compatibility with Django 2.1.
- Django system checks are now used to diagnose problems with installation.
- Removed support for soon shutdown libravatar service.
- New add-on to mark unchanged translations as needing edit.
- Add support for jumping to specific location while translating.
- Downloaded translations can now be customized.
- Improved calculation of string similarity in translation memory matches.
- Added support by signing Git commits by GnuPG.

4.29.25 Weblate 3.0.1

Released on June 10th 2018.

- Fixed possible migration issue from 2.20.
- Localization updates.
- Removed obsolete hook examples.
- Improved caching documentation.
- Fixed displaying of admin documentation.
- Improved handling of long language names.

4.29.26 Weblate 3.0

Released on June 1st 2018.

- Rewritten access control.
- Several code cleanups that lead to moved and renamed modules.
- New add-on for automatic component discovery.
- The import_project management command has now slightly different parameters.
- Added basic support for Windows RC files.
- New add-on to store contributor names in PO file headers.
- The per component hook scripts are removed, use add-ons instead.
- Add support for collecting contributor agreements.
- Access control changes are now tracked in history.
- New add-on to ensure all components in a project have same translations.
- Support for more variables in commit message templates.
- Add support for providing additional textual context.

4.30 Weblate 2.x series

4.30.1 Weblate 2.20

Released on April 4th 2018.

- Improved speed of cloning subversion repositories.
- Changed repository locking to use third party library.
- Added support for downloading only strings needing action.
- Added support for searching in several languages at once.
- New add-on to configure gettext output wrapping.
- New add-on to configure JSON formatting.
- Added support for authentication in API using RFC 6750 compatible Bearer authentication.
- Added support for automatic translation using machine translation services.
- Added support for HTML markup in whiteboard messages.
- Added support for mass changing state of strings.
- Translate-toolkit at least 2.3.0 is now required, older versions are no longer supported.
- Added built-in translation memory.
- Added componentlists overview to dashboard and per component list overview pages.
- Added support for DeepL machine translation service.
- Machine translation results are now cached inside Weblate.
- Added support for reordering committed changes.

4.30.2 Weblate 2.19.1

Released on February 20th 2018.

- Fixed migration issue on upgrade from 2.18.
- Improved file upload API validation.

4.30.3 Weblate 2.19

Released on February 15th 2018.

- Fixed imports across some file formats.
- Display human friendly browser information in audit log.
- Added TMX exporter for files.
- Various performance improvements for loading translation files.
- Added option to disable access management in Weblate in favor of Django one.
- Improved glossary lookup speed for large strings.
- Compatibility with django_auth_ldap 1.3.0.
- Configuration errors are now stored and reported persistently.
- Honor ignore flags in whitespace autofixer.
- Improved compatibility with some Subversion setups.
- Improved built-in machine translation service.
- Added support for SAP Translation Hub service.
- Added support for Microsoft Terminology service.
- Removed support for advertisement in notification e-mails.
- Improved translation progress reporting at language level.
- Improved support for different plural formulas.
- Added support for Subversion repositories not using stdlayout.
- Added add-ons to customize translation workflows.

4.30.4 Weblate 2.18

Released on December 15th 2017.

- Extended contributor stats.
- Improved configuration of special characters virtual keyboard.
- Added support for DTD file format.
- Changed keyboard shortcuts to less likely collide with browser/system ones.
- Improved support for approved flag in XLIFF files.
- Added support for not wrapping long strings in gettext PO files.
- Added button to copy permalink for current translation.
- Dropped support for Django 1.10 and added support for Django 2.0.
- Removed locking of translations while translating.
- Added support for adding new strings to monolingual translations.

- Added support for translation workflows with dedicated reviewers.

4.30.5 Weblate 2.17.1

Released on October 13th 2017.

- Fixed running testsuite in some specific situations.
- Locales updates.

4.30.6 Weblate 2.17

Released on October 13th 2017.

- Weblate by default does shallow Git clones now.
- Improved performance when updating large translation files.
- Added support for blocking certain e-mails from registration.
- Users can now delete their own comments.
- Added preview step to search and replace feature.
- Client side persistence of settings in search and upload forms.
- Extended search capabilities.
- More fine grained per project ACL configuration.
- Default value of BASE_DIR has been changed.
- Added two step account removal to prevent accidental removal.
- Project access control settings is now editable.
- Added optional spam protection for suggestions using Akismet.

4.30.7 Weblate 2.16

Released on August 11th 2017.

- Various performance improvements.
- Added support for nested JSON format.
- Added support for WebExtension JSON format.
- Fixed git exporter authentication.
- Improved CSV import in certain situations.
- Improved look of Other translations widget.
- The max-length checks is now enforcing length of text in form.
- Make the commit_pending age configurable per component.
- Various user interface cleanups.
- Fixed component/project/site wide search for translations.

4.30.8 Weblate 2.15

Released on June 30th 2017.

- Show more related translations in other translations.
- Add option to see translations of current string to other languages.
- Use 4 plural forms for Lithuanian by default.
- Fixed upload for monolingual files of different format.
- Improved error messages on failed authentication.
- Keep page state when removing word from glossary.
- Added direct link to edit secondary language translation.
- Added Perl format quality check.
- Added support for rejecting reused passwords.
- Extended toolbar for editing RTL languages.

4.30.9 Weblate 2.14.1

Released on May 24th 2017.

- Fixed possible error when paginating search results.
- Fixed migrations from older versions in some corner cases.
- Fixed possible CSRF on project watch and unwatch.
- The password reset no longer authenticates user.
- Fixed possible CAPTCHA bypass on forgotten password.

4.30.10 Weblate 2.14

Released on May 17th 2017.

- Add glossary entries using AJAX.
- The logout now uses POST to avoid CSRF.
- The API key token reset now uses POST to avoid CSRF.
- Weblate sets Content-Security-Policy by default.
- The local editor URL is validated to avoid self-XSS.
- The password is now validated against common flaws by default.
- Notify users about important activity with their account such as password change.
- The CSV exports now escape potential formulas.
- Various minor improvements in security.
- The authentication attempts are now rate limited.
- Suggestion content is stored in the history.
- Store important account activity in audit log.
- Ask for password confirmation when removing account or adding new associations.
- Show time when suggestion has been made.
- There is new quality check for trailing semicolon.

- Ensure that search links can be shared.
- Included source string information and screenshots in the API.
- Allow to overwrite translations through API upload.

4.30.11 Weblate 2.13.1

Released on Apr 12th 2017.

- Fixed listing of managed projects in profile.
- Fixed migration issue where some permissions were missing.
- Fixed listing of current file format in translation download.
- Return HTTP 404 when trying to access project where user lacks privileges.

4.30.12 Weblate 2.13

Released on Apr 12th 2017.

- Fixed quality checks on translation templates.
- Added quality check to trigger on losing translation.
- Add option to view pending suggestions from user.
- Add option to automatically build component lists.
- Default dashboard for unauthenticated users can be configured.
- Add option to browse 25 random strings for review.
- History now indicates string change.
- Better error reporting when adding new translation.
- Added per language search within project.
- Group ACLs can now be limited to certain permissions.
- The per project ALCs are now implemented using Group ACL.
- Added more fine grained privileges control.
- Various minor UI improvements.

4.30.13 Weblate 2.12

Released on Mar 3rd 2017.

- Improved admin interface for groups.
- Added support for Yandex Translate API.
- Improved speed of site wide search.
- Added project and component wide search.
- Added project and component wide search and replace.
- Improved rendering of inconsistent translations.
- Added support for opening source files in local editor.
- Added support for configuring visual keyboard with special characters.
- Improved screenshot management with OCR support for matching source strings.

- Default commit message now includes translation information and URL.
- Added support for Joomla translation format.
- Improved reliability of import across file formats.

4.30.14 Weblate 2.11

Released on Jan 31st 2017.

- Include language detailed information on language page.
- Mercurial backend improvements.
- Added option to specify translation component priority.
- More consistent usage of Group ACL even with less used permissions.
- Added WL_BRANCH variable to hook scripts.
- Improved developer documentation.
- Better compatibility with various Git versions in Git exporter add-on.
- Included per project and component stats.
- Added language code mapping for better support of Microsoft Translate API.
- Moved fulltext cleanup to background job to make translation removal faster.
- Fixed displaying of plural source for languages with single plural form.
- Improved error handling in import_project.
- Various performance improvements.

4.30.15 Weblate 2.10.1

Released on Jan 20th 2017.

- Do not leak account existence on password reset form (CVE-2017-5537).

4.30.16 Weblate 2.10

Released on Dec 15th 2016.

- Added quality check to check whether plurals are translated differently.
- Fixed GitHub hooks for repositories with authentication.
- Added optional Git exporter module.
- Support for Microsoft Cognitive Services Translator API.
- Simplified project and component user interface.
- Added automatic fix to remove control characters.
- Added per language overview to project.
- Added support for CSV export.
- Added CSV download for stats.
- Added matrix view for quick overview of all translations.
- Added basic API for changes and strings.
- Added support for Apertium APy server for machine translations.

4.30.17 Weblate 2.9

Released on Nov 4th 2016.

- Extended parameters for createadmin management command.
- Extended import_json to be able to handle with existing components.
- Added support for YAML files.
- Project owners can now configure translation component and project details.
- Use „Watched” instead of „Subscribed” projects.
- Projects can be watched directly from project page.
- Added multi language status widget.
- Highlight secondary language if not showing source.
- Record suggestion deletion in history.
- Improved UX of languages selection in profile.
- Fixed showing whiteboard messages for component.
- Keep preferences tab selected after saving.
- Show source string comment more prominently.
- Automatically install Gettext PO merge driver for Git repositories.
- Added search and replace feature.
- Added support for uploading visual context (screenshots) for translations.

4.30.18 Weblate 2.8

Released on Aug 31st 2016.

- Documentation improvements.
- Translations.
- Updated bundled javascript libraries.
- Added list_translators management command.
- Django 1.8 is no longer supported.
- Fixed compatibility with Django 1.10.
- Added Subversion support.
- Separated XML validity check from XML mismatched tags.
- Fixed API to honor HIDE_REPO_CREDENTIALS settings.
- Show source change in Zen mode.
- Alt+PageUp/PageDown/Home/End now works in Zen mode as well.
- Add tooltip showing exact time of changes.
- Add option to select filters and search from translation page.
- Added UI for translation removal.
- Improved behavior when inserting placeables.
- Fixed auto locking issues in Zen mode.

4.30.19 Weblate 2.7

Released on Jul 10th 2016.

- Removed Google web translate machine translation.
- Improved commit message when adding translation.
- Fixed Google Translate API for Hebrew language.
- Compatibility with Mercurial 3.8.
- Added import_json management command.
- Correct ordering of listed translations.
- Show full suggestion text, not only a diff.
- Extend API (detailed repository status, statistics, …).
- Testsuite no longer requires network access to test repositories.

4.30.20 Weblate 2.6

Released on Apr 28th 2016.

- Fixed validation of components with language filter.
- Improved support for XLIFF files.
- Fixed machine translation for non English sources.
- Added REST API.
- Django 1.10 compatibility.
- Added categories to whiteboard messages.

4.30.21 Weblate 2.5

Released on Mar 10th 2016.

- Fixed automatic translation for project owners.
- Improved performance of commit and push operations.
- New management command to add suggestions from command line.
- Added support for merging comments on file upload.
- Added support for some GNU extensions to C printf format.
- Documentation improvements.
- Added support for generating translator credits.
- Added support for generating contributor stats.
- Site wide search can search only in one language.
- Improve quality checks for Armenian.
- Support for starting translation components without existing translations.
- Support for adding new translations in Qt TS.
- Improved support for translating PHP files.
- Performance improvements for quality checks.
- Fixed site wide search for failing checks.

- Added option to specify source language.
- Improved support for XLIFF files.
- Extended list of options for import_project.
- Improved targeting for whiteboard messages.
- Support for automatic translation across projects.
- Optimized fulltext search index.
- Added management command for auto translation.
- Added placeables highlighting.
- Added keyboard shortcuts for placeables, checks and machine translations.
- Improved translation locking.
- Added quality check for AngularJS interpolation.
- Added extensive group based ACLs.
- Clarified terminology on strings needing edit (formerly fuzzy).
- Clarified terminology on strings needing action and not translated strings.
- Support for Python 3.
- Dropped support for Django 1.7.
- Dropped dependency on msginit for creating new gettext PO files.
- Added configurable dashboard views.
- Improved notifications on parse errors.
- Added option to import components with duplicate name to import_project.
- Improved support for translating PHP files.
- Added XLIFF export for dictionary.
- Added XLIFF and gettext PO export for all translations.
- Documentation improvements.
- Added support for configurable automatic group assignments.
- Improved adding of new translations.

4.30.22 Weblate 2.4

Released on Sep 20th 2015.

- Improved support for PHP files.
- Ability to add ACL to anonymous user.
- Improved configurability of import_project command.
- Added CSV dump of history.
- Avoid copy/paste errors with whitespace characters.
- Added support for Bitbucket webhooks.
- Tighter control on fuzzy strings on translation upload.
- Several URLs have changed, you might have to update your bookmarks.
- Hook scripts are executed with VCS root as current directory.
- Hook scripts are executed with environment variables describing current component.

- Add management command to optimize fulltext index.
- Added support for error reporting to Rollbar.
- Projects now can have multiple owners.
- Project owners can manage themselves.
- Added support for `javascript-format` used in gettext PO.
- Support for adding new translations in XLIFF.
- Improved file format autodetection.
- Extended keyboard shortcuts.
- Improved dictionary matching for several languages.
- Improved layout of most of pages.
- Support for adding words to dictionary while translating.
- Added support for filtering languages to be managed by Weblate.
- Added support for translating and importing CSV files.
- Rewritten handling of static files.
- Direct login/registration links to third-party service if that's the only one.
- Commit pending changes on account removal.
- Add management command to change site name.
- Add option to configure default committer.
- Add hook after adding new translation.
- Add option to specify multiple files to add to commit.

4.30.23 Weblate 2.3

Released on May 22nd 2015.

- Dropped support for Django 1.6 and South migrations.
- Support for adding new translations when using Java Property files.
- Allow to accept suggestion without editing.
- Improved support for Google OAuth 2.0.
- Added support for Microsoft .resx files.
- Tuned default robots.txt to disallow big crawling of translations.
- Simplified workflow for accepting suggestions.
- Added project owners who always receive important notifications.
- Allow to disable editing of monolingual template.
- More detailed repository status view.
- Direct link for editing template when changing translation.
- Allow to add more permissions to project owners.
- Allow to show secondary language in Zen mode.
- Support for hiding source string in favor of secondary language.

4.30.24 Weblate 2.2

Released on Feb 19th 2015.

- Performance improvements.
- Fulltext search on location and comments fields.
- New SVG/javascript based activity charts.
- Support for Django 1.8.
- Support for deleting comments.
- Added own SVG badge.
- Added support for Google Analytics.
- Improved handling of translation filenames.
- Added support for monolingual JSON translations.
- Record component locking in a history.
- Support for editing source (template) language for monolingual translations.
- Added basic support for Gerrit.

4.30.25 Weblate 2.1

Released on Dec 5th 2014.

- Added support for Mercurial repositories.
- Replaced Glyphicon font by Awesome.
- Added icons for social authentication services.
- Better consistency of button colors and icons.
- Documentation improvements.
- Various bugfixes.
- Automatic hiding of columns in translation listing for small screens.
- Changed configuration of filesystem paths.
- Improved SSH keys handling and storage.
- Improved repository locking.
- Customizable quality checks per source string.
- Allow to hide completed translations from dashboard.

4.30.26 Weblate 2.0

Released on Nov 6th 2014.

- New responsive UI using Bootstrap.
- Rewritten VCS backend.
- Documentation improvements.
- Added whiteboard for site wide messages.
- Configurable strings priority.
- Added support for JSON file format.

- Fixed generating mo files in certain cases.
- Added support for GitLab notifications.
- Added support for disabling translation suggestions.
- Django 1.7 support.
- ACL projects now have user management.
- Extended search possibilities.
- Give more hints to translators about plurals.
- Fixed Git repository locking.
- Compatibility with older Git versions.
- Improved ACL support.
- Added buttons for per language quotes and other special characters.
- Support for exporting stats as JSONP.

4.31 Weblate 1.x series

4.31.1 Weblate 1.9

Released on May 6th 2014.

- Django 1.6 compatibility.
- No longer maintained compatibility with Django 1.4.
- Management commands for locking/unlocking translations.
- Improved support for Qt TS files.
- Users can now delete their account.
- Avatars can be disabled.
- Merged first and last name attributes.
- Avatars are now fetched and cached server side.
- Added support for shields.io badge.

4.31.2 Weblate 1.8

Released on November 7th 2013.

- Please check manual for upgrade instructions.
- Nicer listing of project summary.
- Better visible options for sharing.
- More control over anonymous users privileges.
- Supports login using third party services, check manual for more details.
- Users can login by e-mail instead of username.
- Documentation improvements.
- Improved source strings review.
- Searching across all strings.

- Better tracking of source strings.
- Captcha protection for registration.

4.31.3 Weblate 1.7

Released on October 7th 2013.

- Please check manual for upgrade instructions.
- Support for checking Python brace format string.
- Per component customization of quality checks.
- Detailed per translation stats.
- Changed way of linking suggestions, checks and comments to strings.
- Users can now add text to commit message.
- Support for subscribing on new language requests.
- Support for adding new translations.
- Widgets and charts are now rendered using Pillow instead of Pango + Cairo.
- Add status badge widget.
- Dropped invalid text direction check.
- Changes in dictionary are now logged in history.
- Performance improvements for translating view.

4.31.4 Weblate 1.6

Released on July 25th 2013.

- Nicer error handling on registration.
- Browsing of changes.
- Fixed sorting of machine translation suggestions.
- Improved support for MyMemory machine translation.
- Added support for Amagama machine translation.
- Various optimizations on frequently used pages.
- Highlights searched phrase in search results.
- Support for automatic fixups while saving the message.
- Tracking of translation history and option to revert it.
- Added support for Google Translate API.
- Added support for managing SSH host keys.
- Various form validation improvements.
- Various quality checks improvements.
- Performance improvements for import.
- Added support for voting on suggestions.
- Cleanup of admin interface.

4.31.5 Weblate 1.5

Released on April 16th 2013.

- Please check manual for upgrade instructions.
- Added public user pages.
- Better naming of plural forms.
- Added support for TBX export of glossary.
- Added support for Bitbucket notifications.
- Activity charts are now available for each translation, language or user.
- Extended options of import_project admin command.
- Compatible with Django 1.5.
- Avatars are now shown using libravatar.
- Added possibility to pretty print JSON export.
- Various performance improvements.
- Indicate failing checks or fuzzy strings in progress bars for projects or languages as well.
- Added support for custom pre-commit hooks and committing additional files.
- Rewritten search for better performance and user experience.
- New interface for machine translations.
- Added support for monolingual po files.
- Extend amount of cached metadata to improve speed of various searches.
- Now shows word counts as well.

4.31.6 Weblate 1.4

Released on January 23rd 2013.

- Fixed deleting of checks/comments on string deletion.
- Added option to disable automatic propagation of translations.
- Added option to subscribe for merge failures.
- Correctly import on projects which needs custom ttkit loader.
- Added sitemaps to allow easier access by crawlers.
- Provide direct links to string in notification e-mails or feeds.
- Various improvements to admin interface.
- Provide hints for production setup in admin interface.
- Added per language widgets and engage page.
- Improved translation locking handling.
- Show code snippets for widgets in more variants.
- Indicate failing checks or fuzzy strings in progress bars.
- More options for formatting commit message.
- Fixed error handling with machine translation services.
- Improved automatic translation locking behaviour.

- Support for showing changes from previous source string.
- Added support for substring search.
- Various quality checks improvements.
- Support for per project ACL.
- Basic code coverage by unit tests.

4.31.7 Weblate 1.3

Released on November 16th 2012.

- Compatibility with PostgreSQL database backend.
- Removes languages removed in upstream git repository.
- Improved quality checks processing.
- Added new checks (BB code, XML markup and newlines).
- Support for optional rebasing instead of merge.
- Possibility to relocate Weblate (for example to run it under /weblate path).
- Support for manually choosing file type in case autodetection fails.
- Better support for Android resources.
- Support for generating SSH key from web interface.
- More visible data exports.
- New buttons to enter some special characters.
- Support for exporting dictionary.
- Support for locking down whole Weblate installation.
- Checks for source strings and support for source strings review.
- Support for user comments for both translations and source strings.
- Better changes log tracking.
- Changes can now be monitored using RSS.
- Improved support for RTL languages.

4.31.8 Weblate 1.2

Released on August 14th 2012.

- Weblate now uses South for database migration, please check upgrade instructions if you are upgrading.
- Fixed minor issues with linked git repos.
- New introduction page for engaging people with translating using Weblate.
- Added widgets which can be used for promoting translation projects.
- Added option to reset repository to origin (for privileged users).
- Project or component can now be locked for translations.
- Possibility to disable some translations.
- Configurable options for adding new translations.
- Configuration of git commits per project.

- Simple antispam protection.
- Better layout of main page.
- Support for automatically pushing changes on every commit.
- Support for e-mail notifications of translators.
- List only used languages in preferences.
- Improved handling of not known languages when importing project.
- Support for locking translation by translator.
- Optionally maintain `Language-Team` header in po file.
- Include some statistics in about page.
- Supports (and requires) django-registration 0.8.
- Caching counts of strings with failing checks.
- Checking of requirements during setup.
- Documentation improvements.

4.31.9 Weblate 1.1

Released on July 4th 2012.

- Improved several translations.
- Better validation while creating component.
- Added support for shared git repositories across components.
- Do not necessary commit on every attempt to pull remote repo.
- Added support for offloading indexing.

4.31.10 Weblate 1.0

Released on May 10th 2012.

- Improved validation while adding/saving component.
- Experimental support for Android component files (needs patched ttkit).
- Updates from hooks are run in background.
- Improved installation instructions.
- Improved navigation in dictionary.

4.32 Weblate 0.x series

4.32.1 Weblate 0.9

Released on April 18th 2012.

- Fixed import of unknown languages.
- Improved listing of nearby messages.
- Improved several checks.
- Documentation updates.

- Added definition for several more languages.
- Various code cleanups.
- Documentation improvements.
- Changed file layout.
- Update helper scripts to Django 1.4.
- Improved navigation while translating.
- Better handling of po file renames.
- Better validation while creating component.
- Integrated full setup into syncdb.
- Added list of recent changes to all translation pages.
- Check for not translated strings ignores format string only messages.

4.32.2 Weblate 0.8

Released on April 3rd 2012.

- Replaced own full text search with Whoosh.
- Various fixes and improvements to checks.
- New command updatechecks.
- Lot of translation updates.
- Added dictionary for storing most frequently used terms.
- Added /admin/report/ for overview of repositories status.
- Machine translation services no longer block page loading.
- Management interface now contains also useful actions to update data.
- Records log of changes made by users.
- Ability to postpone commit to Git to generate less commits from single user.
- Possibility to browse failing checks.
- Automatic translation using already translated strings.
- New about page showing used versions.
- Django 1.4 compatibility.
- Ability to push changes to remote repo from web interface.
- Added review of translations done by others.

4.32.3 Weblate 0.7

Released on February 16th 2012.

- Direct support for GitHub notifications.
- Added support for cleaning up orphaned checks and translations.
- Displays nearby strings while translating.
- Displays similar strings while translating.
- Improved searching for string.

4.32.4 Weblate 0.6

Released on February 14th 2012.

- Added various checks for translated messages.
- Tunable access control.
- Improved handling of translations with new lines.
- Added client side sorting of tables.
- Please check upgrading instructions in case you are upgrading.

4.32.5 Weblate 0.5

Released on February 12th 2012.

- **Support for machine translation using following online services:**
  - Apertium
  - Microsoft Translator
  - MyMemory
- Several new translations.
- Improved merging of upstream changes.
- Better handle concurrent git pull and translation.
- Propagating works for fuzzy changes as well.
- Propagating works also for file upload.
- Fixed file downloads while using FastCGI (and possibly others).

4.32.6 Weblate 0.4

Released on February 8th 2012.

- Added usage guide to documentation.
- Fixed API hooks not to require CSRF protection.

4.32.7 Weblate 0.3

Released on February 8th 2012.

- Better display of source for plural translations.
- New documentation in Sphinx format.
- Displays secondary languages while translating.
- Improved error page to give list of existing projects.
- New per language stats.

4.32.8 Weblate 0.2

Released on February 7th 2012.

- Improved validation of several forms.
- Warn users on profile upgrade.
- Remember URL for login.
- Naming of text areas while entering plural forms.
- Automatic expanding of translation area.

4.32.9 Weblate 0.1

Released on February 6th 2012.

- Initial release.

Indexul de Module Python

W

`wlc`, 138
`wlc.config`, 139
`wlc.main`, 139

HTTP Routing Table

/

ANY /, 90

/api

GET /api/, 93

/api/addons

GET /api/addons/, 127
GET /api/addons/(int:id)/, 127
PUT /api/addons/(int:id)/, 128
DELETE /api/addons/(int:id)/, 128
PATCH /api/addons/(int:id)/, 127

/api/changes

GET /api/changes/, 124
GET /api/changes/(int:id)/, 124

/api/component-lists

GET /api/component-lists/, 128
GET /api/component-lists/(str:slug)/, 128
POST /api/component-lists/(str:slug)/components/, 129
PUT /api/component-lists/(str:slug)/, 128
DELETE /api/component-lists/(str:slug)/, 129
DELETE /api/component-lists/(str:slug)/components/(str:component/slug), 129
PATCH /api/component-lists/(str:slug)/, 128

/api/components

GET /api/components/, 108
GET /api/components/(string:project)/(string:compo, 108
GET /api/components/(string:project)/(string:compo, 112
GET /api/components/(string:project)/(string:compo, 116
GET /api/components/(string:project)/(string:compo, 112
GET /api/components/(string:project)/(string:compo, 114
GET /api/components/(string:project)/(string:compo, 114
GET /api/components/(string:project)/(string:compo, 113
GET /api/components/(string:project)/(string:compo, 112
GET /api/components/(string:project)/(string:compo, 116
GET /api/components/(string:project)/(string:compo, 114
POST /api/components/(string:project)/(string:comp, 127
POST /api/components/(string:project)/(string:comp, 116
POST /api/components/(string:project)/(string:comp, 112
POST /api/components/(string:project)/(string:comp, 113
POST /api/components/(string:project)/(string:comp, 114
PUT /api/components/(string:project)/(string:compo, 111
DELETE /api/components/(string:project)/(string:c, 111
DELETE /api/components/(string:project)/(string:c, 116
PATCH /api/components/(string:project)/(string:com, 110

/api/groups

GET /api/groups/, 96
GET /api/groups/(int:id)/, 96
POST /api/groups/, 96
POST /api/groups/(int:id)/componentlists/, 98
POST /api/groups/(int:id)/components/, 98
POST /api/groups/(int:id)/languages/, 98

POST /api/groups/(int:id)/projects/, 98
POST /api/groups/(int:id)/roles/, 98
PUT /api/groups/(int:id)/, 97
DELETE /api/groups/(int:id)/, 97
DELETE /api/groups/(int:id)/componentlists/(int:component_list_id), 99
DELETE /api/groups/(int:id)/components/(int:component_id), 98
DELETE /api/groups/(int:id)/languages/(string:language_code), 98
DELETE /api/groups/(int:id)/projects/(int:project_id), 98
PATCH /api/groups/(int:id)/, 97

/api/languages

GET /api/languages/, 100
GET /api/languages/(string:language)/, 100
GET /api/languages/(string:language)/statistics/, 101
POST /api/languages/, 100
PUT /api/languages/(string:language)/, 101
DELETE /api/languages/(string:language)/, 101
PATCH /api/languages/(string:language)/, 101

/api/metrics

GET /api/metrics/, 130

/api/projects

GET /api/projects/, 102
GET /api/projects/(string:project)/, 102
GET /api/projects/(string:project)/changes/, 103
GET /api/projects/(string:project)/components/, 104
GET /api/projects/(string:project)/languages/, 107
GET /api/projects/(string:project)/repository/, 103
GET /api/projects/(string:project)/statistics/, 107
POST /api/projects/, 102
POST /api/projects/(string:project)/components/, 105
POST /api/projects/(string:project)/repository/, 104
PUT /api/projects/(string:project)/, 103
DELETE /api/projects/(string:project)/, 103
PATCH /api/projects/(string:project)/, 103

/api/roles

GET /api/roles/, 99
GET /api/roles/(int:id)/, 99
POST /api/roles/, 99
PUT /api/roles/(int:id)/, 99
DELETE /api/roles/(int:id)/, 100
PATCH /api/roles/(int:id)/, 100

/api/screenshots

GET /api/screenshots/, 125
GET /api/screenshots/(int:id)/, 125
GET /api/screenshots/(int:id)/file/, 125
POST /api/screenshots/, 126
POST /api/screenshots/(int:id)/file/, 125
POST /api/screenshots/(int:id)/units/, 125
PUT /api/screenshots/(int:id)/, 126
DELETE /api/screenshots/(int:id)/, 127
DELETE /api/screenshots/(int:id)/units/(int:unit_ 126
PATCH /api/screenshots/(int:id)/, 126

/api/tasks

GET /api/tasks/, 129
GET /api/tasks/(str:uuid)/, 129

/api/translations

GET /api/translations/, 117
GET /api/translations/(string:project)/(string:co 117
GET /api/translations/(string:project)/(string:co 119
GET /api/translations/(string:project)/(string:co 120
GET /api/translations/(string:project)/(string:co 121
GET /api/translations/(string:project)/(string:co 121
GET /api/translations/(string:project)/(string:co 119
POST /api/translations/(string:project)/(string:c 120
POST /api/translations/(string:project)/(string:c 120
POST /api/translations/(string:project)/(string:c 121
POST /api/translations/(string:project)/(string:c 119
DELETE /api/translations/(string:project)/(string 119

/api/units

GET /api/units/, 122
GET /api/units/(int:id)/, 122
PUT /api/units/(int:id)/, 123

DELETE /api/units/(int:id)/, 124
PATCH /api/units/(int:id)/, 123

/api/users

GET /api/users/, 93
GET /api/users/(str:username)/, 93
GET /api/users/(str:username)/notifications/, 95
GET /api/users/(str:username)/notifications/(int:subscription_id)/, 95
GET /api/users/(str:username)/statistics/, 95
POST /api/users/, 93
POST /api/users/(str:username)/groups/, 95
POST /api/users/(str:username)/notifications/, 95
PUT /api/users/(str:username)/, 94
PUT /api/users/(str:username)/notifications/(int:subscription_id)/, 95
DELETE /api/users/(str:username)/, 95
DELETE /api/users/(str:username)/notifications/(int:subscription_id)/, 96
PATCH /api/users/(str:username)/, 94
PATCH /api/users/(str:username)/notifications/(int:subscription_id)/, 96

/exports

GET /exports/rss/, 133
GET /exports/rss/(string:project)/, 133
GET /exports/rss/(string:project)/(string:component)/, 133
GET /exports/rss/(string:project)/(string:component)/(string:language)/, 133
GET /exports/rss/language/(string:language)/, 133
GET /exports/stats/(string:project)/(string:component)/, 132

/hooks

GET /hooks/update/(string:project)/, 130
GET /hooks/update/(string:project)/(string:component)/, 130
POST /hooks/azure/, 131
POST /hooks/bitbucket/, 131
POST /hooks/gitea/, 131
POST /hooks/gitee/, 131
POST /hooks/github/, 130
POST /hooks/gitlab/, 131
POST /hooks/pagure/, 131

Index

Simboluri

.XML resource file
    file format, 76
--add
    auto_translate command line option, 350
--addon ADDON
    install_addon command line option, 356
--age HOURS
    commit_pending command line option, 351
--author USER@EXAMPLE.COM
    add_suggestions command line option, 349
--base-file-template TEMPLATE
    import_project command line option, 354
--check
    importusers command line option, 355
--config PATH
    wlc command line option, 135
--config-section SECTION
    wlc command line option, 135
--configuration CONFIG
    install_addon command line option, 356
--convert
    wlc command line option, 136
--email USER@EXAMPLE.COM
    createadmin command line option, 351
--file-format FORMAT
    import_project command line option, 354
--force
    loadpo command line option, 357
--force-commit
    pushgit command line option, 358
--format {csv,json,text,html}
    wlc command line option, 135
--ignore
    import_json command line option, 352
--inconsistent
    auto_translate command line option, 349
--input
    wlc command line option, 136
--key KEY
    wlc command line option, 135
--lang LANGUAGE
    loadpo command line option, 357
--language-code
    list_translators command line option, 356
--language-map LANGMAP
    import_memory command line option, 353
--language-regex REGEX
    import_project command line option, 354
--license NAME
    import_project command line option, 354
--license-url URL
    import_project command line option, 354
--main-component
    import_project command line option, 354
--main-component COMPONENT
    import_json command line option, 352
--mode MODE
    auto_translate command line option, 350
--mt MT
    auto_translate command line option, 350
--name
    createadmin command line option, 351
--name-template TEMPLATE
    import_project command line option, 354
--new-base-template TEMPLATE
    import_project command line option, 354
--no-password
    createadmin command line option, 351

--no-privs-update
setupgroups command line option, 358
--no-projects-update
setupgroups command line option, 358
--no-update
setuplang command line option, 358
--output
wlc command line option, 136
--overwrite
auto_translate command line option, 349
wlc command line option, 136
--password PASSWORD
createadmin command line option, 351
--project PROJECT
import_json command line option, 352
--source PROJECT/COMPONENT
auto_translate command line option, 349
--threshold THRESHOLD
auto_translate command line option, 350
--update
createadmin command line option, 351
import_json command line option, 352
install_addon command line option, 356
--url URL
wlc command line option, 135
--user USERNAME
auto_translate command line option, 349
--username USERNAME
createadmin command line option, 351
--vcs NAME
import_project command line option, 354

A

add_suggestions
weblate admin command, 349
add_suggestions command line option
--author USER@EXAMPLE.COM, 349
ADMINS
setting, 183
AKISMET_API_KEY
setting, 303
ALLOWED_HOSTS
setting, 183
Android
file format, 71
ANONYMOUS_USER_NAME
setting, 304
API, 90, 133, 138
Apple strings
file format, 72
ARB
file format, 75
AUDITLOG_EXPIRY
setting, 304
AUTH_LOCK_ATTEMPTS
setting, 304
AUTH_TOKEN_VALID
setting, 305
auto_translate
weblate admin command, 349
auto_translate command line option
--add, 350
--inconsistent, 349
--mode MODE, 350
--mt MT, 350
--overwrite, 349
--source PROJECT/COMPONENT, 349
--threshold THRESHOLD, 350
--user USERNAME, 349
AUTO_UPDATE
setting, 304
AUTOFIX_LIST
setting, 305
AVATAR_URL_PREFIX
setting, 305

B

BACKGROUND_TASKS
setting, 306
BASE_DIR
setting, 306
BaseAddon (*clasa în weblate.addons.base*), 393
BASIC_LANGUAGES
setting, 306
bilingual
translation, 63

C

can_install() (*metoda clasei weblate.addons.base.BaseAddon*), 393
CELERY_BACKUP_OPTIONS, 145, 159
CELERY_BEAT_OPTIONS, 145, 159
CELERY_MAIN_OPTIONS, 145, 159
CELERY_MEMORY_OPTIONS, 145, 159
CELERY_NOTIFY_OPTIONS, 145, 159
celery_queues
weblate admin command, 350
CELERY_TRANSLATE_OPTIONS, 145, 159
changes
wlc command line option, 136
CHECK_LIST
setting, 307
checkgit
weblate admin command, 350
cleanup
wlc command line option, 136
cleanuptrans
weblate admin command, 351
Comma separated values
file format, 76

Command (*clasa în wlc.main*), 140
COMMENT_CLEANUP_DAYS
    setting, 307
commit
    wlc command line option, 135
commit_pending
    weblate admin command, 351
commit_pending command line option
    --age HOURS, 351
COMMIT_PENDING_HOURS
    setting, 308
commitgit
    weblate admin command, 350
configure() (*metoda weblate.addons.base.BaseAddon*), 393
CONTACT_FORM
    setting, 308
createadmin
    weblate admin command, 351
createadmin command line option
    --email USER@EXAMPLE.COM, 351
    --name, 351
    --no-password, 351
    --password PASSWORD, 351
    --update, 351
    --username USERNAME, 351
CSP_CONNECT_SRC
    setting, 307
CSP_FONT_SRC
    setting, 307
CSP_IMG_SRC
    setting, 307
CSP_SCRIPT_SRC
    setting, 307
CSP_STYLE_SRC
    setting, 307
CSV
    file format, 76

D

daily() (*metoda weblate.addons.base.BaseAddon*), 393
DATA_DIR
    setting, 308
DATABASE_BACKUP
    setting, 309
DATABASES
    setting, 183
DEBUG
    setting, 183
DEFAULT_ACCESS_CONTROL
    setting, 309
DEFAULT_ADD_MESSAGE
    setting, 310
DEFAULT_ADDON_MESSAGE
    setting, 310
DEFAULT_ADDONS
    setting, 310
DEFAULT_AUTO_WATCH
    setting, 309
DEFAULT_COMMIT_MESSAGE
    setting, 310
DEFAULT_COMMITER_EMAIL
    setting, 310
DEFAULT_COMMITER_NAME
    setting, 310
DEFAULT_DELETE_MESSAGE
    setting, 310
DEFAULT_FROM_EMAIL
    setting, 183
DEFAULT_LANGUAGE
    setting, 311
DEFAULT_MERGE_MESSAGE
    setting, 310
DEFAULT_MERGE_STYLE
    setting, 311
DEFAULT_PAGE_LIMIT
    setting, 325
DEFAULT_PULL_MESSAGE
    setting, 311
DEFAULT_RESTRICTED_COMPONENT
    setting, 309
DEFAULT_SHARED_TM
    setting, 311
DEFAULT_TRANSLATION_PROPAGATION
    setting, 311
download
    wlc command line option, 136
DTD
    file format, 78
dump_memory
    weblate admin command, 352
dumpuserdata
    weblate admin command, 352

E

ENABLE_AVATARS
    setting, 312
ENABLE_HOOKS
    setting, 312
ENABLE_HTTPS
    setting, 312
ENABLE_SHARING
    setting, 312

F

file format
    .XML resource file, 76
    Android, 71
    Apple strings, 72
    ARB, 75
    Comma separated values, 76
    CSV, 76
    DTD, 78
    gettext, 65
    go-i18n, 74

GWT properties, 69
i18next, 74
INI translations, 69, 70
Java properties, 68
Joomla translations, 70
JSON, 73
mi18n lang, 69
PHP strings, 72
PO, 65
Qt, 70
RC, 79
RESX, 76
Ruby YAML, 77
Ruby YAML Ain't Markup Language, 77
string resources, 71
TS, 70
XLIFF, 66
XML, 78
YAML, 77
YAML Ain't Markup Language, 77

G

get() (*metoda wlc.Weblate*), 138
get_add_form() (*metoda clasei weblate.addons.base.BaseAddon*), 393
GET_HELP_URL
setting, 312
get_settings_form() (*metoda weblate.addons.base.BaseAddon*), 393
gettext
file format, 65
GITHUB_CREDENTIALS
setting, 313
GITHUB_TOKEN
setting, 314
GITHUB_USERNAME
setting, 313
GITLAB_CREDENTIALS
setting, 312
GITLAB_TOKEN
setting, 313
GITLAB_USERNAME
setting, 313
go-i18n
file format, 74
GOOGLE_ANALYTICS_ID
setting, 314
GWT properties
file format, 69

H

HIDE_REPO_CREDENTIALS
setting, 314
HIDE_VERSION
setting, 314

I

i18next
file format, 74
import_demo
weblate admin command, 352
import_json
weblate admin command, 352
import_json command line option
--ignore, 352
--main-component COMPONENT, 352
--project PROJECT, 352
--update, 352
import_memory
weblate admin command, 353
import_memory command line option
--language-map LANGMAP, 353
import_project
weblate admin command, 353
import_project command line option
--base-file-template TEMPLATE, 354
--file-format FORMAT, 354
--language-regex REGEX, 354
--license NAME, 354
--license-url URL, 354
--main-component, 354
--name-template TEMPLATE, 354
--new-base-template TEMPLATE, 354
--vcs NAME, 354
importuserdata
weblate admin command, 355
importusers
weblate admin command, 355
importusers command line option
--check, 355
INI translations
file format, 69, 70
install_addon
weblate admin command, 356
install_addon command line option
--addon ADDON, 356
--configuration CONFIG, 356
--update, 356
IP_BEHIND_REVERSE_PROXY
setting, 314
IP_PROXY_HEADER
setting, 315
IP_PROXY_OFFSET
setting, 315
iPad
translation, 72
iPhone
translation, 72

J

Java properties
file format, 68
Joomla translations
file format, 70
JSON
file format, 73

L

LEGAL_URL
    setting, 315
LICENSE_EXTRA
    setting, 316
LICENSE_FILTER
    setting, 316
LICENSE_REQUIRED
    setting, 316
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH
    setting, 317
list_languages
    weblate admin command, 356
list_translators
    weblate admin command, 356
list_translators command line option
    --language-code, 356
list_versions
    weblate admin command, 357
list-components
    wlc command line option, 135
list-languages
    wlc command line option, 135
list-projects
    wlc command line option, 135
list-translations
    wlc command line option, 135
load() (*metoda wlc.config.WeblateConfig*), 139
loadpo
    weblate admin command, 357
loadpo command line option
    --force, 357
    --lang LANGUAGE, 357
LOCALIZE_CDN_PATH
    setting, 317
LOCALIZE_CDN_URL
    setting, 317
lock
    wlc command line option, 136
lock_translation
    weblate admin command, 357
lock-status
    wlc command line option, 136
LOGIN_REQUIRED_URLS
    setting, 317
LOGIN_REQUIRED_URLS_EXCEPTIONS
    setting, 317
ls
    wlc command line option, 135

M

MACHINE_TRANSLATION_SERVICES
    setting, 318
main() (*în modulul wlc.main*), 139
MATOMO_SITE_ID
    setting, 318
MATOMO_URL
    setting, 318
mi18n lang
    file format, 69
modul
    wlc, 138
    wlc.config, 139
    wlc.main, 139
monolingual
    translation, 63
move_language
    weblate admin command, 357
MT_APERTIUM_APY
    setting, 319
MT_AWS_ACCESS_KEY_ID
    setting, 319
MT_AWS_REGION
    setting, 319
MT_AWS_SECRET_ACCESS_KEY
    setting, 319
MT_BAIDU_ID
    setting, 320
MT_BAIDU_SECRET
    setting, 320
MT_DEEPL_API_URL
    setting, 320
MT_DEEPL_KEY
    setting, 320
MT_GOOGLE_CREDENTIALS
    setting, 321
MT_GOOGLE_KEY
    setting, 321
MT_GOOGLE_LOCATION
    setting, 321
MT_GOOGLE_PROJECT
    setting, 321
MT_LIBRETRANSLATE_API_URL
    setting, 321
MT_LIBRETRANSLATE_KEY
    setting, 321
MT_MICROSOFT_BASE_URL
    setting, 322
MT_MICROSOFT_COGNITIVE_KEY
    setting, 322
MT_MICROSOFT_ENDPOINT_URL
    setting, 322
MT_MICROSOFT_REGION
    setting, 322
MT_MODERNMT_KEY
    setting, 322
MT_MODERNMT_URL
    setting, 322
MT_MYMEMORY_EMAIL
    setting, 323
MT_MYMEMORY_KEY
    setting, 323
MT_MYMEMORY_USER
    setting, 323
MT_NETEASE_KEY
    setting, 323

MT_NETEASE_SECRET
setting, 323
MT_SAP_BASE_URL
setting, 324
MT_SAP_PASSWORD
setting, 324
MT_SAP_SANDBOX_APIKEY
setting, 324
MT_SAP_USE_MT
setting, 324
MT_SAP_USERNAME
setting, 324
MT_SERVICES
setting, 318
MT_TMSERVER
setting, 323
MT_YANDEX_KEY
setting, 323
MT_YOUDAO_ID
setting, 324
MT_YOUDAO_SECRET
setting, 324

N

NEARBY_MESSAGES
setting, 325

P

PAGURE_CREDENTIALS
setting, 325
PAGURE_TOKEN
setting, 325
PAGURE_USERNAME
setting, 325
PHP strings
file format, 72
PIWIK_SITE_ID
setting, 318
PIWIK_URL
setting, 318
PO
file format, 65
post() (*metoda wlc.Weblate*), 139
post_add() (*metoda weblate.addons.base.BaseAddon*), 393
post_commit() (*metoda weblate.addons.base.BaseAddon*), 393
post_push() (*metoda weblate.addons.base.BaseAddon*), 393
post_update() (*metoda weblate.addons.base.BaseAddon*), 393
pre_commit() (*metoda weblate.addons.base.BaseAddon*), 393
pre_push() (*metoda weblate.addons.base.BaseAddon*), 393
pre_update() (*metoda weblate.addons.base.BaseAddon*), 393
PRIVACY_URL
setting, 326
pull
wlc command line option, 135
push
wlc command line option, 136
pushgit
weblate admin command, 358
pushgit command line option
--force-commit, 358
Python, 138

Q

Qt
file format, 70

R

RATELIMIT_ATTEMPTS
setting, 326
RATELIMIT_LOCKOUT
setting, 326
RATELIMIT_WINDOW
setting, 326
RC
file format, 79
register_command() (*în modulul wlc.main*), 139
REGISTRATION_ALLOW_BACKENDS
setting, 327
REGISTRATION_CAPTCHA
setting, 327
REGISTRATION_EMAIL_MATCH
setting, 327
REGISTRATION_OPEN
setting, 327
repo
wlc command line option, 136
REPOSITORY_ALERT_THRESHOLD
setting, 328
REQUIRE_LOGIN
setting, 328
reset
wlc command line option, 136
REST, 90
RESX
file format, 76
RFC
RFC 5646, 62
Ruby YAML
file format, 77
Ruby YAML Ain't Markup Language
file format, 77

S

save_state() (*metoda weblate.addons.base.BaseAddon*), 393
SECRET_KEY
setting, 183
SENTRY_DSN
setting, 328

SERVER_EMAIL
    setting, 184
SESSION_COOKIE_AGE_AUTHENTICATED
    setting, 328
SESSION_ENGINE
    setting, 183
setting
    ADMINS, 183
    AKISMET_API_KEY, 303
    ALLOWED_HOSTS, 183
    ANONYMOUS_USER_NAME, 304
    AUDITLOG_EXPIRY, 304
    AUTH_LOCK_ATTEMPTS, 304
    AUTH_TOKEN_VALID, 305
    AUTO_UPDATE, 304
    AUTOFIX_LIST, 305
    AVATAR_URL_PREFIX, 305
    BACKGROUND_TASKS, 306
    BASE_DIR, 306
    BASIC_LANGUAGES, 306
    CHECK_LIST, 307
    COMMENT_CLEANUP_DAYS, 307
    COMMIT_PENDING_HOURS, 308
    CONTACT_FORM, 308
    CSP_CONNECT_SRC, 307
    CSP_FONT_SRC, 307
    CSP_IMG_SRC, 307
    CSP_SCRIPT_SRC, 307
    CSP_STYLE_SRC, 307
    DATA_DIR, 308
    DATABASE_BACKUP, 309
    DATABASES, 183
    DEBUG, 183
    DEFAULT_ACCESS_CONTROL, 309
    DEFAULT_ADD_MESSAGE, 310
    DEFAULT_ADDON_MESSAGE, 310
    DEFAULT_ADDONS, 310
    DEFAULT_AUTO_WATCH, 309
    DEFAULT_COMMIT_MESSAGE, 310
    DEFAULT_COMMITER_EMAIL, 310
    DEFAULT_COMMITER_NAME, 310
    DEFAULT_DELETE_MESSAGE, 310
    DEFAULT_FROM_EMAIL, 183
    DEFAULT_LANGUAGE, 311
    DEFAULT_MERGE_MESSAGE, 310
    DEFAULT_MERGE_STYLE, 311
    DEFAULT_PAGE_LIMIT, 325
    DEFAULT_PULL_MESSAGE, 311
    DEFAULT_RESTRICTED_COMPONENT, 309
    DEFAULT_SHARED_TM, 311
    DEFAULT_TRANSLATION_PROPAGATION, 311
    ENABLE_AVATARS, 312
    ENABLE_HOOKS, 312
    ENABLE_HTTPS, 312
    ENABLE_SHARING, 312
    GET_HELP_URL, 312
    GITHUB_CREDENTIALS, 313
    GITHUB_TOKEN, 314
    GITHUB_USERNAME, 313
    GITLAB_CREDENTIALS, 312
    GITLAB_TOKEN, 313
    GITLAB_USERNAME, 313
    GOOGLE_ANALYTICS_ID, 314
    HIDE_REPO_CREDENTIALS, 314
    HIDE_VERSION, 314
    IP_BEHIND_REVERSE_PROXY, 314
    IP_PROXY_HEADER, 315
    IP_PROXY_OFFSET, 315
    LEGAL_URL, 315
    LICENSE_EXTRA, 316
    LICENSE_FILTER, 316
    LICENSE_REQUIRED, 316
    LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH, 317
    LOCALIZE_CDN_PATH, 317
    LOCALIZE_CDN_URL, 317
    LOGIN_REQUIRED_URLS, 317
    LOGIN_REQUIRED_URLS_EXCEPTIONS, 317
    MACHINE_TRANSLATION_SERVICES, 318
    MATOMO_SITE_ID, 318
    MATOMO_URL, 318
    MT_APERTIUM_APY, 319
    MT_AWS_ACCESS_KEY_ID, 319
    MT_AWS_REGION, 319
    MT_AWS_SECRET_ACCESS_KEY, 319
    MT_BAIDU_ID, 320
    MT_BAIDU_SECRET, 320
    MT_DEEPL_API_URL, 320
    MT_DEEPL_KEY, 320
    MT_GOOGLE_CREDENTIALS, 321
    MT_GOOGLE_KEY, 321
    MT_GOOGLE_LOCATION, 321
    MT_GOOGLE_PROJECT, 321
    MT_LIBRETRANSLATE_API_URL, 321
    MT_LIBRETRANSLATE_KEY, 321
    MT_MICROSOFT_BASE_URL, 322
    MT_MICROSOFT_COGNITIVE_KEY, 322
    MT_MICROSOFT_ENDPOINT_URL, 322
    MT_MICROSOFT_REGION, 322
    MT_MODERNMT_KEY, 322
    MT_MODERNMT_URL, 322
    MT_MYMEMORY_EMAIL, 323
    MT_MYMEMORY_KEY, 323
    MT_MYMEMORY_USER, 323
    MT_NETEASE_KEY, 323
    MT_NETEASE_SECRET, 323
    MT_SAP_BASE_URL, 324
    MT_SAP_PASSWORD, 324
    MT_SAP_SANDBOX_APIKEY, 324
    MT_SAP_USE_MT, 324
    MT_SAP_USERNAME, 324
    MT_SERVICES, 318
    MT_TMSERVER, 323
    MT_YANDEX_KEY, 323
    MT_YOUDAO_ID, 324

MT_YOUDAO_SECRET, 324
NEARBY_MESSAGES, 325
PAGURE_CREDENTIALS, 325
PAGURE_TOKEN, 325
PAGURE_USERNAME, 325
PIWIK_SITE_ID, 318
PIWIK_URL, 318
PRIVACY_URL, 326
RATELIMIT_ATTEMPTS, 326
RATELIMIT_LOCKOUT, 326
RATELIMIT_WINDOW, 326
REGISTRATION_ALLOW_BACKENDS, 327
REGISTRATION_CAPTCHA, 327
REGISTRATION_EMAIL_MATCH, 327
REGISTRATION_OPEN, 327
REPOSITORY_ALERT_THRESHOLD, 328
REQUIRE_LOGIN, 328
SECRET_KEY, 183
SENTRY_DSN, 328
SERVER_EMAIL, 184
SESSION_COOKIE_AGE_AUTHENTICATED, 328
SESSION_ENGINE, 183
SIMPLIFY_LANGUAGES, 328
SINGLE_PROJECT, 329
SITE_DOMAIN, 329
SITE_TITLE, 329
SPECIAL_CHARS, 329
STATUS_URL, 330
SUGGESTION_CLEANUP_DAYS, 330
UPDATE_LANGUAGES, 330
URL_PREFIX, 330
VCS_BACKENDS, 330
VCS_CLONE_DEPTH, 331
WEBLATE_ADDONS, 331
WEBLATE_EXPORTERS, 332
WEBLATE_FORMATS, 332
WEBLATE_GPG_IDENTITY, 332
WEBSITE_REQUIRED, 332
setupgroups
weblate admin command, 358
setupgroups command line option
--no-privs-update, 358
--no-projects-update, 358
setuplang
weblate admin command, 358
setuplang command line option
--no-update, 358
show
wlc command line option, 135
SIMPLIFY_LANGUAGES
setting, 328
SINGLE_PROJECT
setting, 329
SITE_DOMAIN
setting, 329
SITE_TITLE
setting, 329
SPECIAL_CHARS
setting, 329
statistics
wlc command line option, 136
STATUS_URL
setting, 330
stay_on_create (*atribut* *weblate.addons.base.BaseAddon*), 393
store_post_load() (*metoda* *weblate.addons.base.BaseAddon*), 393
string resources
file format, 71
SUGGESTION_CLEANUP_DAYS
setting, 330

T

translation
bilingual, 63
iPad, 72
iPhone, 72
monolingual, 63
TS
file format, 70

U

unit_pre_create() (*metoda* *weblate.addons.base.BaseAddon*), 393
unlock
wlc command line option, 136
unlock_translation
weblate admin command, 358
UPDATE_LANGUAGES
setting, 330
updatechecks
weblate admin command, 359
updategit
weblate admin command, 359
upload
wlc command line option, 136
URL_PREFIX
setting, 330
UWSGI_WORKERS, 145, 159

V

variabilă de mediu
CELERY_BACKUP_OPTIONS, 145, 159
CELERY_BEAT_OPTIONS, 145, 159, 160
CELERY_MAIN_OPTIONS, 145, 159
CELERY_MEMORY_OPTIONS, 145, 159
CELERY_NOTIFY_OPTIONS, 145, 159
CELERY_TRANSLATE_OPTIONS, 145, 159
POSTGRES_ALTER_ROLE, 156
POSTGRES_CONN_MAX_AGE, 156
POSTGRES_DATABASE, 155
POSTGRES_HOST, 155
POSTGRES_PASSWORD, 155
POSTGRES_PASSWORD_FILE, 155
POSTGRES_PORT, 155

POSTGRES_SSL_MODE, 155
POSTGRES_USER, 155
REDIS_DB, 156
REDIS_HOST, 156
REDIS_PASSWORD, 156
REDIS_PORT, 156
REDIS_TLS, 156
REDIS_VERIFY_SSL, 156
ROLLBAR_ENVIRONMENT, 158
ROLLBAR_KEY, 158
SENTRY_DSN, 158
SENTRY_ENVIRONMENT, 158
SOCIAL_AUTH_SLACK_SECRET, 155
UWSGI_WORKERS, 145, 159, 160
WEBLATE_ADD_ADDONS, 159
WEBLATE_ADD_APPS, 159
WEBLATE_ADD_AUTOFIX, 159
WEBLATE_ADD_CHECK, 159
WEBLATE_ADD_LOGIN_REQUIRED_URLS_EXCEPTIONS, 149
WEBLATE_ADMIN_EMAIL, 145--147, 151
WEBLATE_ADMIN_NAME, 145--147
WEBLATE_ADMIN_PASSWORD, 142, 145--147
WEBLATE_ADMIN_PASSWORD_FILE, 147
WEBLATE_AKISMET_API_KEY, 150, 366
WEBLATE_ALLOWED_HOSTS, 147, 183, 187, 188, 329
WEBLATE_AUTH_LDAP_BIND_DN, 152
WEBLATE_AUTH_LDAP_BIND_PASSWORD, 152
WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS, 152
WEBLATE_AUTH_LDAP_SERVER_URI, 152
WEBLATE_AUTH_LDAP_USER_ATTR_MAP, 152
WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE, 152
WEBLATE_AUTH_LDAP_USER_SEARCH, 152
WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER, 152
WEBLATE_AUTH_LDAP_USER_SEARCH_UNION, 152
WEBLATE_AUTH_LDAP_USER_SEARCH_UNION_DELIMITER, 152
WEBLATE_BASIC_LANGUAGES, 150
WEBLATE_CONTACT_FORM, 147
WEBLATE_CSP_CONNECT_SRC, 150
WEBLATE_CSP_FONT_SRC, 150
WEBLATE_CSP_IMG_SRC, 150
WEBLATE_CSP_SCRIPT_SRC, 150
WEBLATE_CSP_STYLE_SRC, 150
WEBLATE_DATABASE_BACKUP, 156
WEBLATE_DEBUG, 146
WEBLATE_DEFAULT_ACCESS_CONTROL, 149
WEBLATE_DEFAULT_AUTO_WATCH, 150
WEBLATE_DEFAULT_COMMITER_EMAIL, 149
WEBLATE_DEFAULT_COMMITER_NAME, 149
WEBLATE_DEFAULT_FROM_EMAIL, 147
WEBLATE_DEFAULT_RESTRICTED_COMPONENT, 149
WEBLATE_DEFAULT_SHARED_TM, 150
WEBLATE_DEFAULT_TRANSLATION_PROPAGATION, 149
WEBLATE_EMAIL_BACKEND, 158
WEBLATE_EMAIL_HOST, 157
WEBLATE_EMAIL_HOST_PASSWORD, 157
WEBLATE_EMAIL_HOST_PASSWORD_FILE, 157
WEBLATE_EMAIL_HOST_USER, 157
WEBLATE_EMAIL_PORT, 157, 158
WEBLATE_EMAIL_USE_SSL, 157, 158
WEBLATE_EMAIL_USE_TLS, 157
WEBLATE_ENABLE_AVATARS, 151
WEBLATE_ENABLE_HTTPS, 147, 215
WEBLATE_GET_HELP_URL, 158
WEBLATE_GITHUB_TOKEN, 149
WEBLATE_GITHUB_USERNAME, 149
WEBLATE_GITLAB_TOKEN, 149
WEBLATE_GITLAB_USERNAME, 149
WEBLATE_GOOGLE_ANALYTICS_ID, 149
WEBLATE_GPG_IDENTITY, 150
WEBLATE_HIDE_VERSION, 150
WEBLATE_IP_PROXY_HEADER, 148
WEBLATE_LEGAL_URL, 158
WEBLATE_LICENSE_FILTER, 150
WEBLATE_LICENSE_REQUIRED, 150
WEBLATE_LOCALIZE_CDN_PATH, 158
WEBLATE_LOCALIZE_CDN_URL, 158
WEBLATE_LOGIN_REQUIRED_URLS_EXCEPTIONS, 148
WEBLATE_LOGLEVEL, 146
WEBLATE_MT_APERTIUM_APY, 151
WEBLATE_MT_AWS_ACCESS_KEY_ID, 151
WEBLATE_MT_AWS_REGION, 151
WEBLATE_MT_AWS_SECRET_ACCESS_KEY, 151
WEBLATE_MT_DEEPL_API_URL, 151
WEBLATE_MT_DEEPL_KEY, 151
WEBLATE_MT_GLOSBE_ENABLED, 152
WEBLATE_MT_GOOGLE_KEY, 151
WEBLATE_MT_LIBRETRANSLATE_API_URL, 151
WEBLATE_MT_LIBRETRANSLATE_KEY, 151
WEBLATE_MT_MICROSOFT_BASE_URL, 151
WEBLATE_MT_MICROSOFT_COGNITIVE_KEY, 151
WEBLATE_MT_MICROSOFT_ENDPOINT_URL, 151
WEBLATE_MT_MICROSOFT_REGION, 151
WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED, 152
WEBLATE_MT_MODERNMT_KEY, 151
WEBLATE_MT_MYMEMORY_ENABLED, 151
WEBLATE_MT_SAP_BASE_URL, 152
WEBLATE_MT_SAP_PASSWORD, 152
WEBLATE_MT_SAP_SANDBOX_APIKEY, 152

WEBLATE_MT_SAP_USE_MT, 152
WEBLATE_MT_SAP_USERNAME, 152
WEBLATE_NO_EMAIL_AUTH, 155
WEBLATE_PAGURE_TOKEN, 149
WEBLATE_PAGURE_USERNAME, 149
WEBLATE_PRIVACY_URL, 158
WEBLATE_RATELIMIT_ATTEMPTS, 150, 368
WEBLATE_RATELIMIT_LOCKOUT, 150
WEBLATE_RATELIMIT_WINDOW, 150
WEBLATE_REGISTRATION_ALLOW_BACKENDS, 147
WEBLATE_REGISTRATION_OPEN, 147
WEBLATE_REMOVE_ADDONS, 159
WEBLATE_REMOVE_APPS, 159
WEBLATE_REMOVE_AUTOFIX, 159
WEBLATE_REMOVE_CHECK, 159
WEBLATE_REMOVE_LOGIN_REQUIRED_URLS_EXCEPTIONS, 149
WEBLATE_REQUIRE_LOGIN, 148, 328
WEBLATE_SAML_IDP_ENTITY_ID, 155
WEBLATE_SAML_IDP_URL, 155
WEBLATE_SAML_IDP_X509CERT, 155
WEBLATE_SECURE_PROXY_SSL_HEADER, 148
WEBLATE_SERVER_EMAIL, 147
WEBLATE_SERVICE, 145, 160
WEBLATE_SILENCED_SYSTEM_CHECKS, 150, 212
WEBLATE_SIMPLIFY_LANGUAGES, 149
WEBLATE_SITE_DOMAIN, 146, 185, 203, 329
WEBLATE_SITE_TITLE, 146
WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_KEY, 154
WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET, 154
WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY, 154
WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECRET, 154
WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENANT_NAME, 154
WEBLATE_SOCIAL_AUTH_BITBUCKET_KEY, 153
WEBLATE_SOCIAL_AUTH_BITBUCKET_SECRET, 153
WEBLATE_SOCIAL_AUTH_FACEBOOK_KEY, 154
WEBLATE_SOCIAL_AUTH_FACEBOOK_SECRET, 154
WEBLATE_SOCIAL_AUTH_FEDORA, 155
WEBLATE_SOCIAL_AUTH_GITHUB_KEY, 153
WEBLATE_SOCIAL_AUTH_GITHUB_SECRET, 153
WEBLATE_SOCIAL_AUTH_GITLAB_API_URL, 154
WEBLATE_SOCIAL_AUTH_GITLAB_KEY, 154
WEBLATE_SOCIAL_AUTH_GITLAB_SECRET, 154
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_KEY, 154
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET, 154
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_ 154
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_ 154
WEBLATE_SOCIAL_AUTH_KEYCLOAK_ACCESS_TOKEN_URL, 154
WEBLATE_SOCIAL_AUTH_KEYCLOAK_ALGORITHM, 154
WEBLATE_SOCIAL_AUTH_KEYCLOAK_AUTHORIZATION_URI 154
WEBLATE_SOCIAL_AUTH_KEYCLOAK_KEY, 154
WEBLATE_SOCIAL_AUTH_KEYCLOAK_PUBLIC_KEY, 154
WEBLATE_SOCIAL_AUTH_KEYCLOAK_SECRET, 154
WEBLATE_SOCIAL_AUTH_OPENSUSE, 155
WEBLATE_SOCIAL_AUTH_SLACK_KEY, 155
WEBLATE_SOCIAL_AUTH_UBUNTU, 155
WEBLATE_STATUS_URL, 158
WEBLATE_TIME_ZONE, 147
WEBLATE_URL_PREFIX, 150
WEBLATE_WEBSITE_REQUIRED, 150
WEBLATE_WORKERS, 145, 159, 160
WL_BRANCH, 300
WL_COMPONENT_NAME, 301
WL_COMPONENT_SLUG, 301
WL_COMPONENT_URL, 301
WL_ENGAGE_URL, 301
WL_FILE_FORMAT, 300
WL_FILEMASK, 300
WL_LANGUAGE, 301
WL_NEW_BASE, 300
WL_PATH, 300
WL_PREVIOUS_HEAD, 301
WL_PROJECT_NAME, 301
WL_PROJECT_SLUG, 301
WL_REPO, 300
WL_TEMPLATE, 300
WL_VCS, 300
VCS_BACKENDS
setting, 330
VCS_CLONE_DEPTH
setting, 331
version
wlc command line option, 135

W

Weblate (*clasa în wlc*), 138
weblate admin command
add_suggestions, 349
auto_translate, 349
celery_queues, 350
checkgit, 350

cleanuptrans, 351
commit_pending, 351
commitgit, 350
createadmin, 351
dump_memory, 352
dumpuserdata, 352
import_demo, 352
import_json, 352
import_memory, 353
import_project, 353
importuserdata, 355
importusers, 355
install_addon, 356
list_languages, 356
list_translators, 356
list_versions, 357
loadpo, 357
lock_translation, 357
move_language, 357
pushgit, 358
setupgroups, 358
setuplang, 358
unlock_translation, 358
updatechecks, 359
updategit, 359
WEBLATE_ADDONS
setting, 331
WEBLATE_ADMIN_EMAIL, 145--147, 151
WEBLATE_ADMIN_NAME, 145--147
WEBLATE_ADMIN_PASSWORD, 142, 145--147
WEBLATE_ADMIN_PASSWORD_FILE, 147
WEBLATE_AKISMET_API_KEY, 366
WEBLATE_ALLOWED_HOSTS, 183, 187, 188, 329
WEBLATE_EMAIL_HOST_PASSWORD, 157
WEBLATE_EMAIL_PORT, 157, 158
WEBLATE_EMAIL_USE_SSL, 157, 158
WEBLATE_EMAIL_USE_TLS, 157
WEBLATE_ENABLE_HTTPS, 215
WEBLATE_EXPORTERS
setting, 332
WEBLATE_FORMATS
setting, 332
WEBLATE_GPG_IDENTITY
setting, 332
WEBLATE_LOCALIZE_CDN_PATH, 158
WEBLATE_RATELIMIT_ATTEMPTS, 368
WEBLATE_REQUIRE_LOGIN, 328
WEBLATE_SECURE_PROXY_SSL_HEADER, 148
WEBLATE_SERVICE, 145
WEBLATE_SILENCED_SYSTEM_CHECKS, 212
WEBLATE_SITE_DOMAIN, 185, 203, 329
WEBLATE_WORKERS, 145, 160
WeblateConfig (*clasa în wlc.config*), 139
WeblateException, 138
WEBSITE_REQUIRED
setting, 332
wlc, 133
modul, 138
wlc command line option
--config PATH, 135
--config-section SECTION, 135
--convert, 136
--format {csv,json,text,html}, 135
--input, 136
--key KEY, 135
--output, 136
--overwrite, 136
--url URL, 135
changes, 136
cleanup, 136
commit, 135
download, 136
list-components, 135
list-languages, 135
list-projects, 135
list-translations, 135
lock, 136
lock-status, 136
ls, 135
pull, 135
push, 136
repo, 136
reset, 136
show, 135
statistics, 136
unlock, 136
upload, 136
version, 135
wlc.config
modul, 139
wlc.main
modul, 139

X

XLIFF
file format, 66
XML
file format, 78

Y

YAML
file format, 77
YAML Ain't Markup Language
file format, 77