The Weblate Manual

发布 *4.2.1*

Michal Čihař

2020 年 08 月 21 日

User docs

CHAPTER 1

User docs

1.1 Weblate basics

1.1.1 Project 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. Disable it as per *Component configuration*, still producing errors for seemingly inconsistent resulting translations.

1.2 Registration and user profile

1.2.1 注册

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.

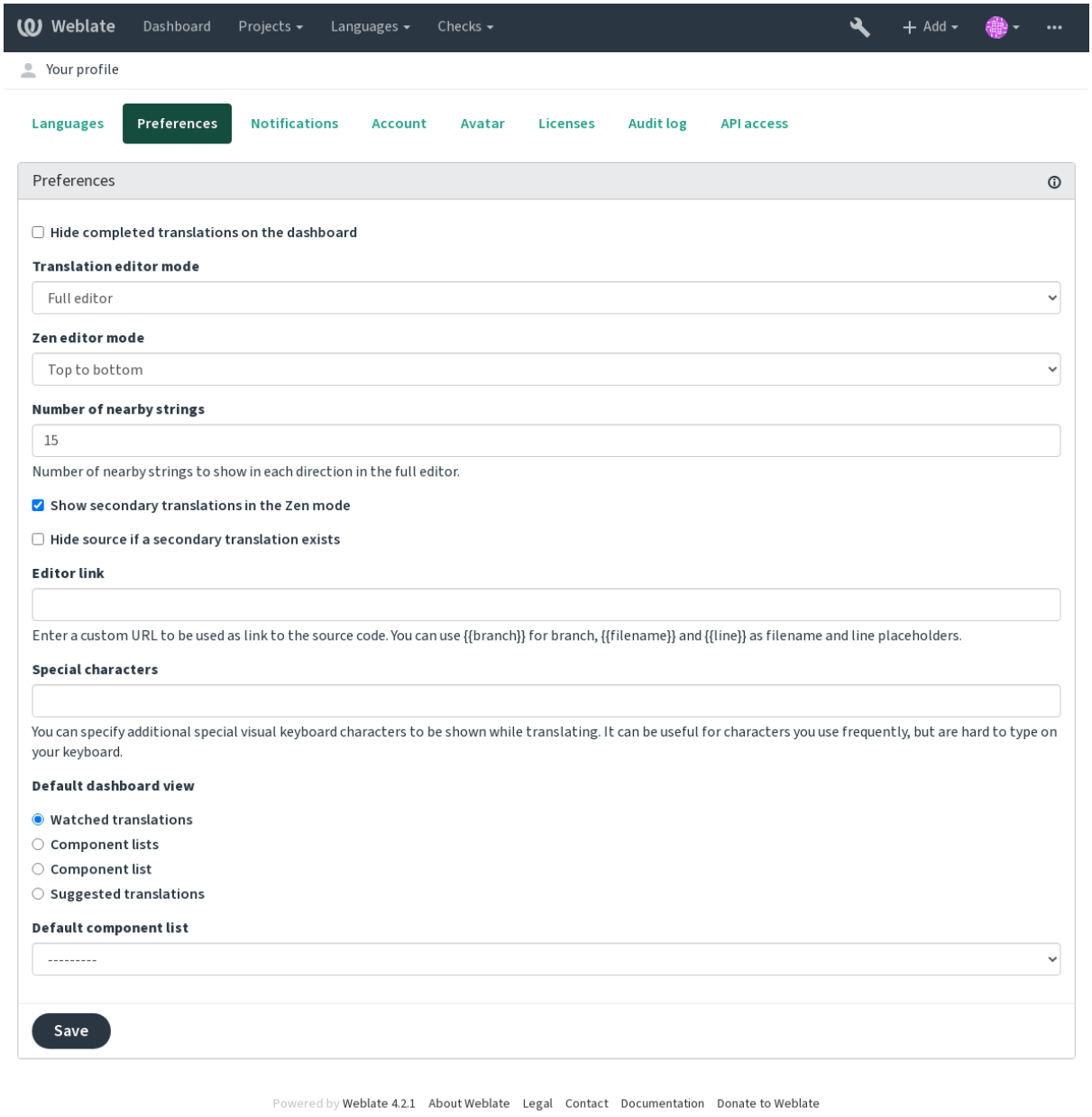
1.2.2 控制面板

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

2.5 新版功能.

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

提示： 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 组件列表.

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

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

1.2.3 用户资料

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

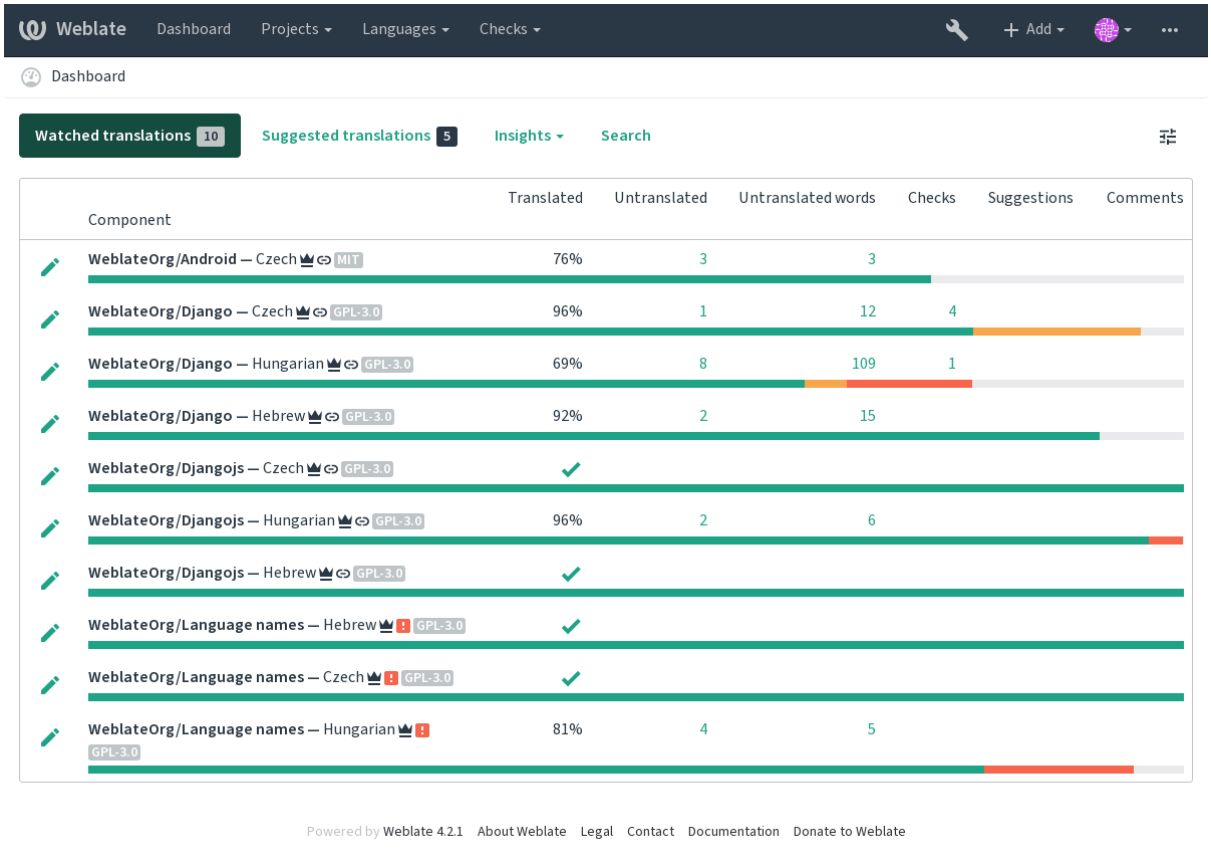
The user profile contains your preferences. Name and e-mail address is used in VCS commits, so keep this info accurate.

注解： All language selections only offer currently translated languages.

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

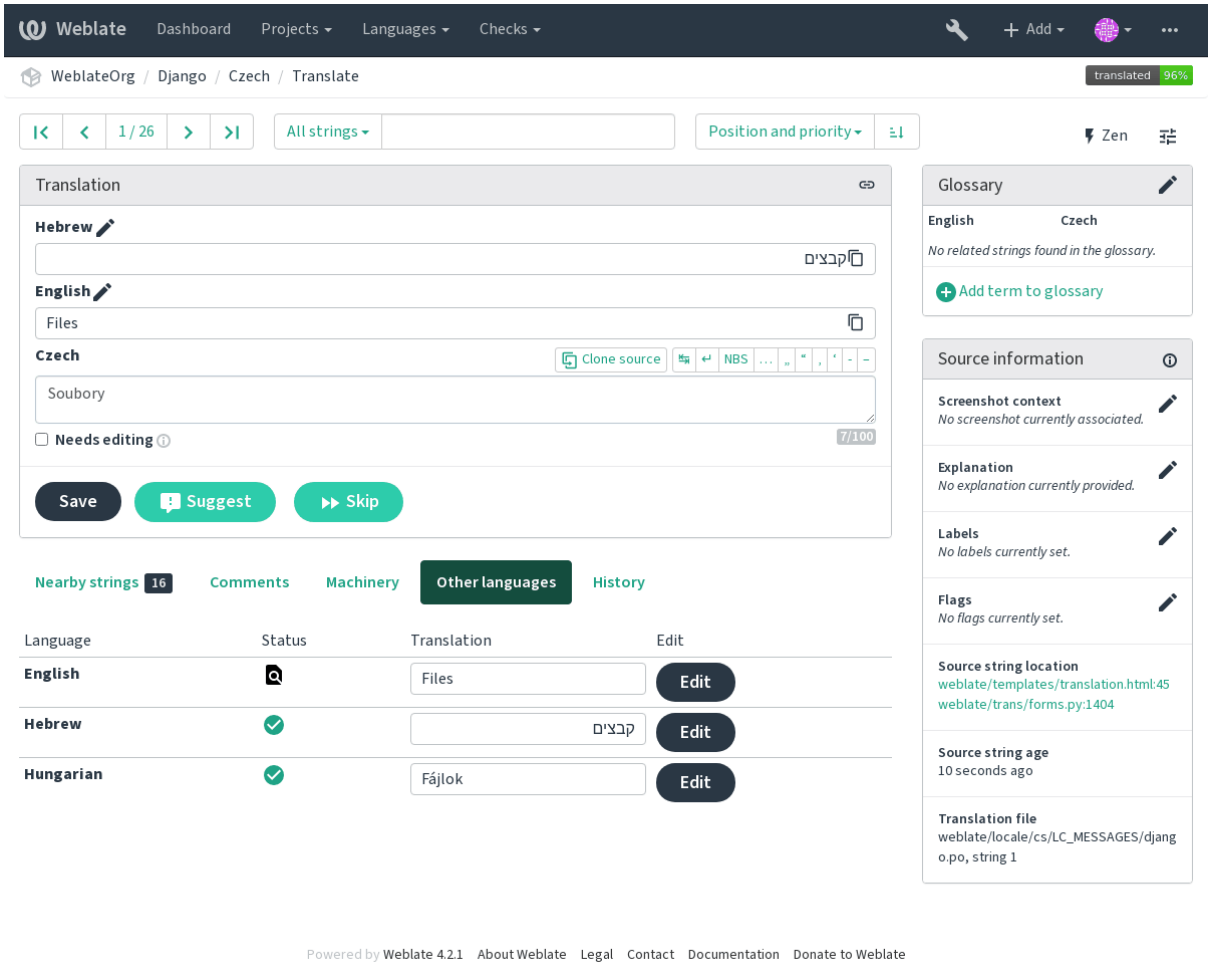
已翻译的语言

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.



第二语言

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:



控制面板默认界面

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.

参见:

组件列表

头像

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

编辑者链接

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

提示： 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.

参见：

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

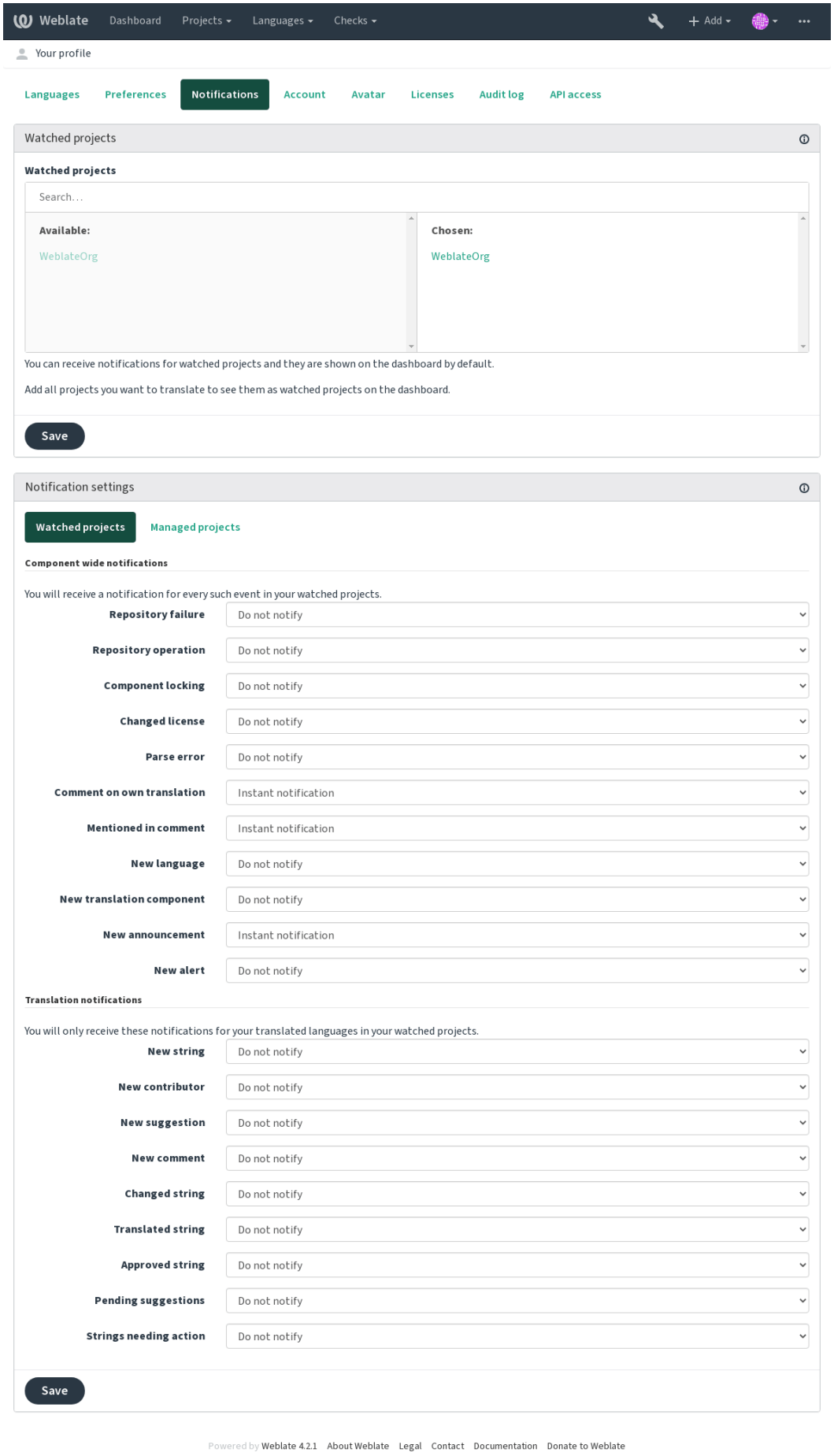
1.2.4 通知

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 page and select appropriate choice from the *Watching* menu.

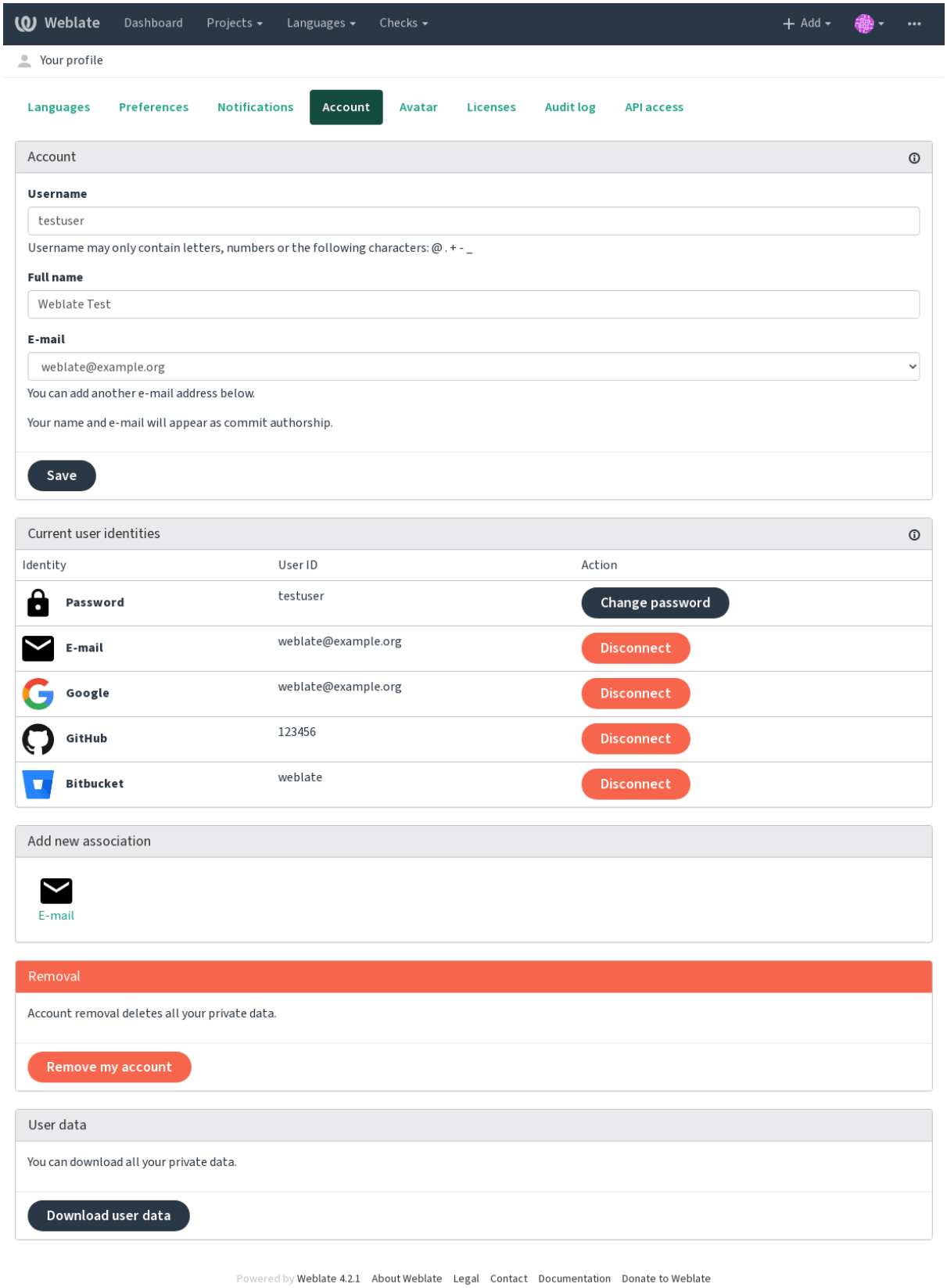
注解： You will not receive notifications for your own actions.


Weblate
Dashboard
Projects
Languages
Checks
Add
Your profile
Languages
Preferences
Notifications
Account
Avatar
Licenses
Audit log
API access
Watched projects
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
Watched projects
Managed projects
Component wide notifications
You will receive a notification for every such event in your watched projects.
Repository failure
Repository operation
Component locking
Changed license
Parse error
Comment on own translation
Mentioned in comment
New language
New translation component
New announcement
New alert
Do not notify
Instant notification
Translation notifications
You will only receive these notifications for your translated languages in your watched projects.
New string
New contributor
New suggestion
New comment
Changed string
Translated string
Approved string
Pending suggestions
Strings needing action
Save
Powered by Weblate 4.2.1
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.2.5 账户

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*).

注解： 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
Add
Your profile
Languages
Preferences
Notifications
Account
Avatar
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 4.2.1
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.2.6 审计日志

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.

参见:

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 accepts only suggestions, which are automatically validated once a defined number of votes is reached

Please see 翻译工作流 for more information on translation workflow.

Options for translation project visibility:

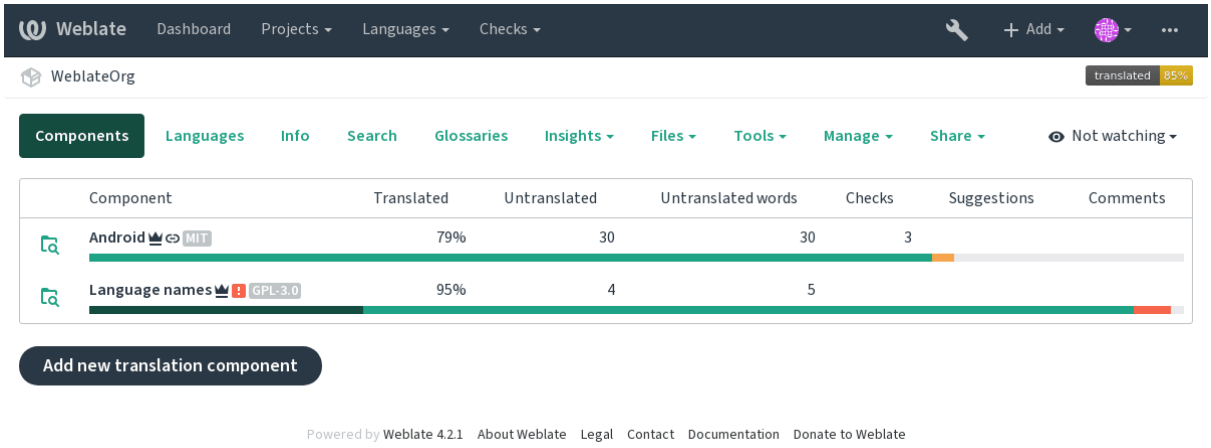
- Publicly visible and anybody can contribute
- Visible only to a certain group of translators

参见:

访问控制, 翻译工作流

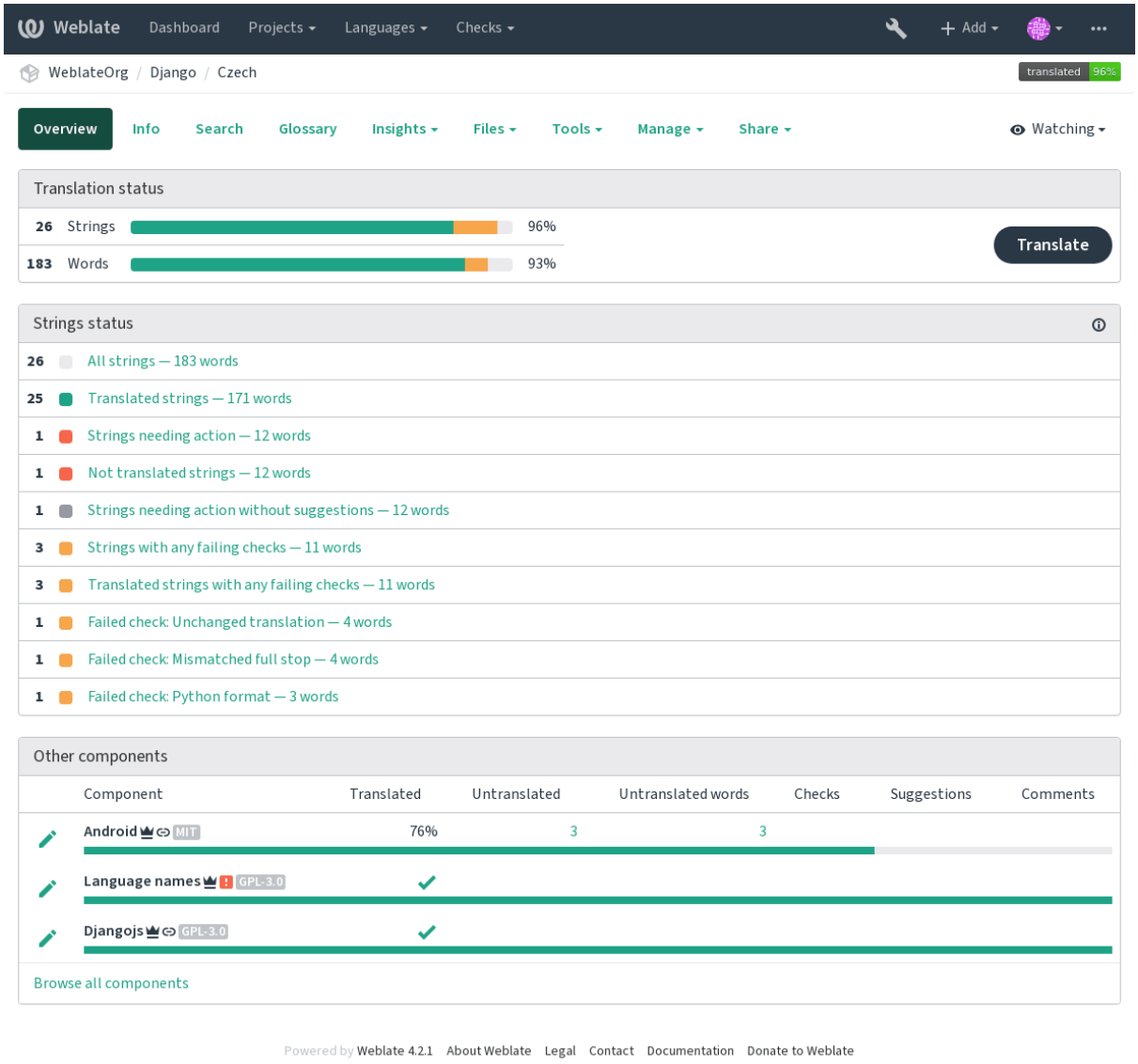
1.3.1 翻译项目

Translation projects hold related components, related to the same software, book, or project.



1.3.2 Translation links

Having navigated to a component, a set of links lead to actual translation. The translation is further divided into individual checks, like *Untranslated* or *Needing review*. If the whole project is translated, without error, *All translations* is still available. Alternatively you can use the search field to find a specific string or term.



1.3.3 建议

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

Anonymous users can only (if permitted) forward suggestions. Doing so is still available to signed in users, in cases where uncertainty about the translation arises, which will prompt another translator to review it.

The suggestions are scanned on a daily basis to remove duplicate ones or suggestions that match the current translation.

1.3.4 注释

The comments can be posted in two scopes - source string or translation. Choose the one which matches the topic you want to discuss. The source string comments are good for providing feedback on the original string, for example that it should be rephrased or it is confusing.

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

1.3.5 变体

Variants are used to group variants of the string in different lengths. The frontend can use different strings depending on the screen or window size.

参见:

String variants

1.3.6 标签

Labels are used to categorize strings within a project. These can be used to further customize the localization workflow, for example to define categories of strings.

参见:

String labels

1.3.7 Translating

On the translation page, the source string and an edit area for translating are shown. Should the translation be plural, multiple source strings and edit areas are shown, each described and labeled in plural form.

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 information 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). When you choose secondary languages in your preferences, translation to these languages will be shown (see 第二语言) above the source string.

Below the translation, any suggestion made by others will be shown, which you can in turn accept, accept with changes, or delete.

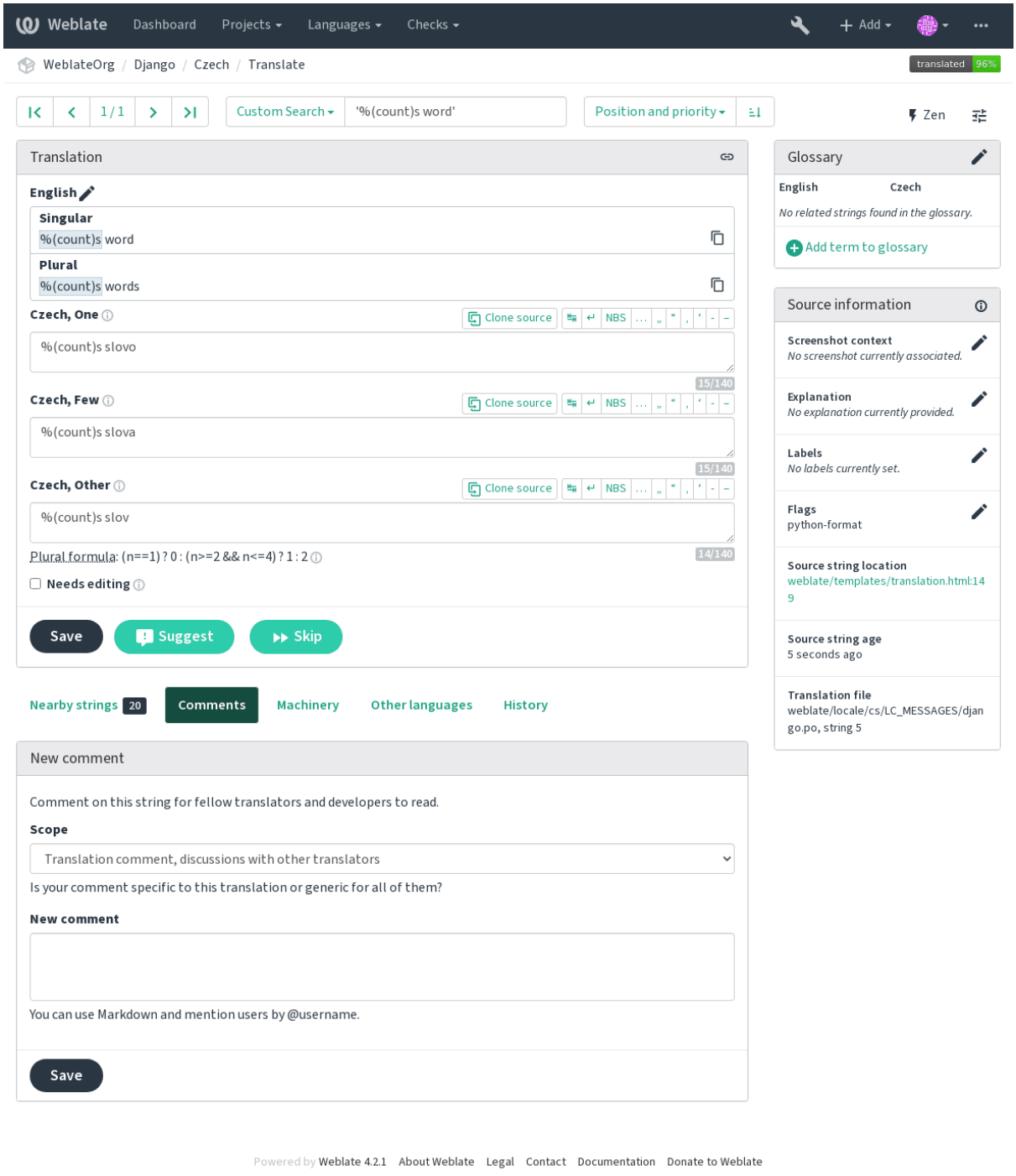
复数形式

Words that change form to account of their numeric designation are called plurals. Each language has its own definition of plurals. English, for example, supports one plural. 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 used in the translated application depends on the configured plural formula. Weblate shows the basic information, but you can find a more detailed description in the Language Plural Rules by the Unicode Consortium.

参见:

复数式



Keyboard shortcuts

在 2.18 版更改: 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:

Alt+Home 导航到当前搜索中的第一个翻译。

Alt+End 导航到当前搜索中的最后一个翻译。

Alt+PageUp 导航到当前搜索中的前一处翻译。

Alt+PageDown 导航到当前搜索中的下一处翻译。

Alt+Enter, Ctrl+Enter, or Cmd+Enter 保存当前翻译。

Ctrl+Shift+Enter or Cmd+Shift+Enter Unmarks translation as fuzzy and submits it.

Ctrl+E or Cmd+E Focus translation editor.

Ctrl+U or Cmd+U Focus comment editor.

Ctrl+M or Cmd+M Shows machine translation tab.

Ctrl+<NUMBER> or Cmd+<NUMBER> Copies placeable of given number from source string.

Ctrl+M <NUMBER> or Cmd+M <NUMBER> Copy the machine translation of given number to current translation.

Ctrl+I <NUMBER> or Cmd+I <NUMBER> 忽略失败检查列表中的一个项目。

Ctrl+J or Cmd+J 显示:guilabel: ‘附近字符串 ‘选项卡。

Ctrl+S or Cmd+S Shows search tab.

Ctrl+O or Cmd+O Copies source string.

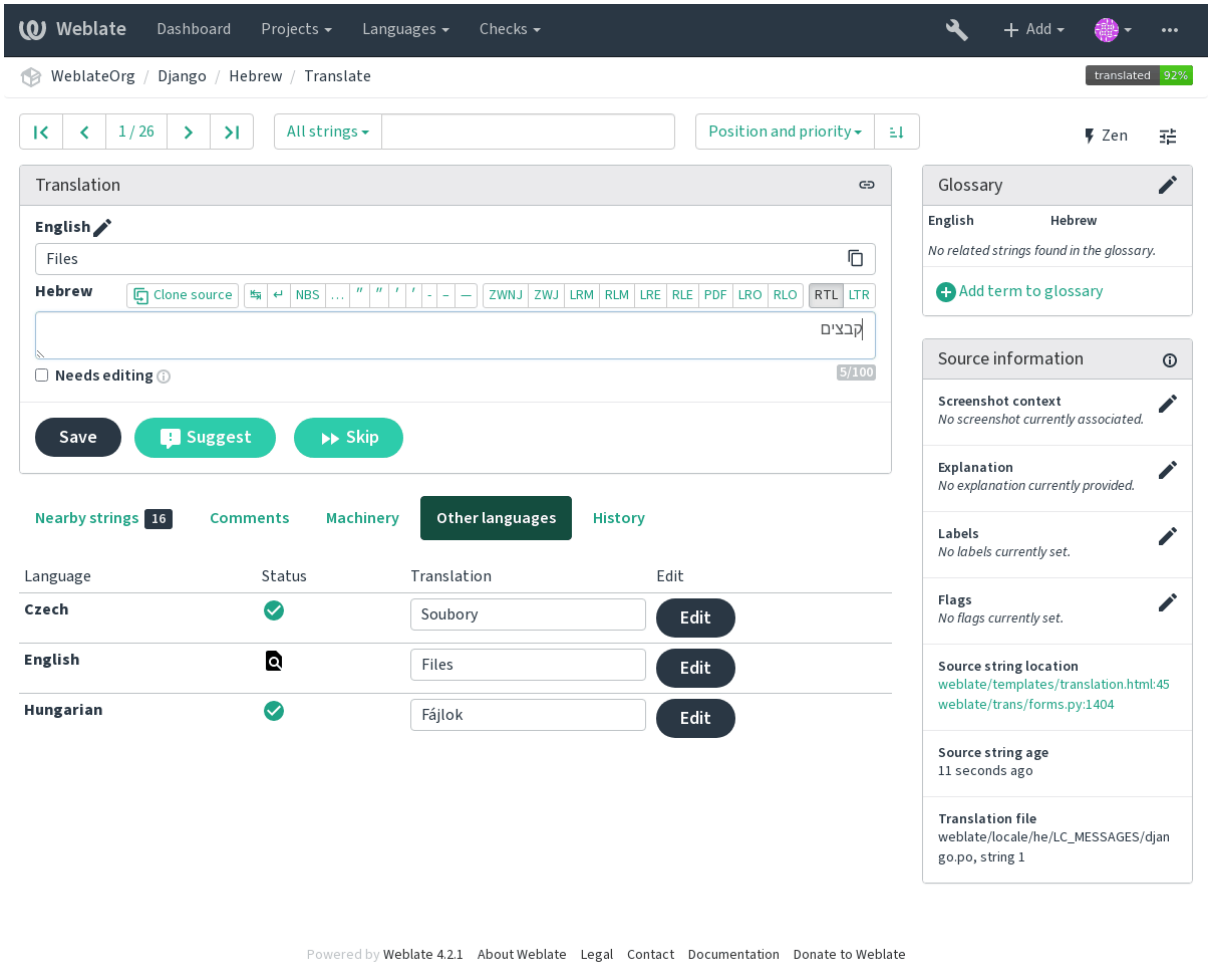
Ctrl+Y or Cmd+Y Toggles the *Needs editing* flag.

Visual keyboard

A small visual keyboard is shown just above the translation field. This can be useful for typing characters not usually found or otherwise hard to type.

The shown symbols factor into three categories:

- User configured characters defined in the 用户资料
- Per language characters provided by Weblate (e.g. quotes or RTL specific characters)
- Chars configured using *SPECIAL_CHARS*



Translation context

This contextual description provides related information about the current string.

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

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

附近字符串 Displays neighbouring messages from the translation file. These are usually also used in a similar context and prove useful in keeping the translation consistent.

其它的出现位置 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 不一致的). You can choose which one to use.

翻译记忆库 Look at similar strings translated in past, see Memory Management.

词汇表 Displays terms from the project glossary used in the current message.

最近的变更 List of people whom have changed this message recently using Weblate.

项目 Project information like instructions for translators, or information about its version control system repository.

If the translation format supports it, you can also follow supplied links to respective source code containing each source string.

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

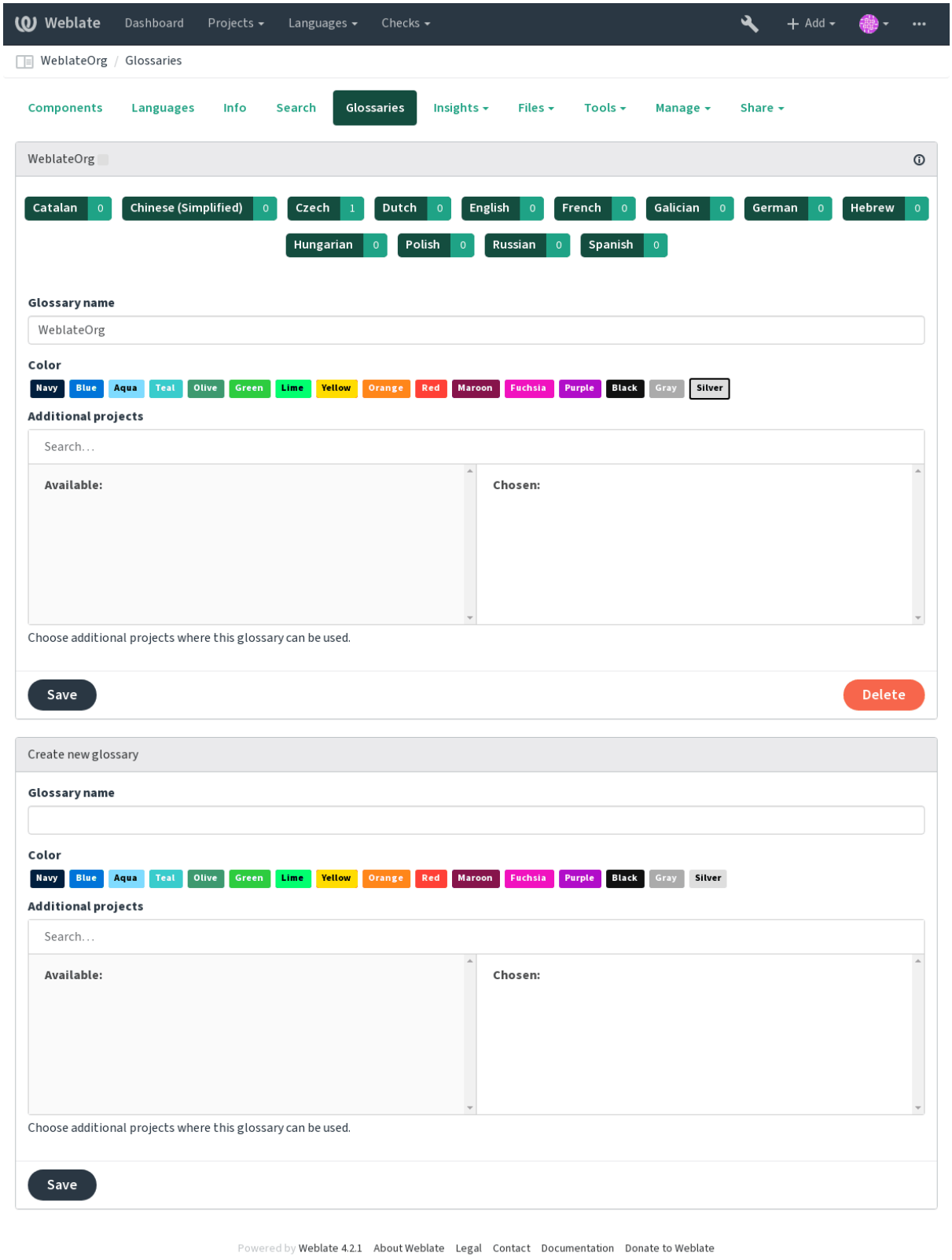
- The default limitation for translation is ten times longer than source string. This can be turned of by *LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH*. In case you are hitting this, it might be also caused by monolingual translation being configured as bilingual, making Weblate see translation key as source string instead of the actual source string. See 双语和单语格式 for more info.
- Maximal length in characters defined by translation file or flag, see 译文最大长度.
- Maximal rendered size in pixels defined by flags, see 最大翻译大小.

1.3.8 词汇表

Each project can have an assigned glossary for any language as a shorthand for storing terminology. Consistency is more easily maintained this way. Terms from the currently translated string can be displayed in the bottom tabs.

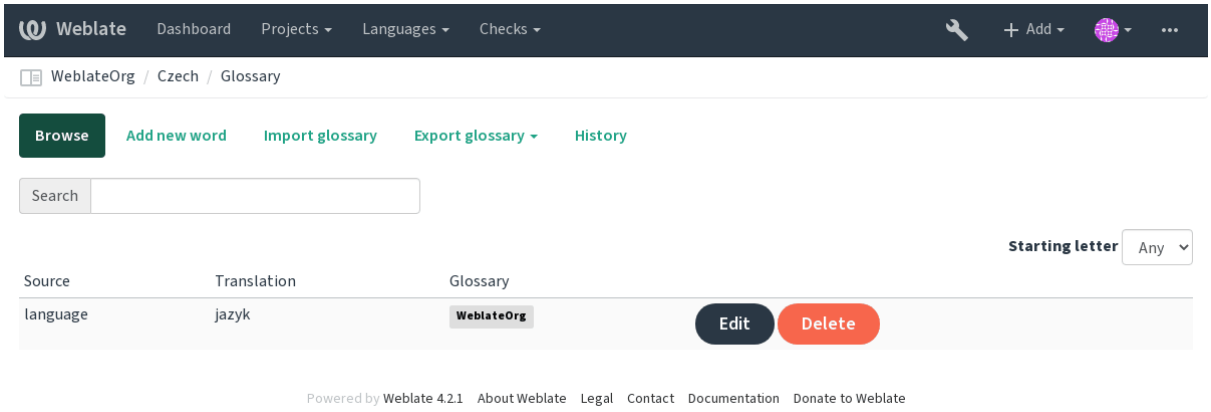
Managing glossaries

On the *Glossaries* tab of each project page, you can edit existing glossaries.



An empty glossary for a given project is automatically created when project is created. Glossaries are shared among all components of the same project and you can also choose to share them with another projects. You can do this only for projects you can administer.

On this list, you can choose which glossary to manage (all languages used in the current project are shown). Following one of the language links will lead you to a page which can be used to edit, import or export the selected glossary, or view the edit history:



1.3.9 机器翻译

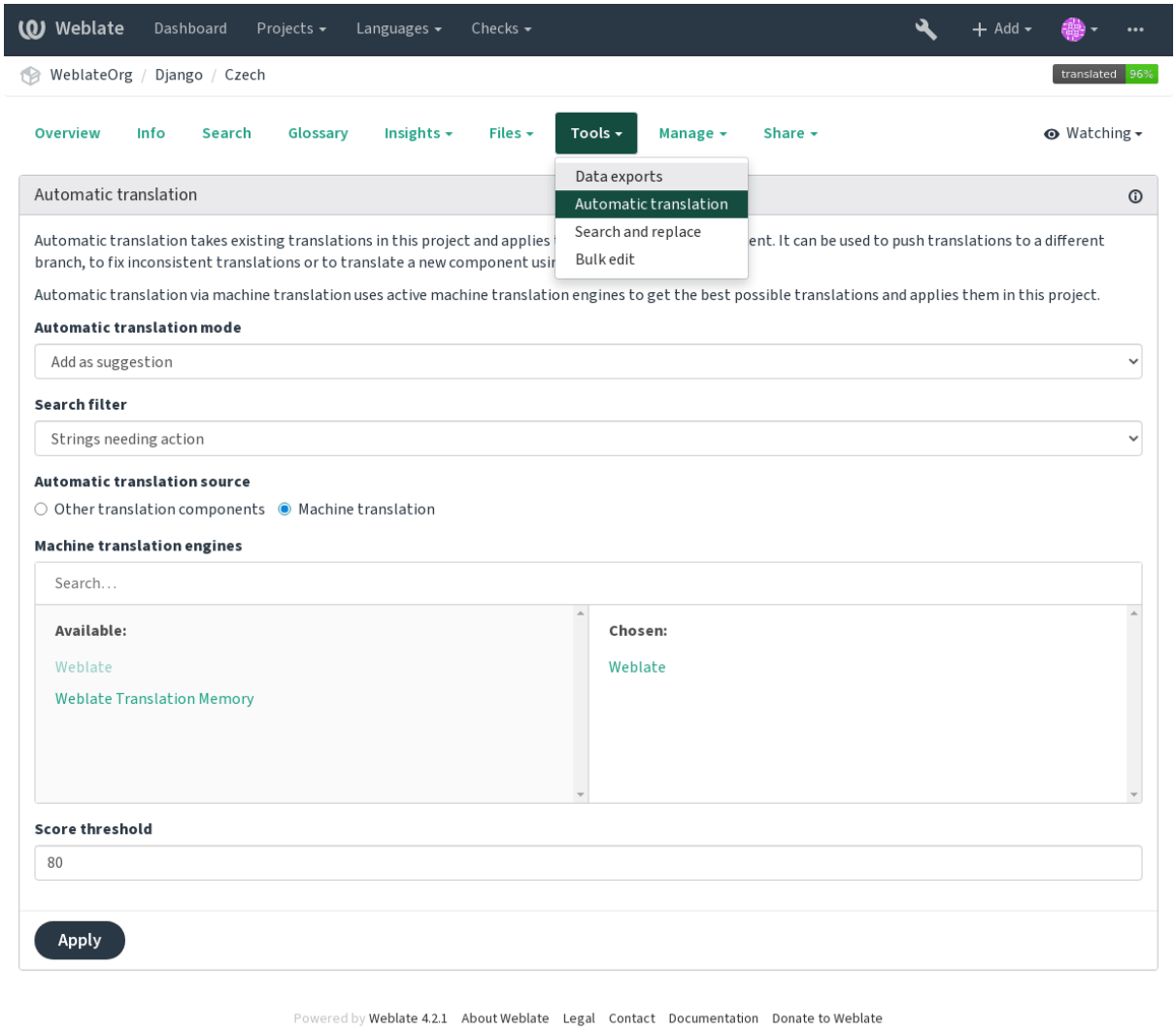
Based on configuration and your translated language, Weblate provides you suggestions from several machine translation tools. All machine translations are available in a single tab of each translation page.

参见:

You can find the list of supported tools in 机器翻译.

1.3.10 自动化翻译

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.

警告： 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 website and application) or when bootstrapping translation for a new component using existing translations (translation memory).

参见：

在部件之间保持翻译一致

1.3.11 Rate limiting

To avoid abuse of the interface, there is rate limiting applied to several operations like searching, sending contact form or translating. In case you are hit by this, you are blocked for a certain period until you can perform the operation again.

The default limits are described in the administrative manual in *Rate limiting*, but can be tweaked by configuration.

1.3.12 批量编辑

Bulk edit allows you to perform operation on number of strings. You define search strings and operation to perform and all matching strings are updated. Following operations are supported:

- Changing string state (for example to approve all strings waiting for review)
- Adjust translation flags (see 定制行为)
- Adjust string labels (see *String labels*)

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

参见：

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.

注解： The available options might be limited by 访问控制.

1.4.1 Downloading translations

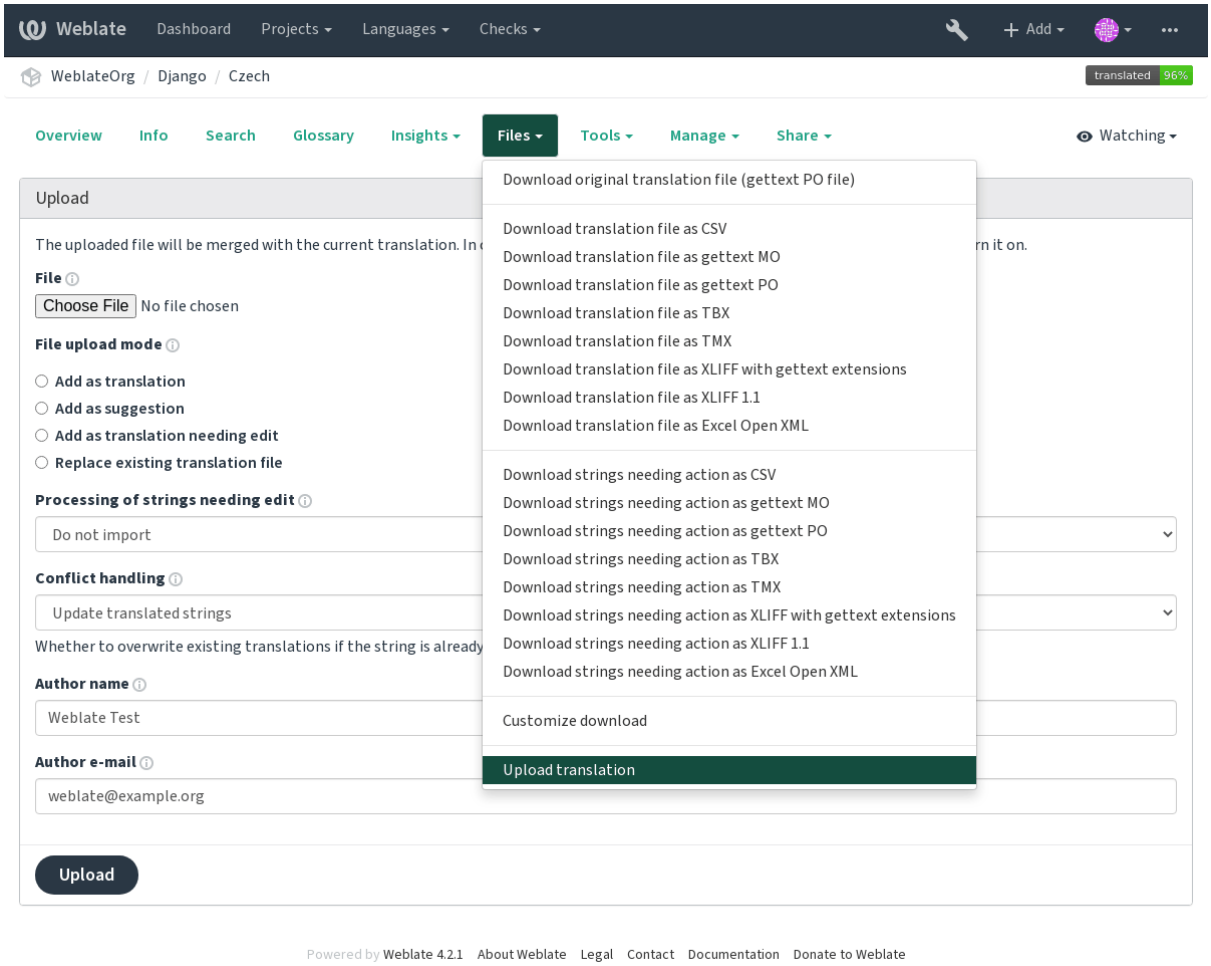
From the project or component dashboard, translatable files can be downloaded using the *Download original translation file* in the *Files* menu, producing a copy of the original file as it is stored in the upstream Version Control System.

You can also download the translation converted into one of 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, including a compiled file to use in your choice of application (for example `.mo` files for GNU Gettext) using the *Files* menu.

1.4.2 Uploading translations

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



支持的文件格式

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.

参见:

支持的文件格式

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:

添加为翻译 (`translate`) Imported translations are added as translations. This is the most common usecase, and the default behavior.

添加为建议 (`suggest`) Imported translations are added as suggestions, do this when you want to have your uploaded strings reviewed.

添加为需要编辑的翻译 (`fuzzy`) Imported translations are added as translations needing edit. This can be useful when you want translations to be used, but also reviewed.

替换现有翻译文件 (`replace`) Existing file is replaced with new content. This can lead to loss of existing translations, use with caution.

更新源字符串 (`source`) Updates source strings in bilingual translation file. This is similar to what *更新 PO 文件以匹配 POT 文件 (msgmerge)* does.

参见:

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".

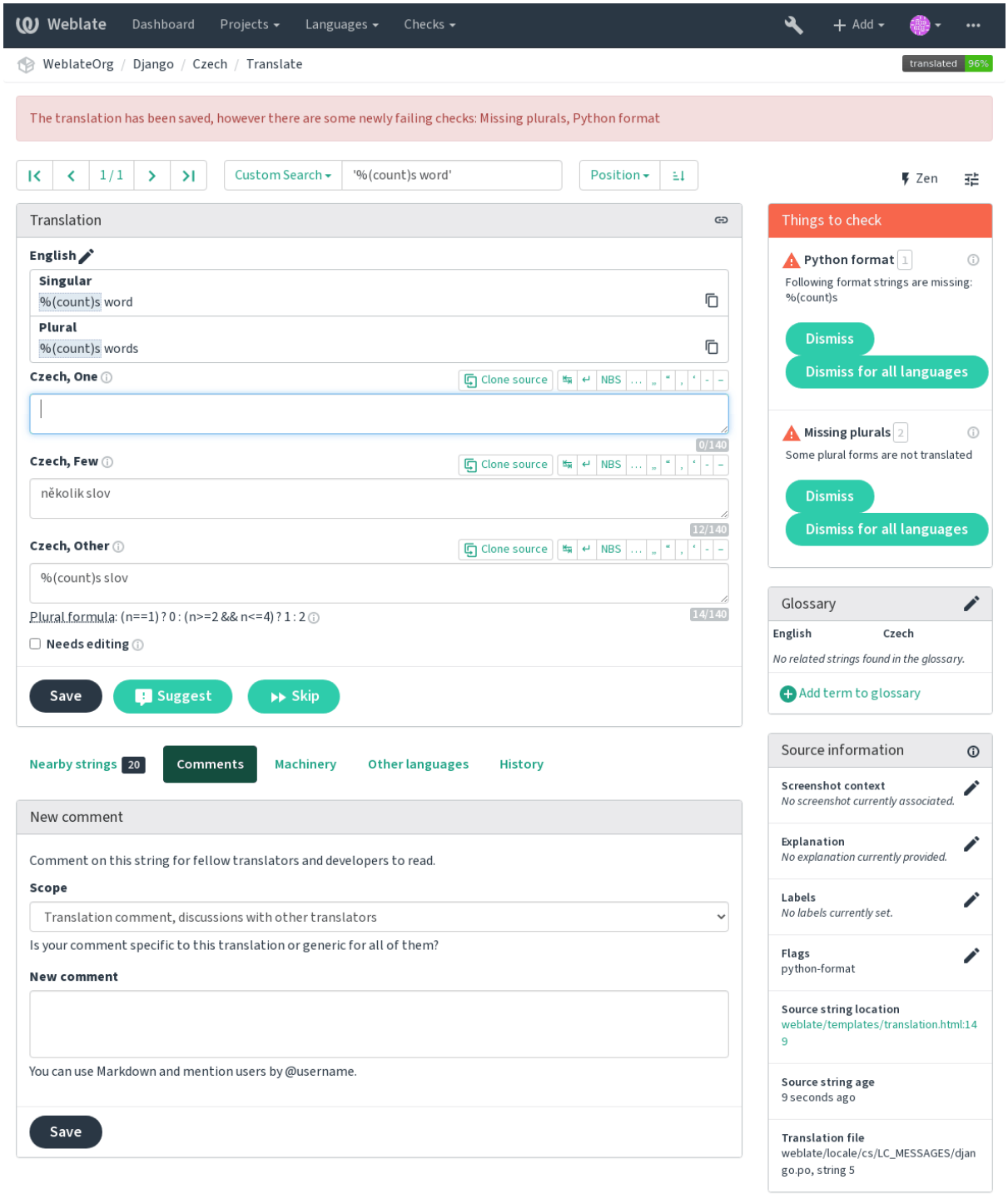
Overriding authorship

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 检查并修正

质量检查有助于发现常见的翻译错误，确保翻译质量良好。如果出现误报，则可以忽略这些检查。

提交未通过检查的翻译后，将立即向用户显示：


Weblate
Dashboard
Projects
Languages
Checks
Add
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
Clone source
NBS
0/140
Czech, Few
Clone source
NBS
několik slov
12/140
Czech, Other
Clone source
NBS
%(count)s slov
14/140
Plural formula: (n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2
Needs editing
Save
Suggest
Skip
Nearby strings 20
Comments
Machinery
Other languages
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
Dismiss for all languages
Missing plurals 2
Some plural forms are not translated
Dismiss
Dismiss for all languages
Glossary
English
Czech
No related strings found in the glossary.
Add term to glossary
Source information
Screenshot context
No screenshot currently associated.
Explanation
No explanation currently provided.
Labels
No labels currently set.
Flags
python-format
Source string location
weblate/templates/translation.html:149
Source string age
9 seconds ago
Translation file
weblate/locale/cs/LC_MESSAGES/django.po, string 5
Powered by Weblate 4.2.1
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.5.1 自动修正

除了质量检查 外，Weblate 还可以自动修复翻译字符串中的一些常见错误。谨慎使用它，不要使其增加错误。

参见：

AUTOFIX_LIST

1.5.2 质量检查

Weblate 对字符串进行了广泛的质量检查。以下部分将对它们进行更详细的描述。还有针对特定语言的检查。如果有错误报告，请提交错误。

参见：

CHECK_LIST, *定制行为*

1.5.3 翻译检查

在每次翻译更改时执行，帮助翻译人员保持高质量的翻译。

BBcode 标记

翻译中的 *BBcode* 与来源不符

BBCode 表示简单的标记，例如以粗体或斜体突出显示消息的重要部分。

此检查确保在翻译中也找到它们。

注解： 当前检测 BBcode 的方法非常简单，因此此检查可能会产生误报。

连续重复的单词

文本连续两次包含相同的单词：

4.1 新版功能.

检查翻译中是否没有连续重复的单词。这通常表示翻译错误。

提示： 此检查包括特定于语言的规则，以避免误报。如果在您的情况下错误触发，请告诉我们。请参阅*Reporting issues in Weblate*。

双空格

翻译包含双空格

检查翻译中是否存在双空格，以避免其他与空格相关的检查出现误报。

当在源中找到双空格时，检查为假，这意味着故意使用双空格。

格式化字符串

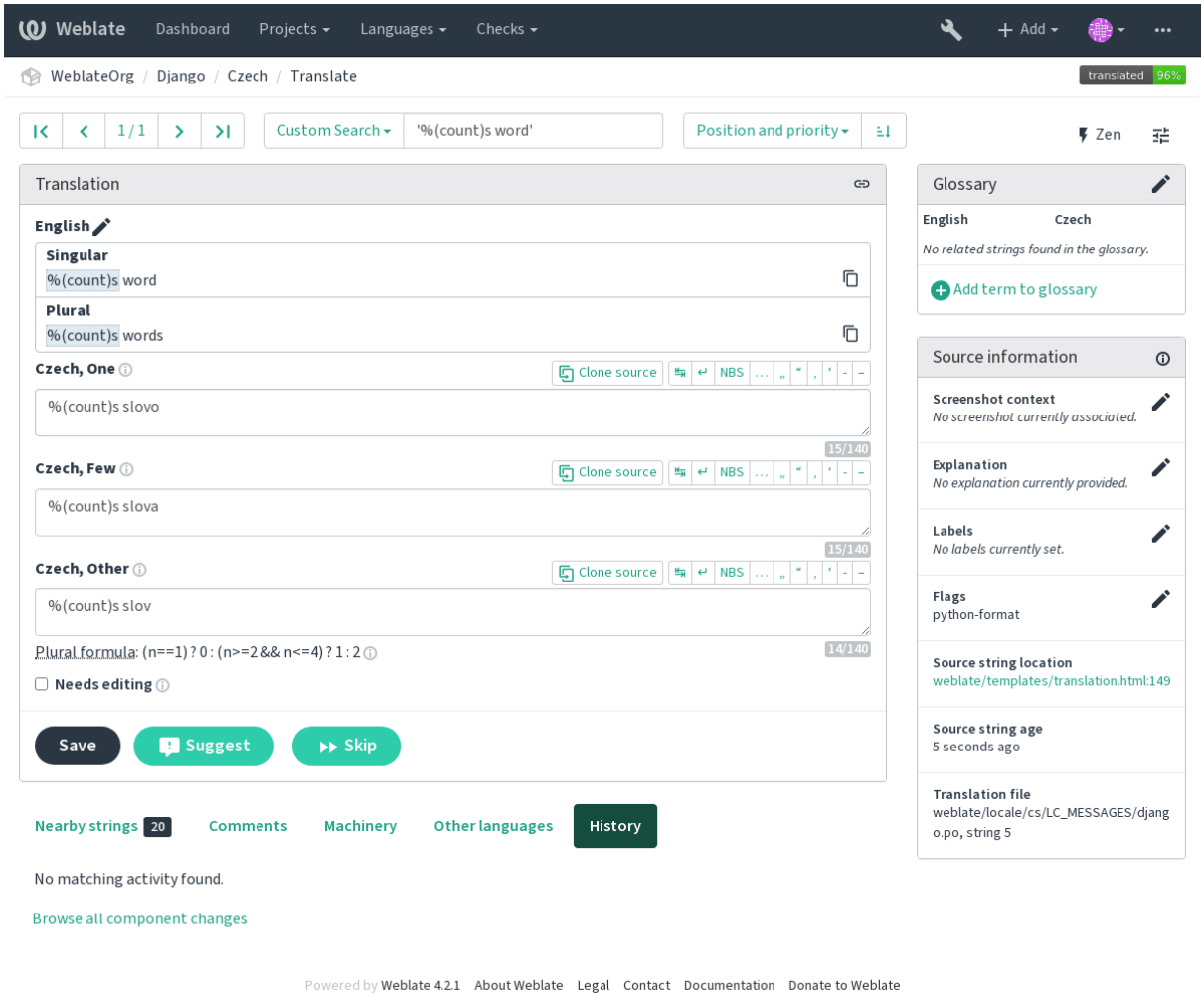
检查字符串格式是否在源和翻译之间复制。在翻译中省略格式字符串通常会导致严重的问题，因此字符串中的格式通常应与源匹配。

Weblate 支持检查几种语言的格式字符串。仅当适当地标记了字符串时（例如，C 格式为 *c-format*），才会自动启用该检查。Gettext 会自动添加它，但是对于其他文件格式，或者如果您的 PO 文件不是由 **xgettext** 生成的，您可能必须手动添加它。

可以按单位（请参阅*Additional info on source strings*）或在*Component configuration* 中完成此操作。为每个组件定义它比较简单，但是如果该字符串未解释为格式化字符串，而碰巧使用了格式化字符串语法，则可能导致误报。

提示： 如果 Weblate 中不提供特定格式的检查，则可以使用通用占位符。

除了检查，这也将高亮格式化字符串，方便将它们插入到已翻译字符串:



AngularJS 插值字符串

AngularJS interpolation strings do not match source

Named format string	`Your balance is {{amount}} {{ currency }}`
Flag to enable	*angularjs-format*

参见:

AngularJS: API: $interpolate

C 格式

C format string does not match source

Simple format string	`There are %d apples`
Position format string	`Your balance is %1$d %2$s`
Flag to enable	*c-format*

参见:

C format strings, C printf format

C# 格式

C# format string does not match source

Position format string	`There are {0} apples`
Flag to enable	*c-sharp-format*

参见:

C# String Format

ECMAScript 模板文字

ECMAScript 模板文字与源不匹配

Interpolation	`There are ${number} apples`
Flag to enable	*es-format*

参见:

Template literals

i18next 插补

The i18next interpolation does not match source

4.0 新版功能.

Interpolation	`There are {{number}} apples`
Nesting	`There are $t(number) apples`
Flag to enable	*i18next-interpolation*

参见:

i18next interpolation

Java 格式

Java format string does not match source

Simple format string	`There are %d apples`
Position format string	`Your balance is %1$d %2$s`
Flag to enable	*java-format*

参见:

Java Format Strings

Java MessageFormat

Java MessageFormat string does not match source

Position format string	`There are {0} apples`
Flag to enable	*java-messageformat* enables the check unconditionally
	auto-java-messageformat enables check only if there is a format string in the source

参见:

Java MessageFormat

JavaScript 格式

JavaScript format string does not match source

Simple format string	`There are %d apples`
Flag to enable	*javascript-format*

参见:

JavaScript formatting strings

百分比占位符

The percent placeholders do not match source

4.0 新版功能.

Simple format string	`There are %number% apples`
Flag to enable	*percent-placeholders*

Perl 格式

Perl format string does not match source

Simple format string	`There are %d apples`
Position format string	`Your balance is %1$d %2$s`
Flag to enable	*perl-format*

参见:

Perl sprintf, Perl Format Strings

PHP 格式

PHP format string does not match source

Simple format string	`There are %d apples`
Position format string	`Your balance is %1$d %2$s`
Flag to enable	*php-format*

参见:

PHP sprintf documentation, PHP Format Strings

Python brace 格式

Python brace format string does not match source

Simple format string	`There are {} apples`
Named format string	`Your balance is {amount} {currency}`
Flag to enable	*python-brace-format*

参见:

Python brace format, Python Format Strings

Python 格式

Python format string does not match source

Simple format string	`There are %d apples`
Named format string	`Your balance is %(amount) %(currency)`
Flag to enable	*python-format*

参见:

Python string formatting, Python Format Strings

Qt 格式

Qt format string does not match source

Position format string	`There are %1 apples`
Flag to enable	*qt-format*

参见:

Qt QString::arg()

Qt 复数格式

Qt plural format string does not match source

Plural format string	`There are %Ln apple(s)`
Flag to enable	*qt-plural-format*

参见:

Qt i18n guide

Ruby 格式

Ruby format string does not match source

Simple format string	`There are %d apples`
Position format string	`Your balance is %1$f %2$s`
Named format string	`Your balance is %+.2<amount>f %<currency>s`
Named template string	`Your balance is %{amount} %{currency}`
Flag to enable	*ruby-format*

参见:

Ruby Kernel#sprintf

已被翻译

This string has been translated in the past

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

不一致的

This string has more than one translation in this project or is not translated in some components.

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.

注解： 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 occurences* tab.

You can use 自动化翻译 addon to automate translating of newly added strings which are already translated in another component.

参见：

在部件之间保持翻译一致

已使用 Kashida 字符

The decorative kashida letters should not be used

3.5 新版功能.

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

参见：

Kashida on Wikipedia

Markdown 链接

Markdown links do not match source

3.5 新版功能.

Markdown links do not match source.

参见：

Markdown links

Markdown 引用

Markdown link references do not match source

3.5 新版功能.

Markdown link references do not match source.

参见:

Markdown links

Markdown 语法

Markdown syntax does not match source

3.5 新版功能.

Markdown 语法与原文不匹配

参见:

Markdown span elements

译文最大长度

Translation should not exceed given length

检查翻译的长度是否可匹配可用的空间。这只检查翻译字符的长度。

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

提示: This checks looks at number of chars, what might not be the best metric when using proportional fonts to render the text. The 最大翻译大小 check does check actual rendering of the text.

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

最大翻译大小

Translation rendered text should not exceed given size

3.7 新版功能.

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 定制行为), 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
```

提示: 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.

参见:

管理字型, 定制行为, 译文最大长度

n 数量

Number of n in translation does not match source

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

不匹配的冒号

Source and translation do not both end with a 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).

参见:

Colon on Wikipedia

不匹配的省略号

Source and translation do not both end with an 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.

参见:

Ellipsis on Wikipedia

不匹配的感叹号

Source and translation do not both end with an exclamation mark

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).

参见:

Exclamation mark on Wikipedia

不匹配的句号

Source and translation do not both end with a full 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).

参见:

Full stop on Wikipedia

不匹配的问号

源文件和译文没有都以问号结尾

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).

参见:

Question mark on Wikipedia

不匹配的分号

Source and translation do not both end with a 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.

参见:

Semicolon on Wikipedia

换行符不符

Number of new lines in translation does not match source

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

缺少复数形式

Some plural forms are not translated

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.

占位符

Translation is missing some placeholders:

3.9 新版功能.

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"
```

参见:

定制行为

标点间距

Missing non breakable space before double punctuation sign

3.9 新版功能.

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.

参见:

French and English spacing on Wikipedia

正则表达式

Translation does not match regular expression:

3.9 新版功能.

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

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

相同复数

Some plural forms are translated in the same way

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

换行开头

Source and translation do not both start with a 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.

参见:

换行结尾

空格开头

Source and translation do not both start with same number of spaces

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

换行结尾

Source and translation do not both end with a 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.

参见:

换行开头

空格结尾

Source and translation do not both end with a 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.

未更改的翻译

Source and translation are identical

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.

参见:

Component configuration, 定制行为

不安全的 HTML 网站

The translation uses unsafe HTML markup

3.9 新版功能.

The translation uses unsafe HTML markup. This check has to be enabled using `safe-html` flag (see 定制行为). There is also accompanied autofixer which can automatically sanitize the markup.

参见:

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

URL

The translation does not contain an URL

3.5 新版功能.

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.

XML 标记

XML tags in translation do not match source

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.

XML 语法

The translation is not valid XML

2.8 新版功能.

The XML markup is not valid.

零宽空格

Translation contains extra zero-width space character

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.

参见:

Zero width space on Wikipedia

1.5.4 Source checks

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

省略号

The string uses three dots (...) instead of an ellipsis character (…)

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.

参见:

Ellipsis on Wikipedia

长期未翻译

The string has not been translated for a long time

4.1 新版功能.

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.

多项检查失败

The translations in several languages have failing checks

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.

多个未命名的变量

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

4.1 新版功能.

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.

未复数化

The string is used as plural, but not using plural forms

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:

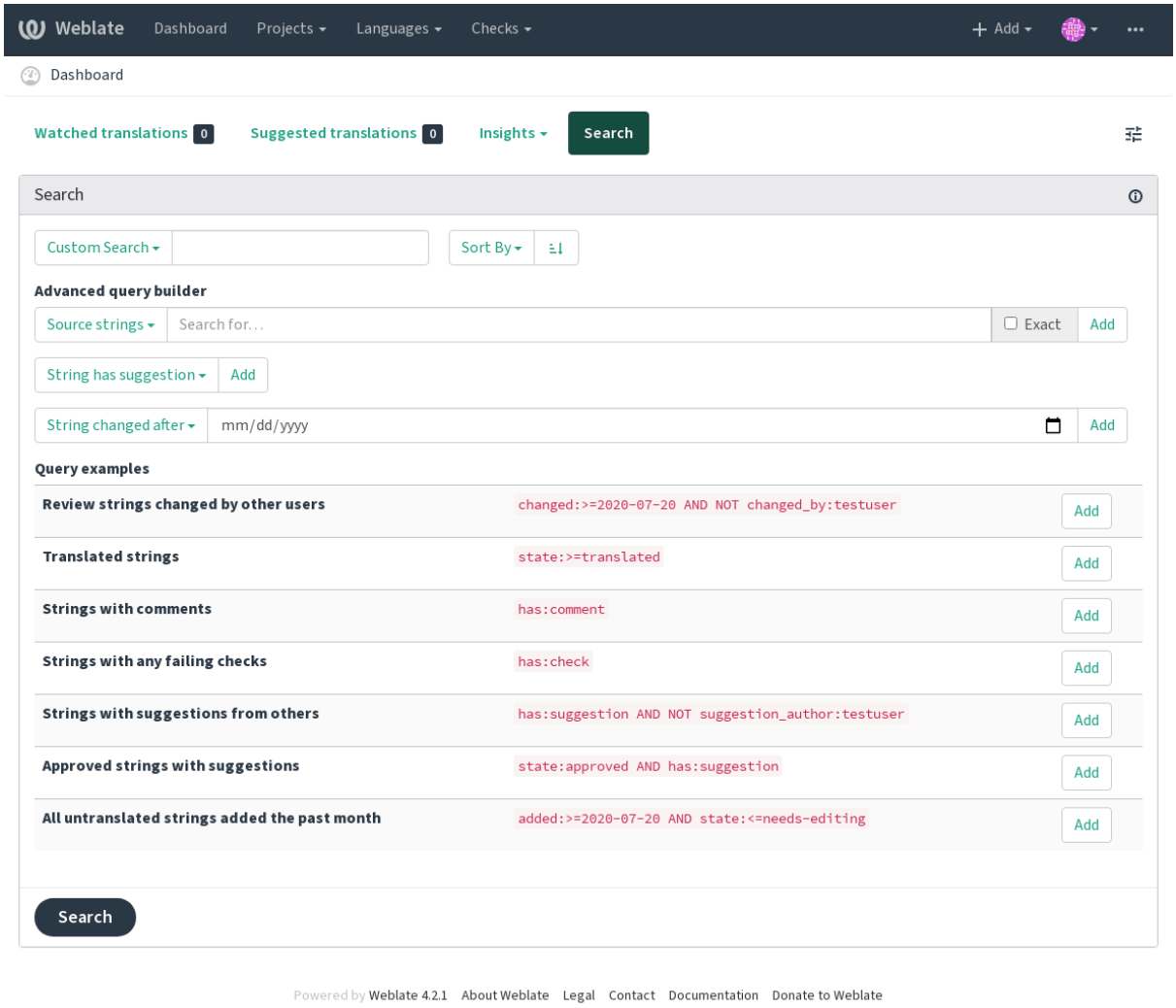
```
from gettext import ngettext

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

1.6 Searching

3.9 新版功能.

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*, *Target* and *Context* fields.



1.6.1 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'`.

1.6.2 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` (works only on source strings).

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

language:TEXT String target language.

组件:TEXT slug 组件, 见*Component slug*.

项目:TEXT Project slug, see *Project slug*.

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

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

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.

1.6.3 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)`

1.6.4 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.

1.6.5 Exact operators

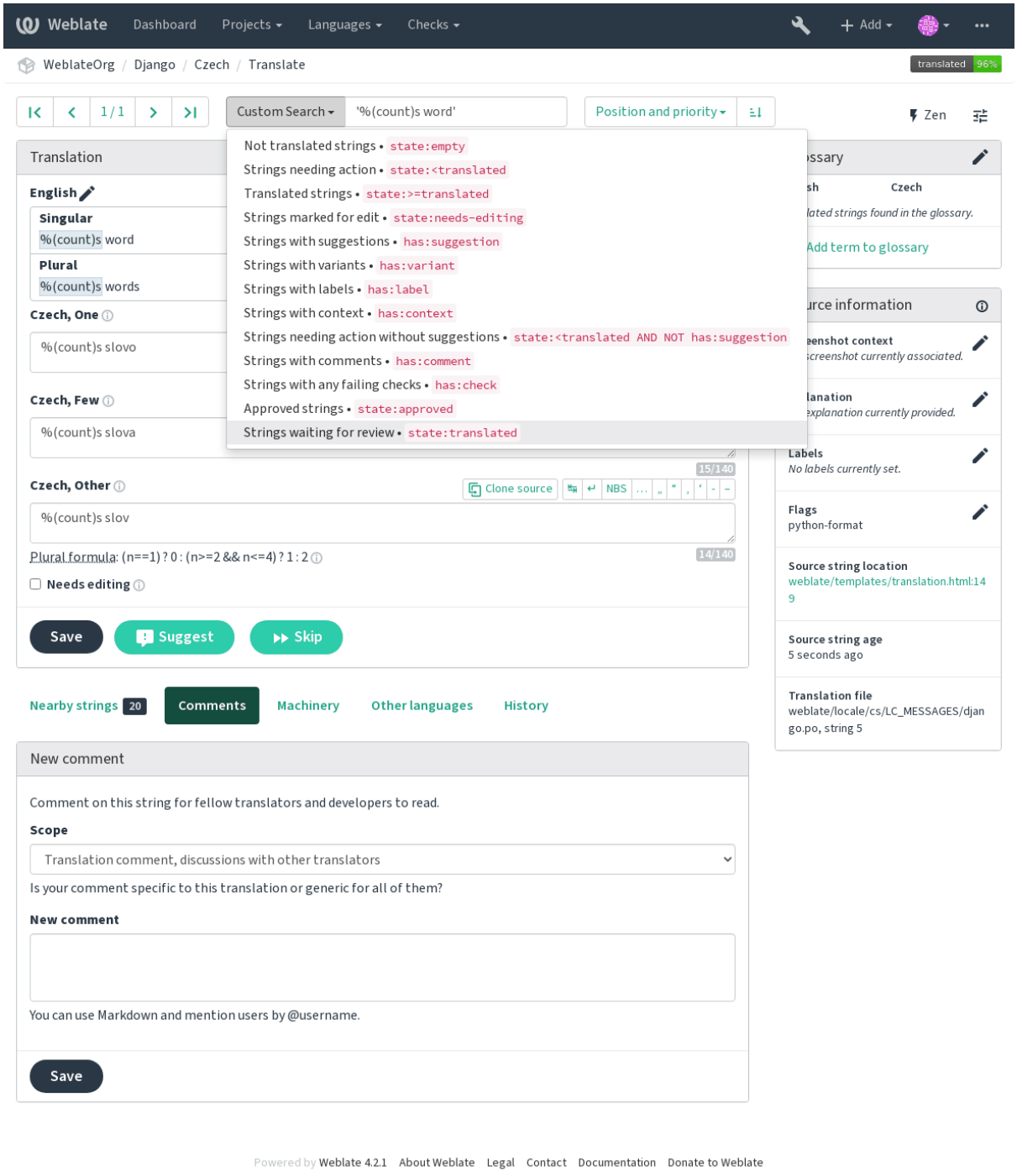
You can do an exact match query on different string fields using = operator. For example, to search for all source strings exactly matching `hello world`, use: `source:="hello world"`. For searching single word expressions, you can skip quotes. For example, to search for all source strings matching `hello`,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

1.6.6 Regular expressions

Anywhere text is accepted you can also specify a regular expression as `r"regexp"`. For instance, to search for all source strings which contain any digit between 2 and 5, use: `source:r"[2-5]"`

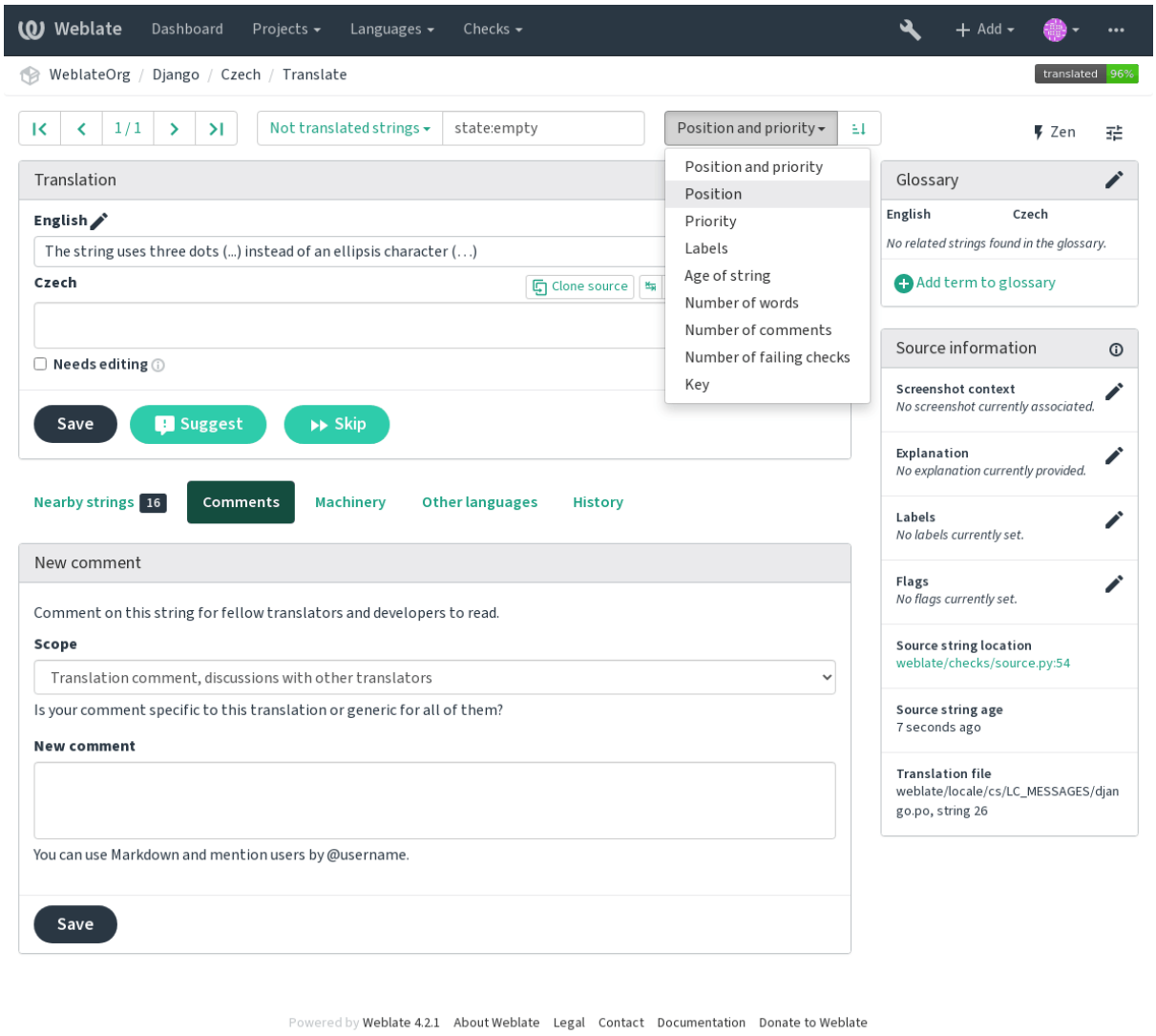
1.6.7 Predefined queries

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



1.6.8 Ordering the results

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



1.7 Application developer guide

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

1.7.1 Starting with internationalization

Have a project and want to translate it into several languages? This guide will help you do so. Several typical situations are showcased, but most of the examples are generic and can be applied to other scenarios as well.

Before translating any software, you should realize that languages around the world are really different and you should not make any assumption based on your experience. For most of languages it will look weird if you try to concatenate a sentence out of translated segments. You also should properly handle plural forms because many languages have complex rules for that and the internationalization framework you end up using should support this.

Last but not least, sometimes it might be necessary to add some context to the translated string. Imagine a translator would get string `Sun` to translate. Without context most people would translate that as our closest star, but it might

be actually used as an abbreviation for Sunday.

Choosing internationalization framework

Choose whatever is standard on your platform, try to avoid reinventing the wheel by creating your own framework to handle localizations. Weblate supports most of the widely used frameworks, see 支持的文件格式 for more information (especially 翻译类型功能).

Our personal recommendation for some platforms is in the following table. This is based on our experience, but that can not cover all use cases, so always consider your environment when doing the choice.

Platform	Recommended format
Android	*Android string resources*
iOS	*Apple iOS strings*
Qt	*Qt Linguist .ts*
Python	*GNU gettext*
PHP	*GNU gettext*[1]
C/C++	*GNU gettext*
C#	*.XML resource files*
Perl	*GNU gettext*
Ruby	*Ruby YAML files*
Web extensions	*WebExtension JSON*
Java	*XLIFF*[2]
JavaScript	*JSON i18next files*[3]

The more detailed workflow for some formats is described in following chapters:

- *Translating software using GNU Gettext*
- *Translating documentation using Sphinx*
- *Translating HTML and JavaScript using Weblate CDN*

Integrating with Weblate

Getting translations updates from Weblate

To fetch updated strings from Weblate you can simply fetch the underlying repository (either from filesystem or it can be made available through *Git exporter*). Prior to this, you might want to commit any pending changes (see 惰性提交). This can be achieved in the user interface (in the *Repository maintenance*) or from command line using *Weblate 客户端*.

This can be automated if you grant Weblate push access to your repository and configure *Push URL* in the *Component configuration*.

参见:

持续本地化集成

[1] The native Gettext support in PHP is buggy and often missing on Windows builds, it is recommended to use third party library motranslator instead.

[2] You can also use *Java properties* if plurals are not needed.

[3] You can also use plain *JSON files* if plurals are not needed.

Pushing string changes to Weblate

To push newly updated strings to Weblate, just let it pull from the upstream repository. This can be achieved in the user interface (in the *Repository maintenance*) or from command line using *Weblate 客户端*.

This can be automated by installing a webhook on your repository to trigger Weblate whenever there is a new commit, see 更新仓库 for more details.

参见:

持续本地化集成

1.7.2 Translating software using GNU Gettext

GNU Gettext is one of the most widely used tool for internationalization of free software. It provides a simple yet flexible way to localize the software. It has great support for plurals, it can add further context to the translated string and there are quite a lot of tools built around it. Of course it has great support in Weblate (see *GNU gettext* file format description).

注解: If you are about to use it in proprietary software, please consult licensing first, it might not be suitable for you.

GNU Gettext can be used from a variety of languages (C, Python, PHP, Ruby, JavaScript and many more) and usually the UI frameworks already come with some support for it. The standard usage is through the *gettext()* function call, which is often aliased to *_()* to make the code simpler and easier to read.

Additionally it provides *pgettext()* call to provide additional context to translators and *ngettext()* which can handle plural types as defined for target language.

As a widely spread tool, it has many wrappers which make its usage really simple, instead of manual invoking of Gettext described below, you might want to try one of them, for example intltool.

Sample program

The simple program in C using Gettext might look like following:

```
#include <libintl.h>
#include <locale.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int count = 1;
    setlocale(LC_ALL, "");
    bindtextdomain("hello", "/usr/share/locale");
    textdomain("hello");
    printf(
        ngettext(
            "Orangutan has %d banana.\n",
            "Orangutan has %d bananas.\n",
            count
        ),
        count
    );
    printf("%s\n", gettext("Thank you for using Weblate."));
    exit(0);
}
```

Extracting translatable strings

Once you have code using the gettext calls, you can use **xgettext** to extract messages from it and store them into a .pot:

```
$ xgettext main.c -o po/hello.pot
```

注解： There are alternative programs to extract strings from the code, for example pybabel.

This creates a template file, which you can use for starting new translations (using **msginit**) or updating existing ones after code change (you would use **msgmerge** for that). The resulting file is simply a structured text file:

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2015-10-23 11:02+0200\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=INTEGER; plural=EXPRESSION;\n"

#: main.c:14
#, c-format
msgid "Orangutan has %d banana.\n"
msgid_plural "Orangutan has %d bananas.\n"
msgstr[0] ""
msgstr[1] ""

#: main.c:20
msgid "Thank you for using Weblate."
msgstr ""
```

Each `msgid` line defines a string to translate, the special empty string in the beginning is the file header containing metadata about the translation.

Starting new translation

With the template in place, we can start our first translation:

```
$ msginit -i po/hello.pot -l cs --no-translator -o po/cs.po
Created cs.po.
```

The just created `cs.po` already has some information filled in. Most importantly it got the proper plural forms definition for chosen language and you can see number of plurals have changed according to that:

```
# Czech translations for PACKAGE package.
# Copyright (C) 2015 THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
```

(下页继续)

(续上页)

```
# Automatically generated, 2015.
#
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2015-10-23 11:02+0200\n"
"PO-Revision-Date: 2015-10-23 11:02+0200\n"
"Last-Translator: Automatically generated\n"
"Language-Team: none\n"
"Language: cs\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=ASCII\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=3; plural=(n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2;\n"

#: main.c:14
#, c-format
msgid "Orangutan has %d banana.\n"
msgid_plural "Orangutan has %d bananas.\n"
msgstr[0] ""
msgstr[1] ""
msgstr[2] ""

#: main.c:20
msgid "Thank you for using Weblate."
msgstr ""
```

This file is compiled into an optimized binary form, the .mo file used by the GNU Gettext functions at runtime.

Updating strings

Once you add more strings or change some strings in your program, you execute again **`xgettext`** which regenerates the template file:

```
$ xgettext main.c -o po/hello.pot
```

Then you can update individual translation files to match newly created templates (this includes reordering the strings to match new template):

```
$ msgmerge --previous --update po/cs.po po/hello.pot
```

Importing to Weblate

To import such translation into Weblate, all you need to define are the following fields when creating component (see *Component configuration* for detailed description of the fields):

Field	Value
源代码库	URL of the VCS repository with your project
文件掩码	`po/*.po`
新翻译的译文模版	`po/hello.pot`
文件格式	Choose *Gettext PO file*
新语言	Choose *Create new language file*

And that’s it, you’re now ready to start translating your software!

参见:

You can find a Gettext example with many languages in the Weblate Hello project on GitHub: <https://github.com/WeblateOrg/hello>.

1.7.3 Translating documentation using Sphinx

Sphinx is a tool for creating beautiful documentation. It uses simple reStructuredText syntax and can generate output in many formats. If you're looking for an example, this documentation is also built using it. The very useful companion for using Sphinx is the Read the Docs service, which will build and publish your documentation for free.

I will not focus on writing documentation itself, if you need guidance with that, just follow instructions on the Sphinx website. Once you have documentation ready, translating it is quite easy as Sphinx comes with support for this and it is quite nicely covered in their Internationalization. It's matter of few configuration directives and invoking of the `sphinx-intl` tool.

If you are using Read the Docs service, you can start building translated documentation on the Read the Docs. Their Localization of Documentation covers pretty much everything you need - creating another project, set its language and link it from main project as a translation.

Now all you need is translating the documentation content. Sphinx generates PO file for each directory or top level file, what can lead to quite a lot of files to translate (depending on `gettext_compact` settings). You can import the `index.po` into Weblate as an initial component and then configure 组件发现 addon to automatically discover all others.

表 1: Component configuration

组件名称	文档
文件掩码	`docs/locales/*/LC_MESSAGES/index.po`
新翻译的译文模版	`docs/locales/index.pot`
文件格式	*gettext PO* 文件
翻译标记	`rst-text`

表 2: 组件发现配置

用于匹配翻译文件的正则表达式	`docs/locales/(?P<language>[^/.]*)/LC_MESSAGES/(?P<component>[^/]*)\.po`
自定义组件名称	`Documentation: {{ component|title }}`
为新的翻译条目指定译文模版文件	`docs/locales/{{ component }}.pot`

提示: Would you prefer Sphinx to generate just single PO file? There is a hacky way to achieve this (used by Weblate documentation) by overriding Sphinx way to get a Gettext domain of a document. Place following snippet to your Sphinx configuration in `conf.py`:

```
import sphinx.transforms.i18n
import sphinx.util.i18n

# Hacky way to have all localized content in single domain
sphinx.transforms.i18n.docname_to_domain = (
    sphinx.util.i18n.docname_to_domain
) = lambda docname, compact: "docs"
```

This might be directly supported by Sphinx in future releases, see <https://github.com/sphinx-doc/sphinx/issues/784>.

参见:

The Odorik python module documentation is built using Sphinx, Read the Docs and translated using Weblate.

1.7.4 Translating HTML and JavaScript using Weblate CDN

Starting with Weblate 4.2 it is posible to export localization to a CDN using *JavaScript 本地化 CDN* addon.

注解： 此功能在托管的 Weblate 上配置。它需要在安装时进行额外的配置，见*LOCALIZE_CDN_URL* 和*LOCALIZE_CDN_PATH*。

Upon installation into your component it will push committed translations (see 惰性提交) to the CDN and these can be used in your web pages to localize them.

创建组件中

First, you need to create a monolingual component which will hold your strings, see 添加翻译项目和组件 for generic instructions on that.

In case you have existing repository to start with (for example the one containing HTML files), create an empty JSON file in the repository for the source language (see 源语言), for example `locales/en.json`. The content should be `{}` to indicate an empty object. Once you have that, the repository can be imported into Weblate and you can start with an addon configuration.

提示： In case you have existing translations, you can place them into the language JSON files and those will be used in Weblate.

For those who do not want to use existing repository (or do not have one), choose *Start from scratch* when creating component and choose *JSON file* as a file format (it is okay to choose any monolingual format at this point).

正在配置 Weblate 内容分发网络附加组件

The *JavaScript 本地化 CDN* addon provides few configuration options.

翻译阈值 Translations translated above this threshold will be included in the CDN.

CSS 选择器 Configures which strings from the HTML documents are translatable, see *Weblate 内容分发网络的字符串提取* and 使用 *Weblate* 内容分发网络对 *HTML* 进行本地化操作.

语言 cookie 名称 Name of cookie which contains user selected language. Used in the JavaScript snippet for 使用 *Weblate* 内容分发网络对 *HTML* 进行本地化操作.

从 HTML 文件里提取字符串 List of files in the repository or URLs where Weblate will look for translatable strings and offer them for a translation, see *Weblate 内容分发网络的字符串提取*.

Weblate 内容分发网络的字符串提取

翻译字符串必须在 Weblate 中出现。可以手动管理，使用 API 来建立，或使用 *Extract strings from HTML files* 列出文件或 URL ， Weblate 会自动提取它们。文件必须出现在残酷中，或者包含远程 URL ，由 Weblate 下载并规则地分析。

CSS selector 的默认配置提取 CSS class `l10n` 的元素，例如从后面的一段中提取两个字符串：

```
<section class="content">
    <div class="row">
        <div class="wrap">
            <h1 class="section-title min-m l10n">Maintenance in progress</h1>
            <div class="page-desc">
                <p class="l10n">We're sorry, but this site is currently down for␣
↪maintenance.</p>
            </div>
```

(下页继续)

(续上页)

```
        </div>
    </div>
</section>
```

在不想修改现有代码的情况下，还可以使用 * 作为选择器处理所有元素。

注解： 目前只有元素的文本被提取。这个附加组件不支持元素属性或具有子类的元素的本地化。

使用 Weblate 内容分发网络对 HTML 进行本地化操作

为了将 HTML 文件本地化，需要导入 `weblate.js` 脚本：

```
<script src="https://weblate-cdn.com/a5ba5dc29f39498aa734528a54b50d0a/weblate.js"␣
↪async></script>
```

导入时，会自动发现所有匹配的翻译元素（基于 *CSS selector* ），并用翻译替代其文本。

从配置的 cookie 检测到用户语言，并退回到在浏览器中配置的用户首选语言。

The *Language cookie name* can be useful for integration with other applications (for example choose `django_language` when using Django).

JavaScript 本地化

单独的翻译在内容分发网络下暴露为双语 JSON 文件。要获取一个，你可以使用以下代码：

```
fetch(("https://weblate-cdn.com/a5ba5dc29f39498aa734528a54b50d0a/cs.json")
  .then(response => response.json())
  .then(data => console.log(data));
```

在这种情况下需要采用实际的本地化逻辑。

1.7.5 Translation component alerts

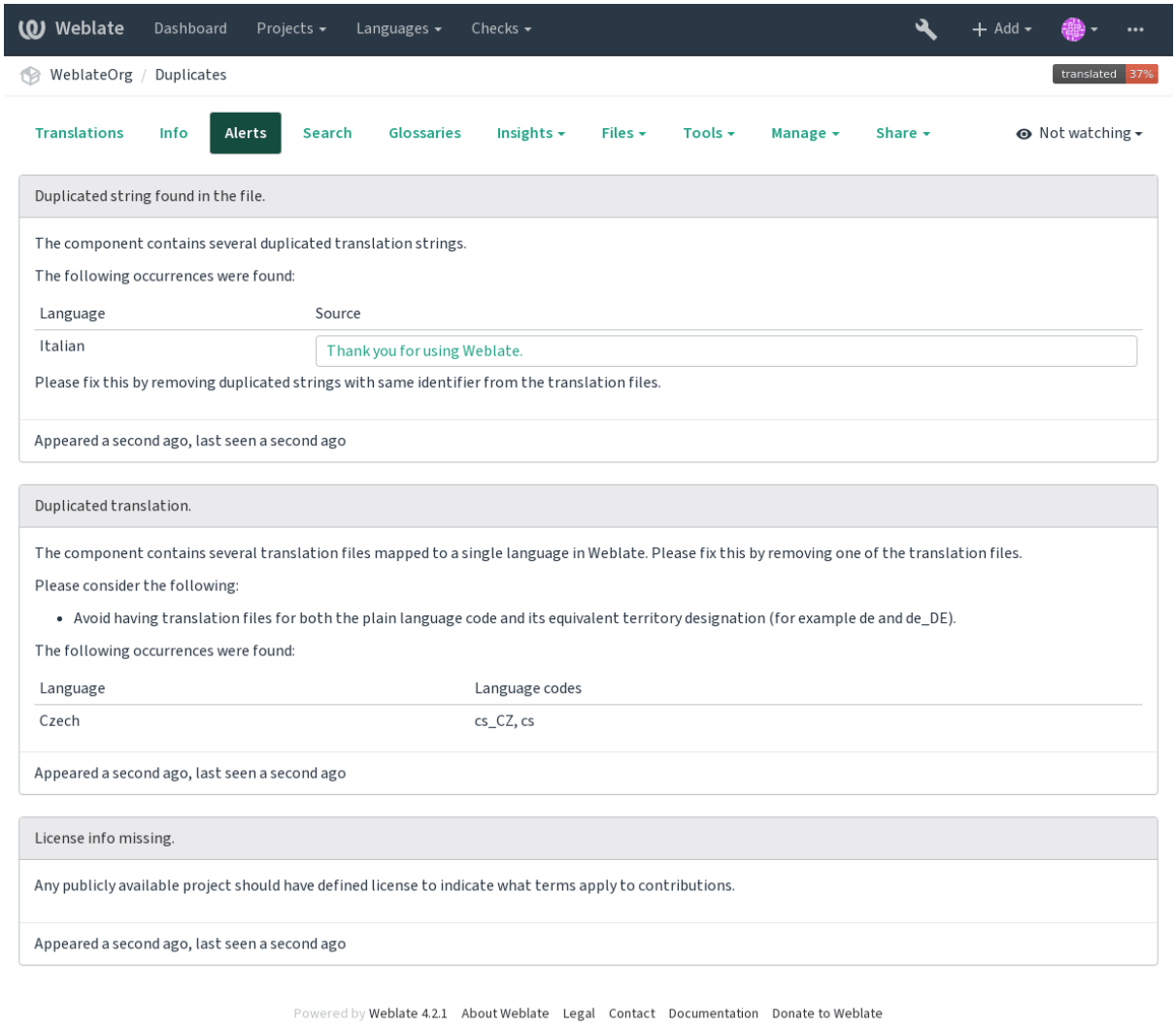
Shows errors in the Weblate configuration or the translation project for any given translation component. Guidance on how to address found issues is also offered.

Currently the following is covered:

- Duplicated source strings in translation files
- Duplicated languages within translations
- Merge or update failures in the source repository
- Unused new base in component settings
- Parse errors in the translation files
- Duplicate filemask used for linked components
- Broken URLs

Alerts are listed on each respective component page as *Alerts*. If it is missing, the component clears all current checks. Alerts can not be ignored, but will disappear once the underlying problem has been fixed.

A component with both duplicated strings and languages looks like this:



参见:

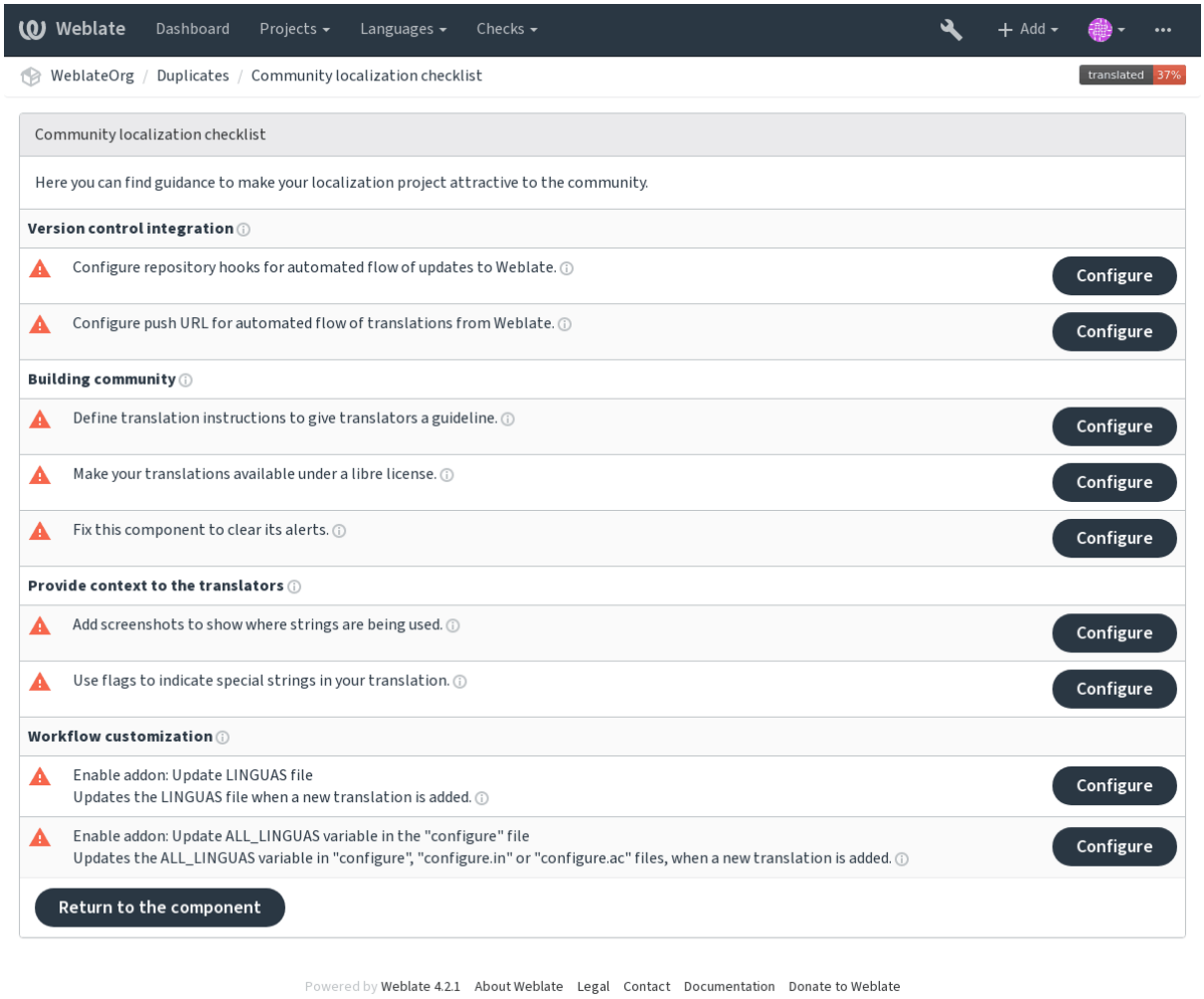
Using custom certificate authority

1.7.6 Building translators community

社区本地化检查清单

3.9 新版功能.

The *Community localization checklist* which can be found in the *Menu* menu of each component can give you guidance to make your localization process easy for community translators.



1.7.7 Managing translations

Adding new translations

New strings can be made available for translation when they appear in the base file, called *Template for new translations* (see *Component configuration*). If your file format doesn't require such a file, as is the case with most monolingual translation flows, you can start with blank files).

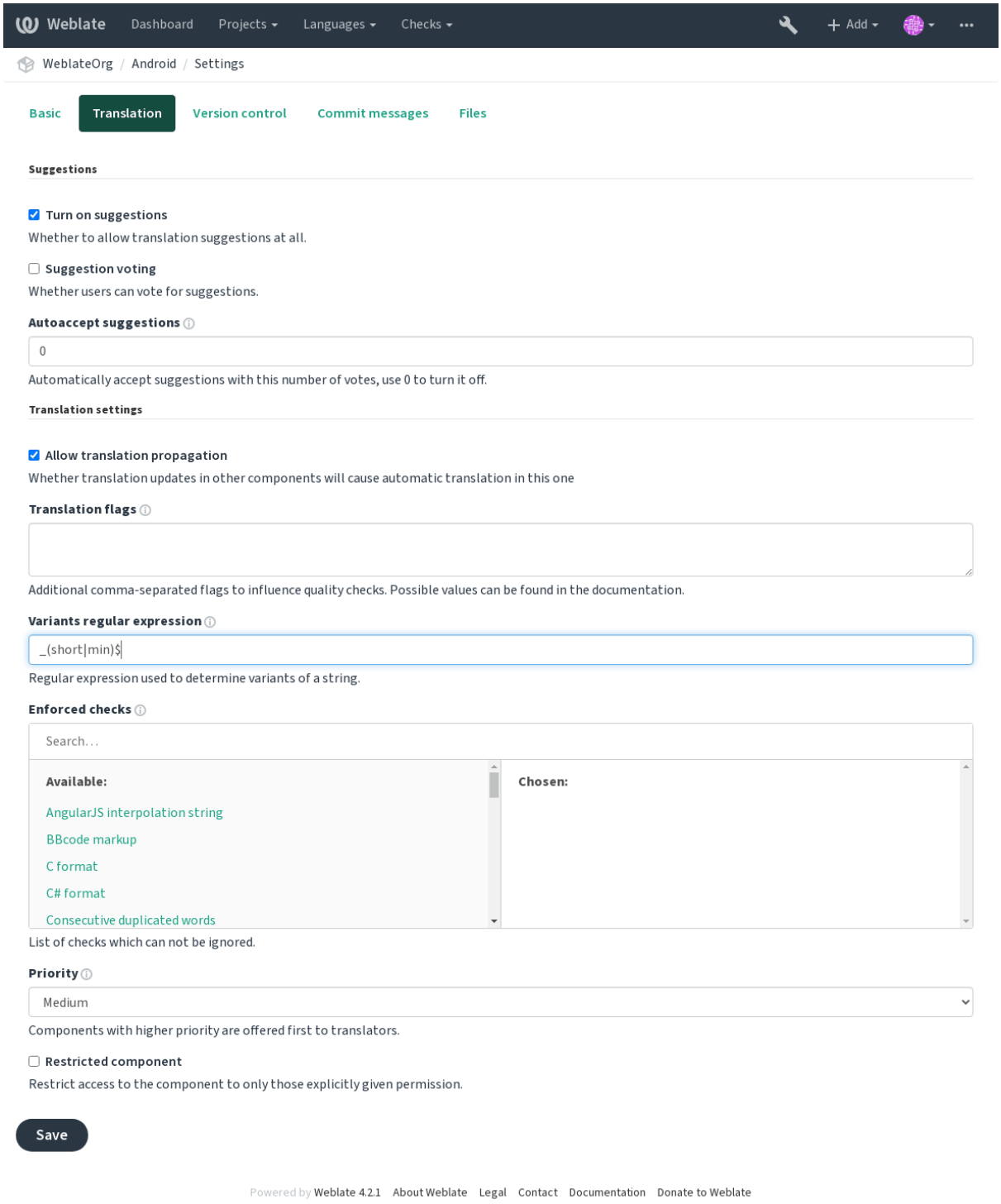
New languages can be added right away when requested by a user in Weblate, or a notification will be sent to project admins for approval and manual addition. This can be done using *Start new translation* in *Component configuration*.

注解： Project admins can always start translation within Weblate directly.

Language files added manually to the VCS are added to the component when Weblate updates the repository. About repository update settings, see 更新仓库).

String variants

Variants are useful to group several strings together so that translators can see all variants of the string at one place. You can define regular expression to group the strings in the *Component configuration*:

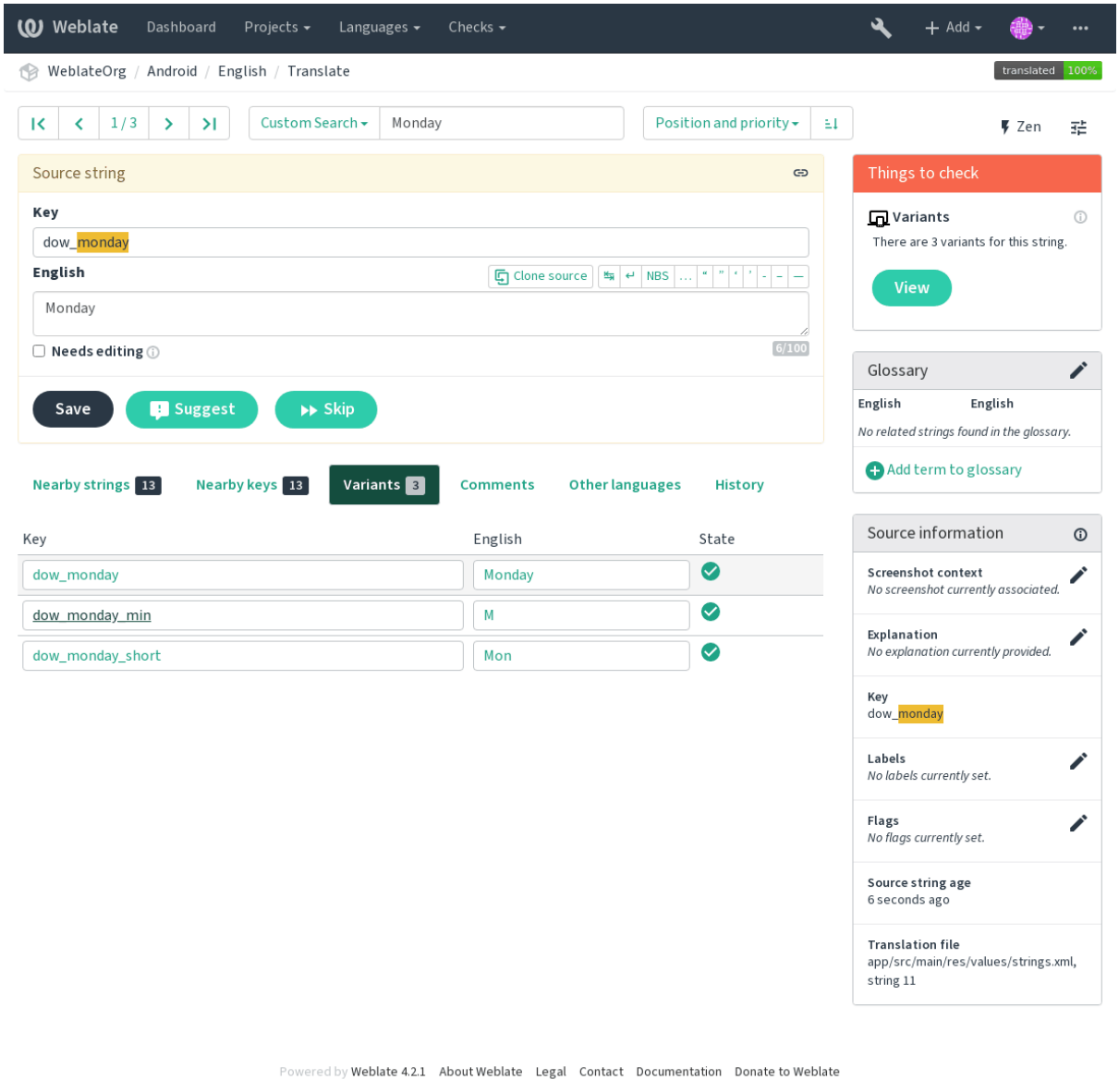


The expression is matched against *Key* to generate root key of the variant. All matching strings are then part of single variants group, including the translation exactly matching the root key, even if that is not matched by the regular expression.

The following table lists some usage examples:

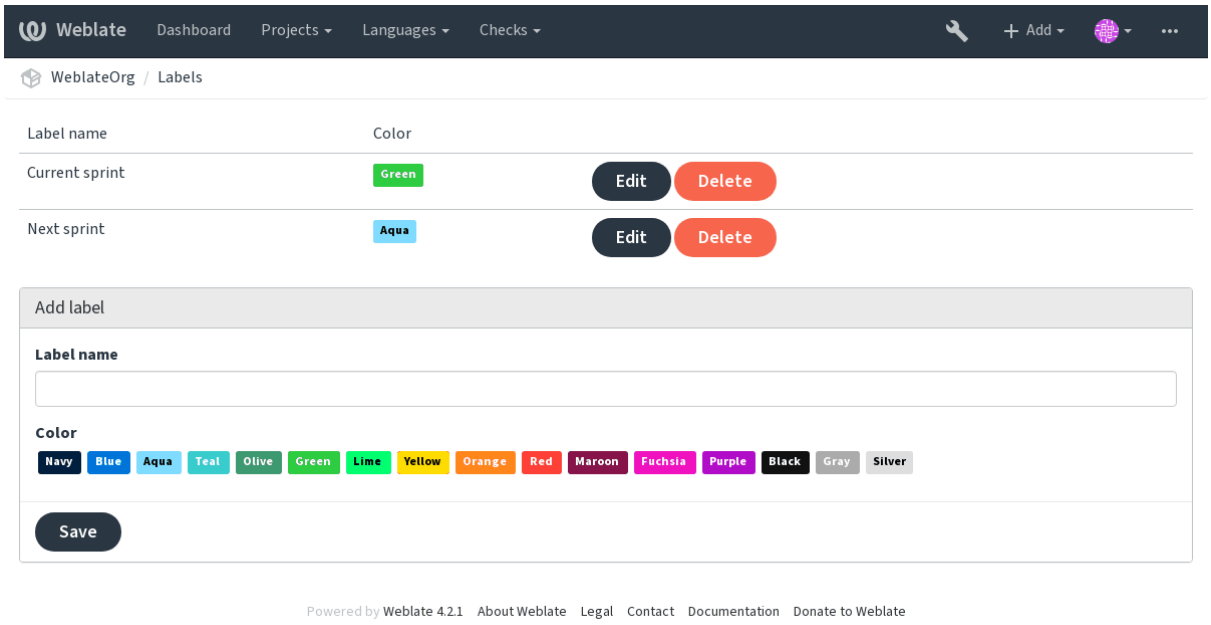
Use case	Regular expression variant	Matched translation keys
Suffix identification	`(Short|Min)$`	`monthShort`, `monthMin`, `month`
Inline identification	`#[SML]`	`dial#S.key`, `dial#M.key`, `dial.key`

The variant is later grouped when translating:



String labels

Split component translation strings into categories by text and colour in the project configuration.



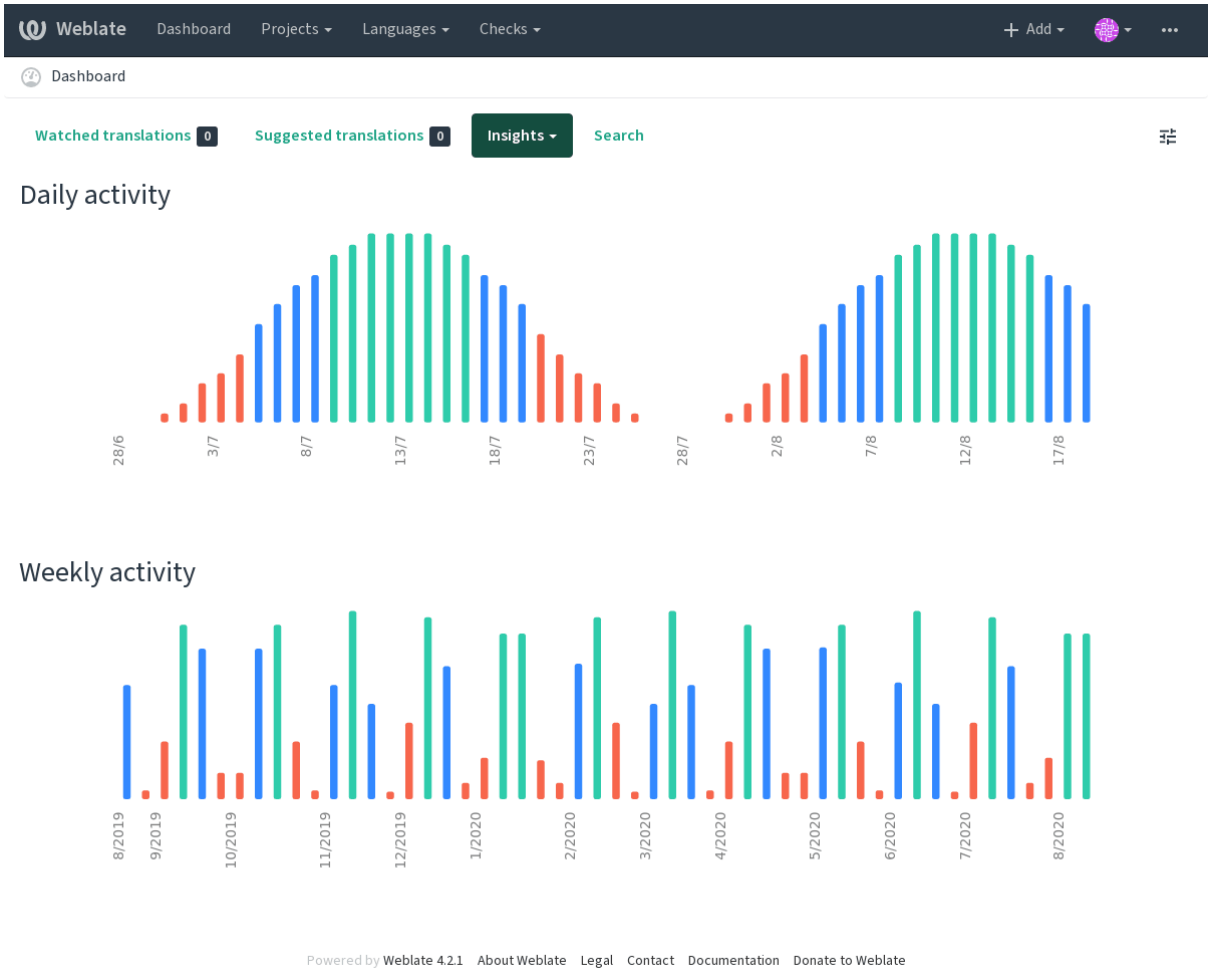
提示： Labels can be assigned to units in *Additional info on source strings* by bulk editing, or using the 批量编辑 addon.

1.7.8 Reviewing strings

Activity reports

Activity reports check changes of translations, for projects, components or individual users.

The activity reports for a project or component is accessible from its dashboard, on the *Insights* tab, selecting *Activity*.



More reports are accessible on the *Insights* tab, selecting *Translation reports*.

The activity of the currently signed in user can be seen by clicking on *Profile* from the user menu on the top right.

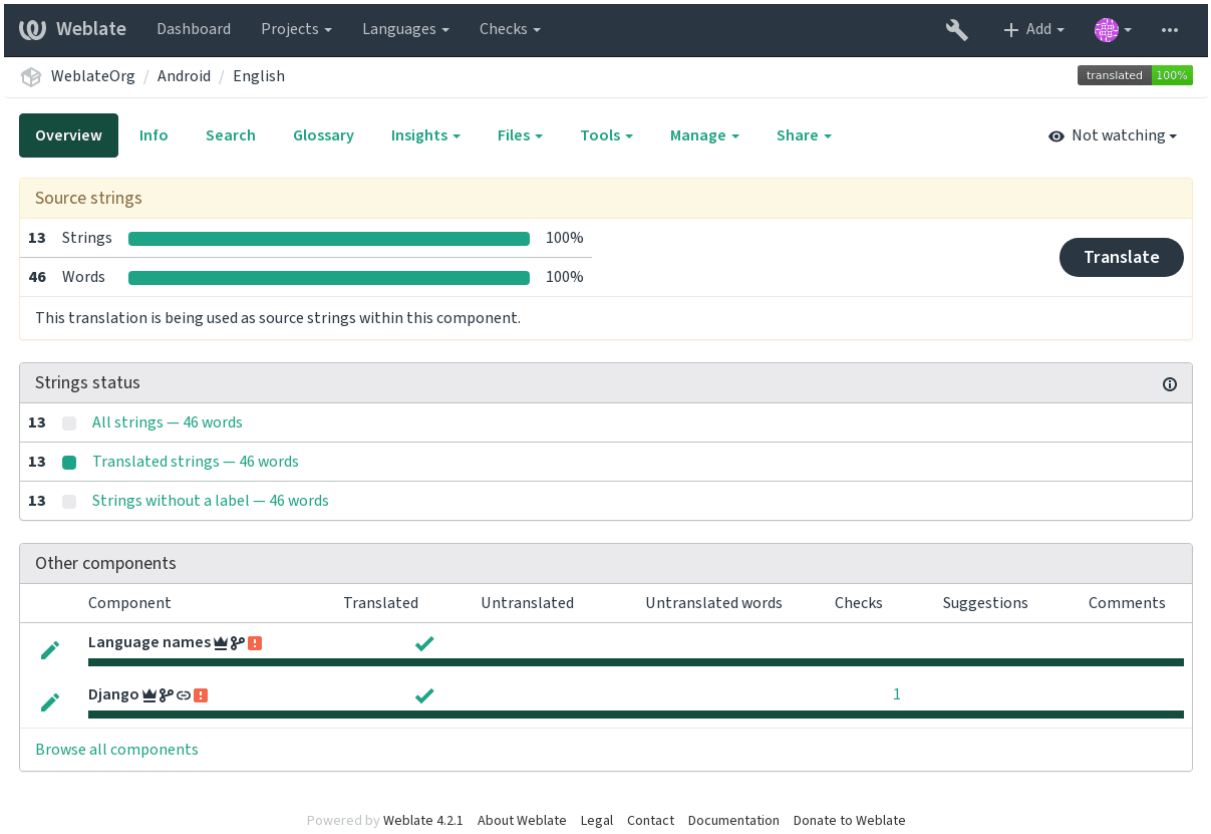
Source strings checks

There are many 质量检查, some of them focus on improving the quality of source strings. Many failing checks suggest a hint to make source strings easier to translate. All types of failing source checks are displayed on the *Source* tab of every component.

Translation string checks

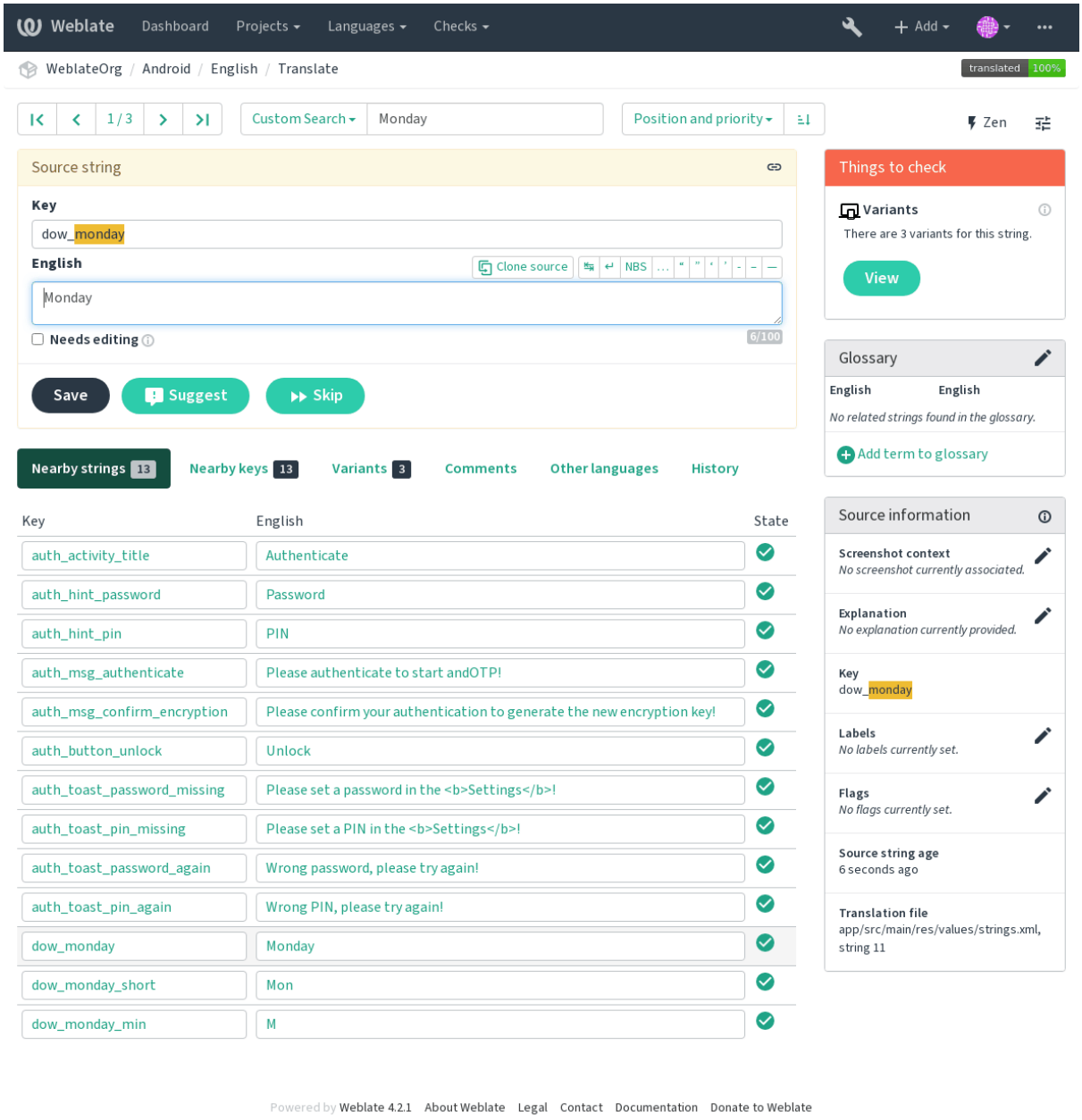
Erroneous failing translation string checks indicate the problem is with the source string. Translators sometimes fix mistakes in the translation instead of reporting it - a typical example is a missing full stop at the end of a sentence.

Reviewing all failing checks can provide valuable feedback to improve its source strings. To make source strings review easier, Weblate automatically creates a translation for the source language and shows you source level checks there:



One of the most interesting checks here is the 多项检查失败 - it is triggered whenever there is failure on multiple translations of a given string. Usually this is something to look for, as this is a string which translators have problems translating properly.

The detailed listing is a per language overview:

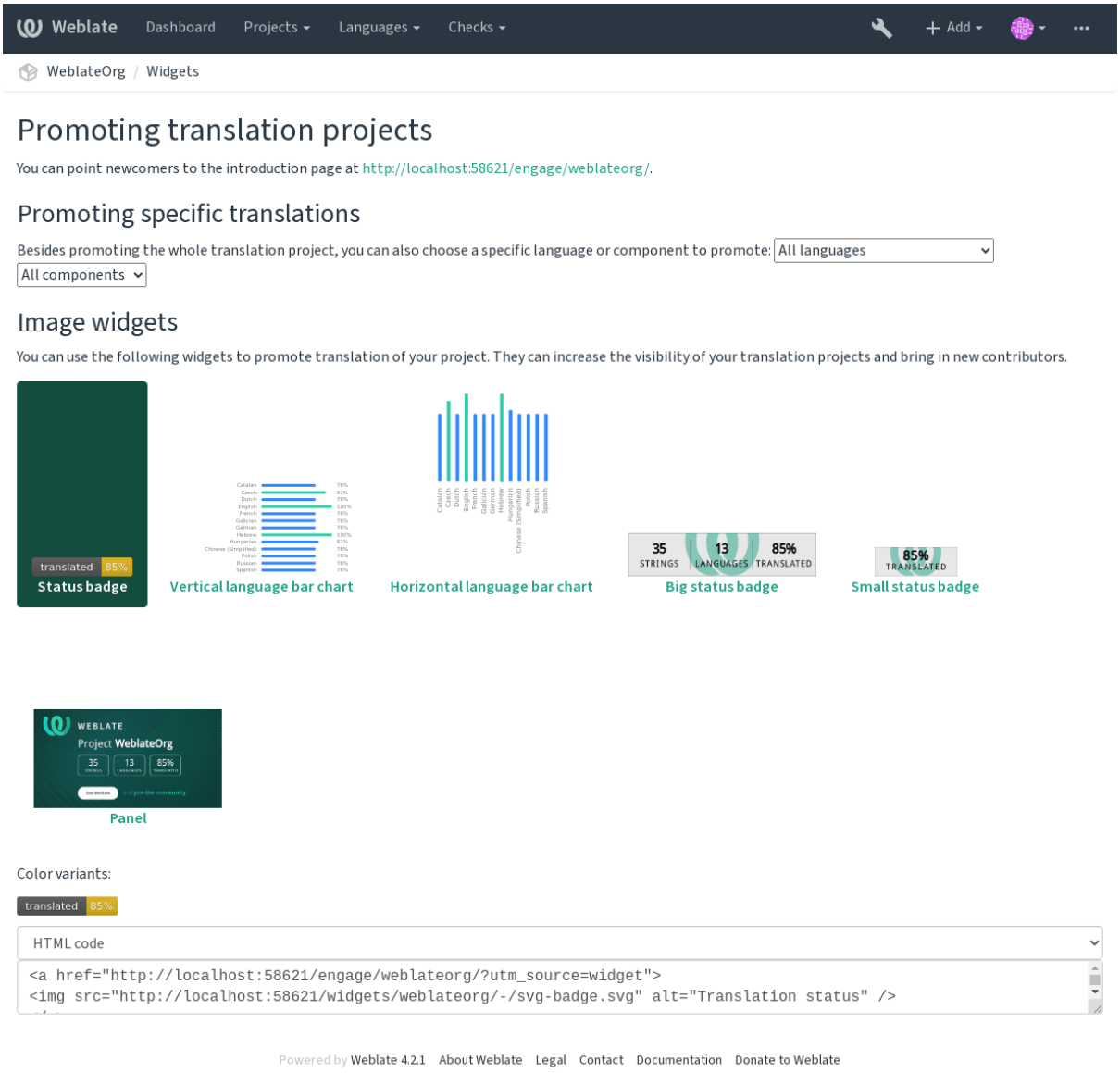


String comments

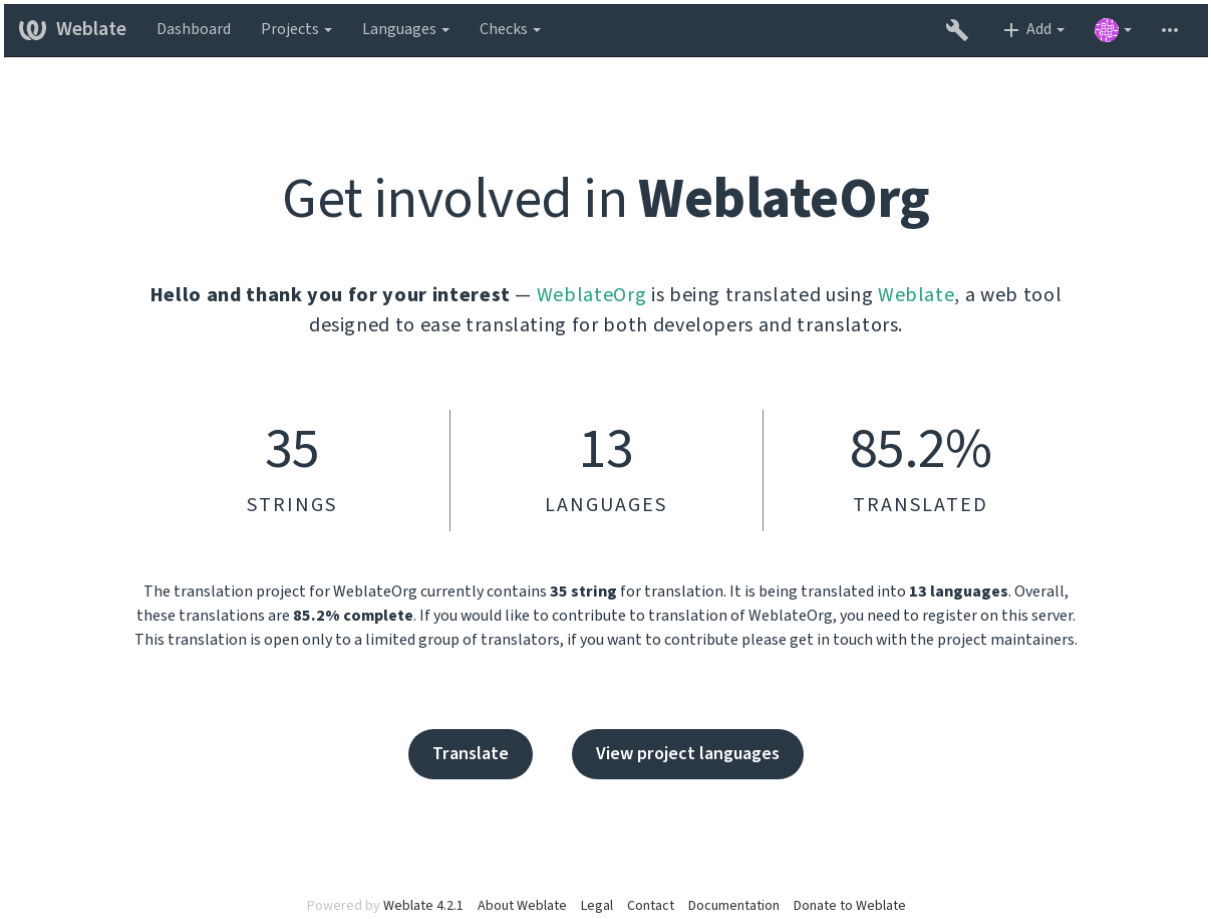
Translators can comment on both translation and source strings. Each *Component configuration* can be configured to receive such comments to an e-mail address, and using the developers mailing list is usually the best approach. This way you can keep an eye on when problems arise in translation, take care of them, and fix them quickly.

1.7.9 Promoting the translation

Weblate provides you widgets to share on your website or other sources to promote the translation project. It also has a nice welcome page for new contributors to give them basic information about the translation. Additionally you can share information about translation using Facebook or Twitter. All these possibilities can be found on the *Share* tab:

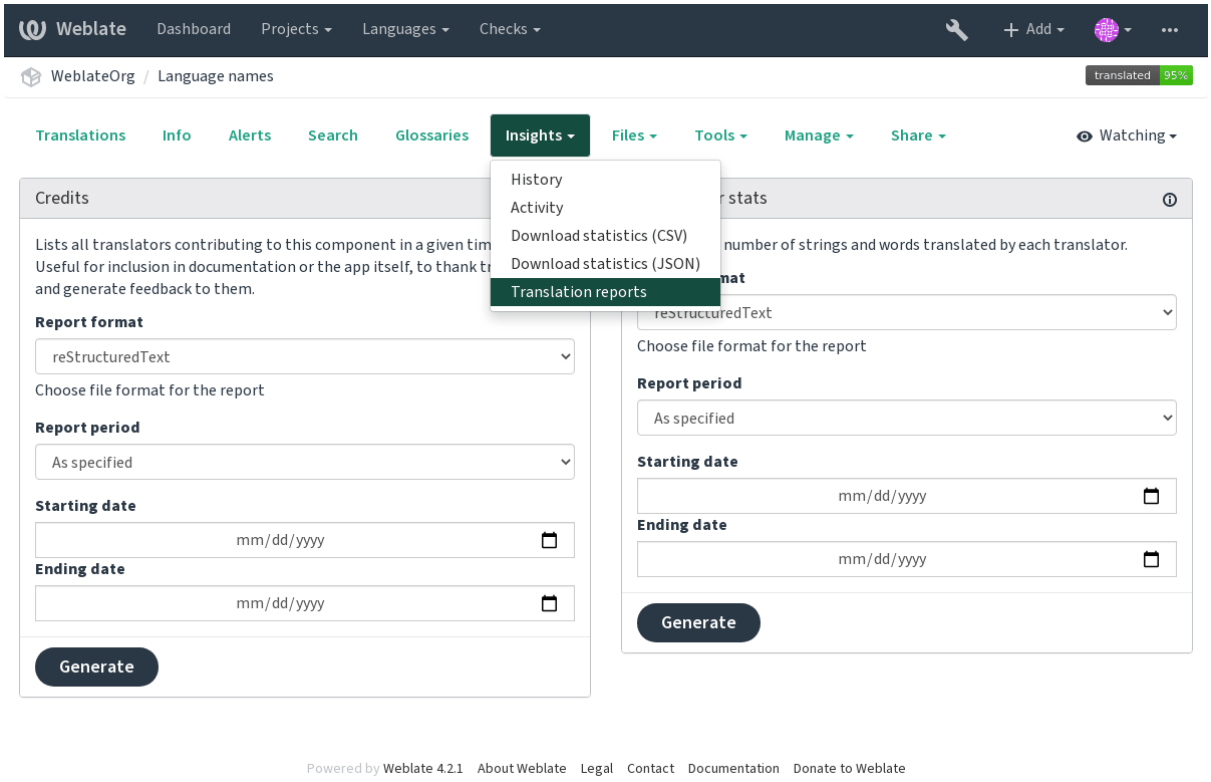


All these badges are provided with the link to simple page which explains users how to translate using Weblate:



1.7.10 Translation progress reporting

Reporting features give insight into how a translation progresses over a given period. A summary of contributions to any given component over time is provided. The reporting tool is found in the *Insights* menu of any translation component, project or on the dashboard:



Several reporting tools are available on this page and all can produce output in HTML, reStructuredText or JSON. The first two formats are suitable for embedding statistics into existing documentation, while JSON is useful for further processing of the data.

Translator credits

Generates a document usable for crediting translators - sorted by language and lists all contributors to a given language:

```
* Czech

    * Michal Čihař <michal@cihar.com> (10)
    * John Doe <john@example.com> (5)

* Dutch

    * Jane Doe <jane@example.com> (42)
```

It will render as:

- 捷克语（čeština）
  - Michal Čihař <michal@cihar.com> (10)
  - John Doe <john@example.com> (5)
- 荷兰语（Nederlands）
  - Jae Doe <jane@example.com> (42)

提示： The number in parenthesis indicates number of contributions in given period.

贡献者统计

Generates the number of translated words and strings by translator name:

```
======================================== ========================================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ========================
Name                                    Email
 ↪Count total              Source words total       Source chars total
 ↪Target words total       Target chars total       Count new
 ↪Source words new         Source chars new         Target words new
 ↪Target chars new         Count approved           Source words approved
 ↪Source chars approved    Target words approved    Target chars approved    Count
 ↪edited             Source words edited      Source chars edited      Target
 ↪words edited      Target chars edited
======================================== ========================================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ========================
Michal Čihař                            michal@cihar.com
 ↪                      1                        3                      24
 ↪                  3                       21                      1
 ↪          3                       24                      3
 ↪    21                      0                       0                      0
 ↪                        0                       0                      0
 ↪                    0                       0                      0
 ↪            0
Allan Nordhøy                           allan@example.com
 ↪                      2                        5                      25
 ↪                  4                       28                      2
 ↪          3                       24                      3
 ↪    21                      0                       0                      0
 ↪                        0                       0                      0
 ↪                    0                       0                      0
 ↪            0
======================================== ========================================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ======================== ========================
 ↪======================== ========================
```

And it will get rendered as:

名称	Email	Count-total	Source-words-total	Source-chars-total	Target-words-total	Target-chars-total	Count-new	Source-words-new	Source-chars-new	Target-words-new	Target-chars-new	Count-approved	Source-words-approved	Source-chars-approved	Target-words-approved	Target-chars-approved	Count-edited	Source-words-edited	Source-chars-edited	Target-words-edited	Target-chars-edited
Michal Čihař	michal@cihar.com	1	3	24	3	21	1	3	24	3	21	0	0	0	0	0	0	0	0	0	0
Allan Nordhøy	allan@example.com	2	5	25	4	28	2	3	24	3	21	0	0	0	0	0	0	0	0	0	0

It can be useful if you pay your translators based on amount of work, it gives you various stats on translators work.

All stats are available in three variants:

Total Overall number of edited strings.

New Newly translated strings which didn’t have translation before.

Approved Count for string approvals in review workflow (see 专门的审核者).

Edited Edited strings which had translation before.

The following metrics are available for each:

Count Number of strings.

Edits Number of edits in the string, measured in Damerau–Levenshtein distance.

Source words Number of words in the source string.

Source characters Number of characters in the source string.

Target words Number of words in the translated string.

Target characters Number of characters in the translated string.

1.8 翻译工作流

支持多种翻译工作流。

以下不是配置 Weblate 的方法的完整列表。您可以将其他工作流基于此处列出的最常见的示例。

1.8.1 翻译访问

访问控制 在工作流中没有太多讨论，因为每个访问控制选项都可以应用于任何工作流。请查阅该文档以获取有关如何管理对翻译的访问的信息。

在以下各章中，*任何用户*都是指有权访问翻译的用户。如果项目是公共项目，则可以是任何经过身份验证的用户，也可以是具有项目 *Translate* 权限的用户。

1.8.2 翻译状态

每个翻译的字符串可以处于以下状态之一：

未翻译 翻译是空的，取决于文件格式，翻译是否可能存储在文件中。

需要编辑 翻译需要编辑，这通常是源字符串更改的结果。转换文件存储在文件中，具体取决于文件格式，它可能被标记为需要编辑（例如，当它获得 fuzzy 标志时）。

等待复查 翻译已完成，但未进行审核。它作为有效翻译存储在文件中。

已批准 翻译已在审核中得到批准。翻译者不能再更改它，只能由审阅者更改。译者只能向其中添加建议。

建议 建议仅存储在 Weblate 中，而不存储在翻译文件中。

1.8.3 直接翻译

这是小型团队最常用的设置，任何人都可以直接翻译。这也是 Weblate 中的默认设置。

- 任何用户都可以编辑翻译。
- 当翻译者不确定更改时，建议是建议更改的可选方法。

设置	Value	备注
启用复查	已关闭	在项目级别配置。
启用建议	已开启	用户可以在不确定时提出建议，这很有用。
建议投票	已关闭	
自动接受建议	0	
翻译组	用户	或使用访问控制 进行 翻译。
审核者组	不可用	不曾用过。

1.8.4 同行评审

使用此工作流程，任何人都可以添加建议，并且需要其他成员的同意才能被接受为翻译。

- 任何用户都可以添加建议。
- 任何用户都可以对建议投票。
- 当给定预定数量的投票时，建议就变成翻译。

设置	Value	备注
启用复查	已关闭	在项目级别配置。
启用建议	已开启	
建议投票	已关闭	
自动接受建议	1	您可以设置更高的值，以要求更多的同行评审。
翻译组	用户	或使用访问控制 进行 翻译。
审核者组	不可用	未使用，所有翻译员都审阅。

1.8.5 专门的审核者

2.18 新版功能: 从 Weblate 2.18 开始，支持正确的审核工作流。

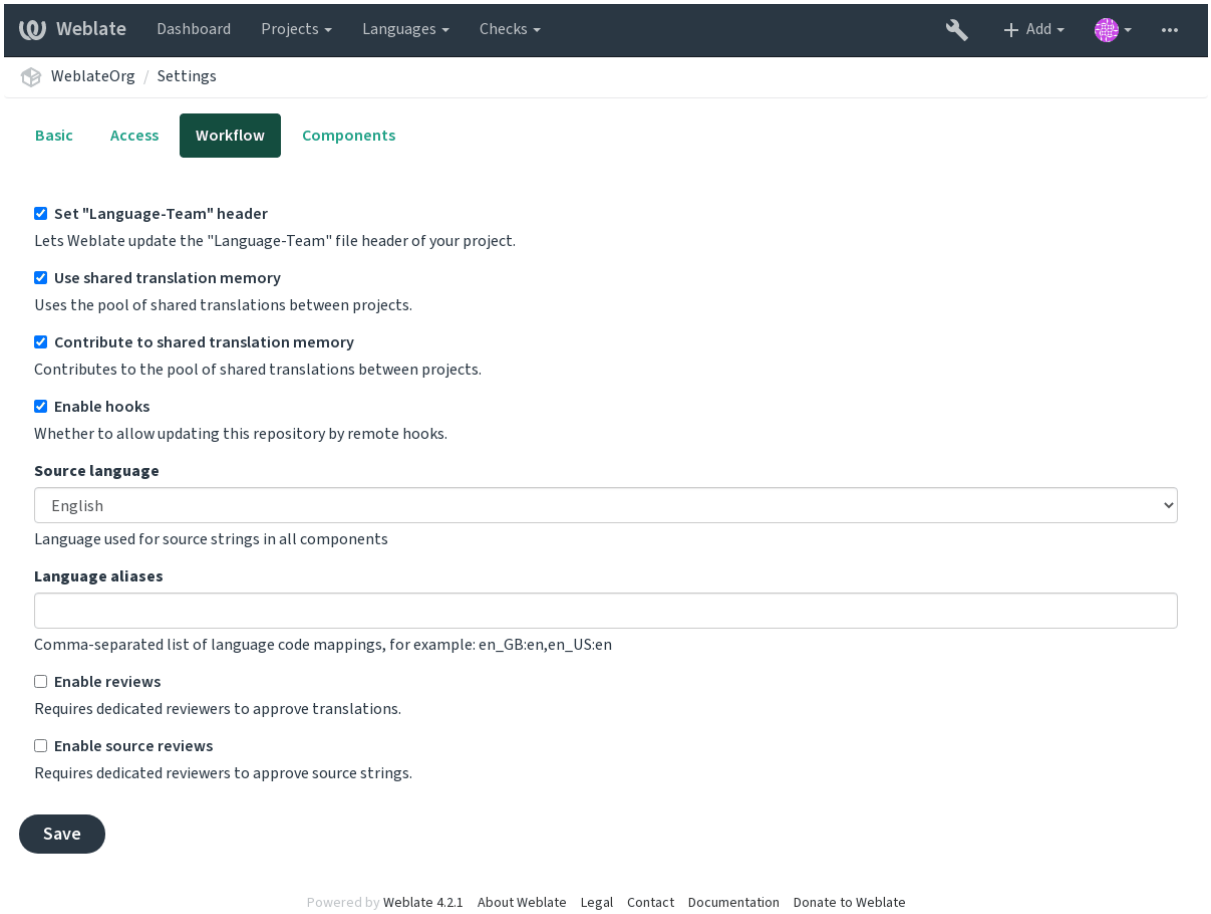
使用专门的审核者，您有两组用户，一组可以提交翻译，而另一组可以审核它们以确保翻译一致且质量良好。

- 任何用户都可以编辑未批准的翻译。
- *Reviewer* 可以批准/否决字符串。
- 审核者可以编辑所有翻译（包括批准的翻译）。
- 建议还可以用于建议更改已批准的字符串。

设置	Value	备注
启用复查	已开启	在项目级别配置。
启用建议	已关闭	用户可以在不确定时提出建议，这很有用。
建议投票	已关闭	
自动接受建议	0	
翻译组	用户	或使用访问控制 进行 翻译。
审核者组	校对	或使用访问控制 进行 审核。

1.8.6 打开审核

可以在项目设置的 *Workflow* 子页面中的项目配置中打开审核（位于 *Manage* → *Settings* 菜单中）:



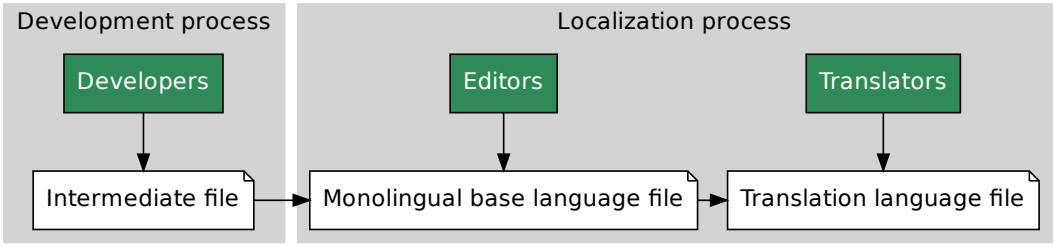
注解: 根据 Weblate 配置的不同，该设置可能对您不可用。例如，在 Hosted Weblate 上，这不适用于免费

托管的项目。

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 中间语言文件, 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.



参见:

中间语言文件, 单语种译文模版语言文件, 双语和单语格式

1.8.8 源字符串复查

通过允许启用来源评论，复查过程可以应用到源字符串上。一旦允许，用户可以汇报源字符串种的情况。实际过程依赖于使用双语种还是单语种格式。

对于单语种格式，源字符串复查的行为与专门的审核者 相似——一旦源字符串汇报了情况，就会被标记为 *Needs editing* 。

双语种格式不允许直接编辑源字符串（他们典型地是从源代码种直接提取的）。在这种情况下，*Source needs review* 标签会贴到翻译者回到的字符串上。可以复查这样的字符串，并在源中编辑或删除标签。

参见:

双语和单语格式, 专门的审核者, *String labels*

1.9 Frequently Asked Questions

1.9.1 配置

如何创建自动化工作流？

Weblate 可以为您半自动处理所有翻译工作。如果授予它对仓库的推送访问权限，则翻译可以在没有交互的情况下进行，除非发生某些合并冲突。

1. 设置您的 Git 仓库以告知 Weblate 何时有任何更改，请参阅通知钩子 以获取有关如何执行此操作的信息。

2. 在 Weblate 中的*Component configuration* 配置中设置推送 URL，这使 Weblate 可以将更改推送到仓库。
3. 在 Weblate 中打开项目配置 配置上的 push-on-commit，这将使 Weblate 在 Weblate 发生更改时将更改推送到仓库。

参见:

持续本地化集成, 避免合并冲突

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 ran into a merge conflict, the easiest way is to solve all conflicts locally at your workstation - is to simply 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.

注解: 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/master

# 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/
```

(下页继续)

(续上页)

```
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 master branch:
git checkout master
git merge weblates-second/master
... # 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/

# Update Weblate remote:
git remote update weblate

# Merge Weblate changes:
git merge weblate/master

# 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
```

参见:

How to export the Git repository that Weblate uses?, 持续本地化集成, 避免合并冲突

How do I translate several branches at once?

Weblate supports pushing translation changes within one 项目配置. 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
```

参见:

在部件之间保持翻译一致

How to translate multi-platform projects?

Weblate supports a wide range of file formats (see 支持的文件格式) 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 添加翻译项目和组件), 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.

参见:

在部件之间保持翻译一致

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 如何创建自动化工作流？).
- 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.

参见:

Component configuration

1.9.2 Usage

How do I review the translations of others?

- You can subscribe to any changes made in 通知 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.

How do I provide feedback on a source string?

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

How can I use existing translations while translating?

- 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.

参见:

机器翻译, 机器翻译

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.

For monolingual files (see 支持的文件格式) Weblate might add new translation strings not present in the *template*, and not in actual translations. It does not however perform any automatic cleanup of stale strings as that might have unexpected outcomes. If you want to do this, please install a pre-commit hook which will handle the cleanup according to your requirements.

Weblate also will not try to update bilingual files in any way, so if you need `po` files being updated from `pot`, you need to do it yourself.

参见:

用脚本处理仓库

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.

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.

注解: 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 *更新 PO 文件以匹配 POT 文件 (msgmerge)*.

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
```

参见:

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']
```

参见:

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 源语言 in the project settings.
- 如果不需要源语言的翻译文件，请从存储库中将其删除。
- 如果需要源语言的翻译文件，但 Weblate 应该忽略它，请调整:ref: *component-language_regex* 来排除它。

1.9.4 功能

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 ref:*vcs-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*.

注解: 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.

参见:

版本控制集成

Weblate 如何记录翻译者？

Weblate 中所做的每个更改都将以翻译者的名称提交到 VCS 中。这样，每个更改都具有适当的作者身份，您可以使用用于代码的标准 VCS 工具来进行跟踪。

此外，如果翻译文件格式支持，则文件头会更新为包含翻译者的名称。

参见:

list_translators, *Translation progress 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. If you feel your project should be translated as one, consider merging these po files. It will make life easier even for translators not using Weblate.

注解： In case there is great demand for this feature, it might be implemented in future versions.

Why does Weblate use language codes such sr_Latn or zh_Hant?

These are language codes defined by **RFC 4646** 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 `sr_Hans`.

1.10 支持的文件格式

Weblate 支持 translate-toolkit 理解的大多数翻译格式，但是每种格式都略有不同，可能会出现未经良好测试的格式问题。

参见：

Translation Related File Formats

注解： 为您的应用程序选择文件格式时，最好在您使用的工具箱/平台中保留一些公认的格式。这样，您的翻译人员可以额外使用他们习惯使用的任何工具，并且更有可能为您的项目做出贡献。

1.10.1 双语和单语格式

支持 monolingual 和 bilingual 格式。双语格式在单个文件中存储两种语言——源和翻译（典型示例是*GNU gettext*，*XLIFF* 或*Apple iOS strings* 字符串）。另一方面，单语格式通过 ID 识别字符串，每个语言文件仅包含那些语言到任何给定语言（通常是*Android string resources*）的映射。两种变体都使用某些文件格式，请参见下面的详细说明。

为了正确使用单语文件，Weblate 要求访问一个包含完整字符串列表的文件，以与其源一起翻译——该文件在 Weblate 中称为单语种译文模版语言文件，尽管命名方式可能会有所不同。

另外，可以利用中间语言文件 扩展此工作流程，以包括开发人员提供的字符串，但不要在最终的字符串中使用。

1.10.2 自动检测

Weblate 可以自动检测几种常用的文件格式，但是这种检测会损害您的性能，并且会限制特定于给定文件格式的功能（例如，自动添加新翻译）。

1.10.3 翻译类型功能

所有受支持格式的功能：

格式	语言能力[1]	复数[2]	注释[3]	语境[4]	位置[5]	标志[8]	附加状态[6]
GNU gettext	双语	yes	yes	yes	yes	yes[9]	needs editing
单语 gettext	mono	yes	yes	yes	yes	yes[9]	needs editing
XLIFF	both	yes	yes	yes	yes	yes[10]	needs editing, approved
Java properties	both	no	yes	no	no	no	
GWT properties	mono	yes	yes	no	no	no	
Joomla translations	mono	no	yes	no	yes	no	
Qt Linguist .ts	both	yes	yes	no	yes	yes[10]	needs editing
Android string resources	mono	yes	yes[7]	no	no	yes[10]	
Apple iOS strings	双语	no	yes	no	no	no	
PHP 字符串	mono	no[11]	yes	no	no	no	
JSON files	mono	no	no	no	no	no	
JSON i18next files	mono	yes	no	no	no	no	
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[10]	
CSV 文件	mono	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	mono	no	no	no	no	yes[10]	
Windows RC files	mono	no	yes	no	no	no	
Excel Open XML	mono	no	yes	yes	yes	no	needs editing
应用商店元数据文件	mono	no	no	no	no	no	
Subtitle files	mono	no	no	no	yes	no	

下页继续

表 3 – 续上页

格式	语言能力[1]	复数[2]	注释[3]	语境[4]	位置[5]	标志[8]	附加状态[6]
HTML files	mono	no	no	no	no	no	
OpenDocumen	mono	no	no	no	no	no	
Format							
IDML Format	mono	no	no	no	no	no	
INI translations	mono	no	no	no	no	no	
Inno Setup INI 翻译	mono	no	no	no	no	no	

1.10.4 GNU gettext

翻译自由软件的最广泛使用的格式。这是 Weblate 支持的第一种格式，至今仍获得最好的支持。

通过调整文件头或链接到相应的源文件，可以支持存储在文件中的上下文信息。

双语 gettext PO 文件通常如下所示：

```
#: 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"
```

典型的 Weblate *Component configuration*	
文件掩码	`po/*.po`
单语种译文模版语言文件	*Empty*
新翻译的译文模版	`po/messages.pot`
文件格式	*Gettext PO file*

参见：

Translating software using GNU Gettext, *Translating documentation using Sphinx*, Gettext on Wikipedia, PO Files, *更新“配置文件”中的 ALL_LINGUAS 变量*, *自定义 gettext 输出*, *更新 LINGUAS 文件*, *生成 MO 文件*, *更新 PO 文件以匹配 POT 文件 (msgmerge)*

[1] See 双语和单语格式

[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 定制行为

[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.

单语 gettext

一些项目决定使用 gettext 作为单语格式——它们仅在源代码中编码 ID，然后将字符串翻译成所有语言，包括英语。支持此功能，尽管在将组件导入 Weblate 时必须明确选择此文件格式。

单语种的 gettext PO 文件通常如下所示：

```
#: 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"
```

基本语言文件将是：

```
#: 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"
```

典型的 Weblate *Component configuration*	
文件掩码	`po/*.po`
单语种译文模版语言文件	`po/en.po`
新翻译的译文模版	`po/messages.pot`
文件格式	*Gettext PO* 文件（单语）

1.10.5 XLIFF

创建基于 XML 的格式来标准化翻译文件，但最终它是该领域中 许多标准 之一。

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

参见：

XML Localization Interchange File Format (XLIFF) specification

翻译状态

在 3.3 版更改: 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.

See 专门的审核者.

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

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.

例如:

```
    <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 定制行为) 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 管理字型).

Typical Weblate *Component configuration* for bilingual XLIFF	
文件掩码	`localizations/*.xliff`
单语种译文模版语言文件	*Empty*
新翻译的译文模版	`localizations/en-US.xliff`
文件格式	*XLIFF Translation File*

Typical Weblate *Component configuration* for monolingual XLIFF	
文件掩码	`localizations/*.xliff`
单语种译文模版语言文件	`localizations/en-US.xliff`
新翻译的译文模版	`localizations/en-US.xliff`
文件格式	*XLIFF Translation File*

参见:

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.

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

典型的 Weblate *Component configuration*	
文件掩码	`src/app/Bundle_*.properties`
单语种译文模版语言文件	`src/app/Bundle.properties`
新翻译的译文模版	*Empty*
文件格式	*Java Properties (ISO-8859-1)*

参见:

Java properties on Wikipedia, Mozilla and Java properties files, 格式化 *Java* 属性文件, 清理翻译文件

1.10.7 GWT properties

Native GWT format for translations.

GWT properties are usually used as monolingual translations.

典型的 Weblate *Component configuration*	
文件掩码	`src/app/Bundle_*.properties`
单语种译文模版语言文件	`src/app/Bundle.properties`
新翻译的译文模版	*Empty*
文件格式	*GWT Properties*

参见:

GWT localization guide Mozilla and Java properties files, 格式化 *Java* 属性文件, 清理翻译文件

1.10.8 INI translations

4.1 新版功能.

INI file format for translations.

INI translations are usually used as monolingual translations.

典型的 Weblate *Component configuration*	
文件掩码	`language/*.ini`
单语种译文模版语言文件	`language/en.ini`
新翻译的译文模版	*Empty*
文件格式	*INI File*

参见:

INI Files, *Joomla translations*, *Inno Setup INI 翻译*

1.10.9 Inno Setup INI 翻译

4.1 新版功能.

用于翻译的 Inno Setup INI 文件格式。

Inno Setup INI 翻译通常用作单语种翻译。

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

典型的 Weblate *Component configuration*	
文件掩码	`language/*.islu`
单语种译文模版语言文件	`language/en.islu`
新翻译的译文模版	*Empty*
文件格式	*Inno Setup INI* 文件

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

参见:

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

1.10.10 Joomla translations

2.12 新版功能.

Native Joomla format for translations.

Joomla translations are usually used as monolingual translations.

典型的 Weblate *Component configuration*	
文件掩码	`language/*/com_foobar.ini`
单语种译文模版语言文件	`language/en-GB/com_foobar.ini`
新翻译的译文模版	*Empty*
文件格式	*Joomla Language File*

参见:

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

1.10.11 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	
文件掩码	`i18n/app.*.ts`
单语种译文模版语言文件	*Empty*
新翻译的译文模版	`i18n/app.de.ts`
文件格式	*Qt Linguist Translation File*

Typical Weblate *Component configuration* when using as monolingual	
文件掩码	`i18n/app.*.ts`
单语种译文模版语言文件	`i18n/app.en.ts`
新翻译的译文模版	`i18n/app.en.ts`
文件格式	*Qt Linguist Translation File*

参见:

Qt Linguist manual, Qt .ts, 双语和单语格式

1.10.12 Android string resources

Android specific file format for translating applications.

Android string resources are monolingual, the *Monolingual base language file* file is stored in a different location from the others `res/values/strings.xml`.

典型的 Weblate *Component configuration*	
文件掩码	`res/values-*/strings.xml`
单语种译文模版语言文件	`res/values/strings.xml`
新翻译的译文模版	*Empty*
文件格式	*Android String Resource*

参见:

Android string resources documentation, Android string resources

---

注解: 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>
```

(下页继续)

(续上页)

```
<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

1.10.13 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.

典型的 Weblate *Component configuration*	
文件掩码	`Resources/*.lproj/Localizable.strings`
单语种译文模版语言文件	`Resources/en.lproj/Localizable.strings`
新翻译的译文模版	*Empty*
文件格式	*iOS Strings (UTF-8)*

参见:

Apple “strings files” documentation, Mac OSX strings

1.10.14 PHP 字符串

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';
```

典型的 Weblate *Component configuration*	
文件掩码	`lang/*/texts.php`
单语种译文模版语言文件	`lang/en/texts.php`
新翻译的译文模版	`lang/en/texts.php`
文件格式	*PHP strings*

Laravel PHP 字符串

在 4.1 版更改.

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

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

参见:

PHP, Localization in Laravel

1.10.15 JSON files

2.0 新版功能.

在 2.16 版更改: Since Weblate 2.16 and with translate-toolkit at-least 2.2.4, nested structure JSON files are supported as well.

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

Weblate currently supports several variants of JSON translations:

- Simple key / value files.
- 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": ""
  }
}
```

警告: Weblate currently handles nested JSON by flattening the keys. This leads to serializing issues when special chars such as `.` or `[]` are used in the actual keys, because Weblate thinks it is indication of nesting.

See <https://github.com/WeblateOrg/weblate/issues/2149>

典型的 Weblate *Component configuration*	
文件掩码	`langs/translation-*.json`
单语种译文模版语言文件	`langs/translation-en.json`
新翻译的译文模版	*Empty*
文件格式	*JSON nested structure file*

参见:

JSON, 自定义 *JSON* 输出, 清理翻译文件,

1.10.16 JSON i18next files

在 2.17 版更改: 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.

注解: 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"
}
```

典型的 Weblate *Component configuration*	
文件掩码	`langs/*.json`
单语种译文模版语言文件	`langs/en.json`
新翻译的译文模版	*Empty*
文件格式	*i18next JSON file*

参见:

JSON, i18next JSON Format, 自定义 *JSON* 输出, 清理翻译文件

1.10.17 go-i18n JSON files

4.1 新版功能.

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

注解： Weblate supports the go-i18n JSON v2 format, it does not support flat JSON files supported in v1.

典型的 Weblate *Component configuration*	
文件掩码	`langs/*.json`
单语种译文模版语言文件	`langs/en.json`
新翻译的译文模版	*Empty*
文件格式	*go-i18n JSON file*

参见：

JSON, go-i18n, 自定义 *JSON* 输出, 清理翻译文件,

1.10.18 ARB File

4.1 新版功能.

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

典型的 Weblate *Component configuration*	
文件掩码	`lib/l10n/intl_*.arb`
单语种译文模版语言文件	`lib/l10n/intl_en.arb`
新翻译的译文模版	*Empty*
文件格式	*ARB file*

参见：

JSON, Application Resource Bundle Specification, Internationalizing Flutter apps, 自定义 *JSON* 输出, 清理翻译文件

1.10.19 WebExtension JSON

2.16 新版功能: 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.

注解： 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",
```

(下页继续)

(续上页)

```
        "example": "https://developer.mozilla.org"
      }
    }
  },
  "orangutan": {
    "message": "",
    "description": "Description"
  },
  "try": {
    "message": "",
    "description": "Description"
  },
  "thanks": {
    "message": "",
    "description": "Description"
  }
}
```

典型的 Weblate *Component configuration*	
文件掩码	`_locales/*/messages.json`
单语种译文模版语言文件	`_locales/en/messages.json`
新翻译的译文模版	*Empty*
文件格式	*WebExtension JSON file*

参见:

JSON, Google chrome.i18n, Mozilla Extensions Internationalization

1.10.20 .XML resource files

2.3 新版功能.

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.

典型的 Weblate *Component configuration*	
文件掩码	`Resources/Language.*.resx`
单语种译文模版语言文件	`Resources/Language.resx`
新翻译的译文模版	*Empty*
文件格式	*.NET 资源文件*

参见:

.NET Resource files (.resx), 清理翻译文件,

1.10.21 CSV 文件

2.4 新版功能.

CSV 文件可以包含源和翻译的简单列表。Weblate 支持以下文件:

- 带有标头定义字段（源，翻译，位置等）的文件。这是推荐的方法，因为它最不容易出错。
- 具有两个字段的文件——源和翻译（按此顺序），选择 *简单 CSV 文件*作为文件格式
- 具有 translate-toolkit 定义的字段的文件: location, source, target, ID, fuzzy, context, translator_comments, developer_comments

警告： CSV 格式当前会自动检测 CSV 文件的方言。在某些情况下，自动检测可能会失败，并且您会得到不同的结果。对于值中包含换行符的 CSV 文件尤其如此。作为一种解决方法，建议省略引号字符。

Example file:

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

典型的 Weblate *Component configuration*	
文件掩码	`locale/*.csv`
单语种译文模版语言文件	*Empty*
新翻译的译文模版	`locale/en.csv`
文件格式	*CSV 文件*

参见：

CSV

1.10.22 YAML files

2.9 新版功能.

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

Example of a YAML file:

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

典型的 Weblate *Component configuration*	
文件掩码	`translations/messages.*.yml`
单语种译文模版语言文件	`translations/messages.en.yml`
新翻译的译文模版	*Empty*
文件格式	*YAML file*

参见：

YAML, *Ruby YAML files*

1.10.23 Ruby YAML files

2.9 新版功能.

Ruby i18n YAML files with language as root node.

Example Ruby i18n YAML file:

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

典型的 Weblate *Component configuration*	
文件掩码	`translations/messages.*.yml`
单语种译文模版语言文件	`translations/messages.en.yml`
新翻译的译文模版	*Empty*
文件格式	*Ruby YAML file*

参见:

YAML, *YAML files*

1.10.24 DTD files

2.18 新版功能.

Example DTD file:

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

典型的 Weblate *Component configuration*	
文件掩码	`locale/*.dtd`
单语种译文模版语言文件	`locale/en.dtd`
新翻译的译文模版	*Empty*
文件格式	*DTD file*

参见:

Mozilla DTD format

1.10.25 Flat XML files

3.9 新版功能.

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>
```

典型的 Weblate *Component configuration*	
文件掩码	`locale/*.xml`
单语种译文模版语言文件	`locale/en.xml`
新翻译的译文模版	*Empty*
文件格式	*Flat XML file*

参见:

Flat XML

1.10.26 Windows RC files

在 4.1 版更改: Support for Windows RC files has been rewritten.

注解: 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
```

典型的 Weblate *Component configuration*	
文件掩码	`lang/*.rc`
单语种译文模版语言文件	`lang/en-US.rc`
新翻译的译文模版	`lang/en-US.rc`
文件格式	*RC file*

参见:

Windows RC files

1.10.27 应用商店元数据文件

3.5 新版功能.

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.

典型的 Weblate *Component configuration*	
文件掩码	`fastlane/android/metadata/*`
单语种译文模版语言文件	`fastlane/android/metadata/en-US`
新翻译的译文模版	`fastlane/android/metadata/en-US`
文件格式	*App store metadata files*

1.10.28 Subtitle files

3.7 新版功能.

Weblate 可以翻译多个字幕文件：

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

典型的 Weblate *Component configuration*	
文件掩码	`path/*.srt`
单语种译文模版语言文件	`path/en.srt`
新翻译的译文模版	`path/en.srt`
文件格式	*SubRip subtitle file*

参见：

Subtitles

1.10.29 Excel Open XML

3.2 新版功能.

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.

1.10.30 HTML files

4.1 新版功能.

注解： 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.

参见：

HTML

1.10.31 OpenDocument Format

4.1 新版功能.

注解： 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.

参见：

OpenDocument Format

1.10.32 IDML Format

4.1 新版功能.

注解： 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.

1.10.33 其它

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.

参见：

Translation Related File Formats

1.10.34 Adding new translations

在 2.18 版更改: 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 新翻译的译文模版 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 *Template for new translations* is empty and the file format supports it, an empty file is created where new strings will be added once they are translated.

The *Language code style* allows you to customize language code used in generated filenames:

基于文件格式的默认值 Dependent on file format, for most of them POSIX is used.

POSIX 风格使用下划线作为分隔 Typically used by gettext and related tools, produces language codes like *pt_BR*.

POSIX 风格使用下划线作为分隔，包括国家/地区代码 POSIX style language code including the country code even when not necessary (for example 'cs_CZ').

GCP 风格使用连字符作为分隔 Typically used on web platforms, produces language codes like *pt-BR*.

GCP 风格使用连字符作为分隔，包括国家/地区代码 BCP style language code including the country code even when not necessary (for example 'cs-CZ').

Android 风格 Only used in Android apps, produces language codes like *pt-rBR*.

Java 风格 Used by Java—mostly BCP with legacy codes for Chinese.

注解： Weblate recognizes any of these when parsing translation files, the above settings only influences how new files are created.

1.10.35 只读字符串

3.10 新版功能.

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 定制行为.

1.11 版本控制集成

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

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 username *weblate* 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 from Weblate side.

The invitations on GitHub are accepted automatically within five minutes, on other services manual processing might be needed, so please be patient.

Once the *weblate* user is added, you can configure 源代码库 and 代码库推送 *URL* using SSH protocol (for example `git@github.com:WeblateOrg/weblate.git`).

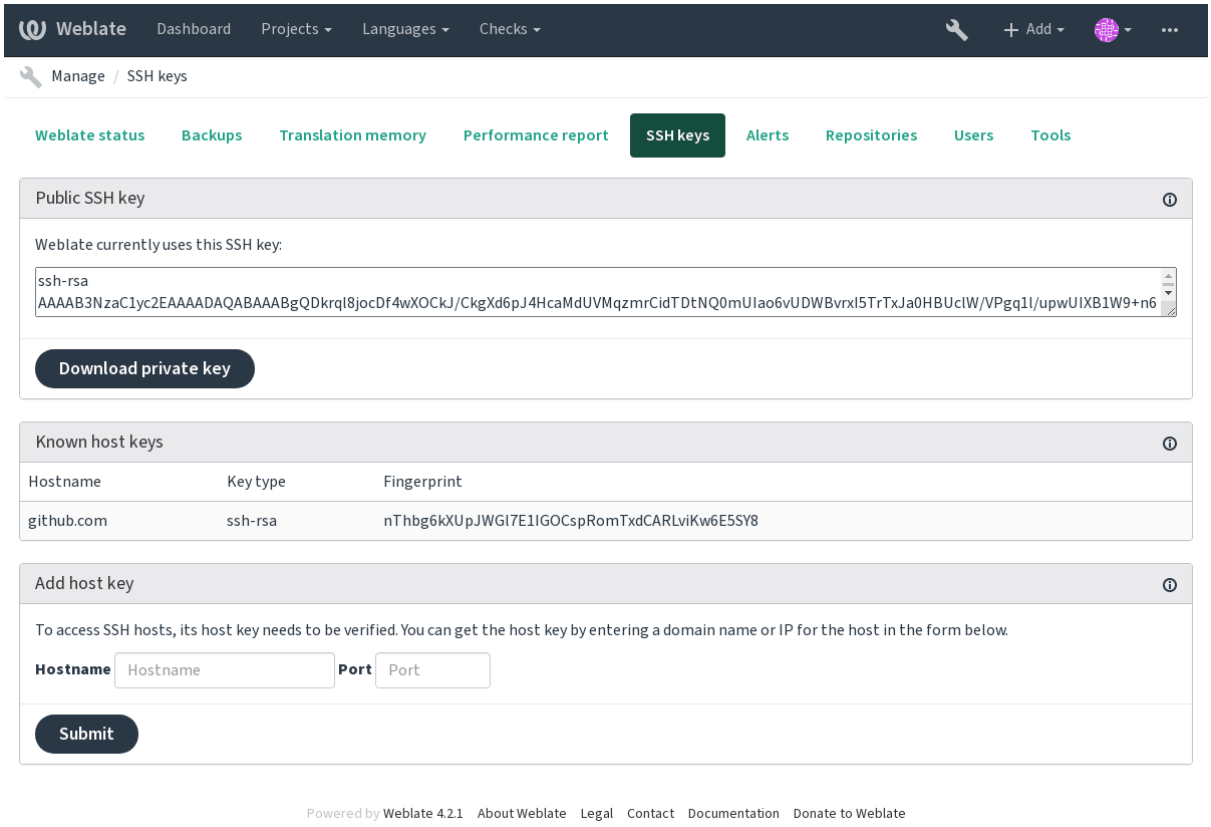
SSH 仓库

访问私有仓库的最常用方法是基于 SSH。授权公共 Weblate SSH 密钥（请参阅*Weblate SSH 密钥*）以这种方式访问上游仓库。

> **警告：** 在 GitHub 上，每个密钥只能添加到一个存储库中，请参阅*GitHub repositories* 和*Accessing repositories from Hosted Weblate*。

Weblate 还会在首次连接时存储主机密钥指纹，并且在以后进行更改时将无法连接到主机（请参阅验证 *SSH 主机密钥*）。

如果需要调整，请从 Weblate 管理界面进行：



Weblate SSH 密钥

Weblate 公钥对浏览 *About* 页面的所有用户可见。

管理员可以在管理界面登录页面的（通过 *SSH keys*）生成或显示 Weblate 当前使用的公共密钥。

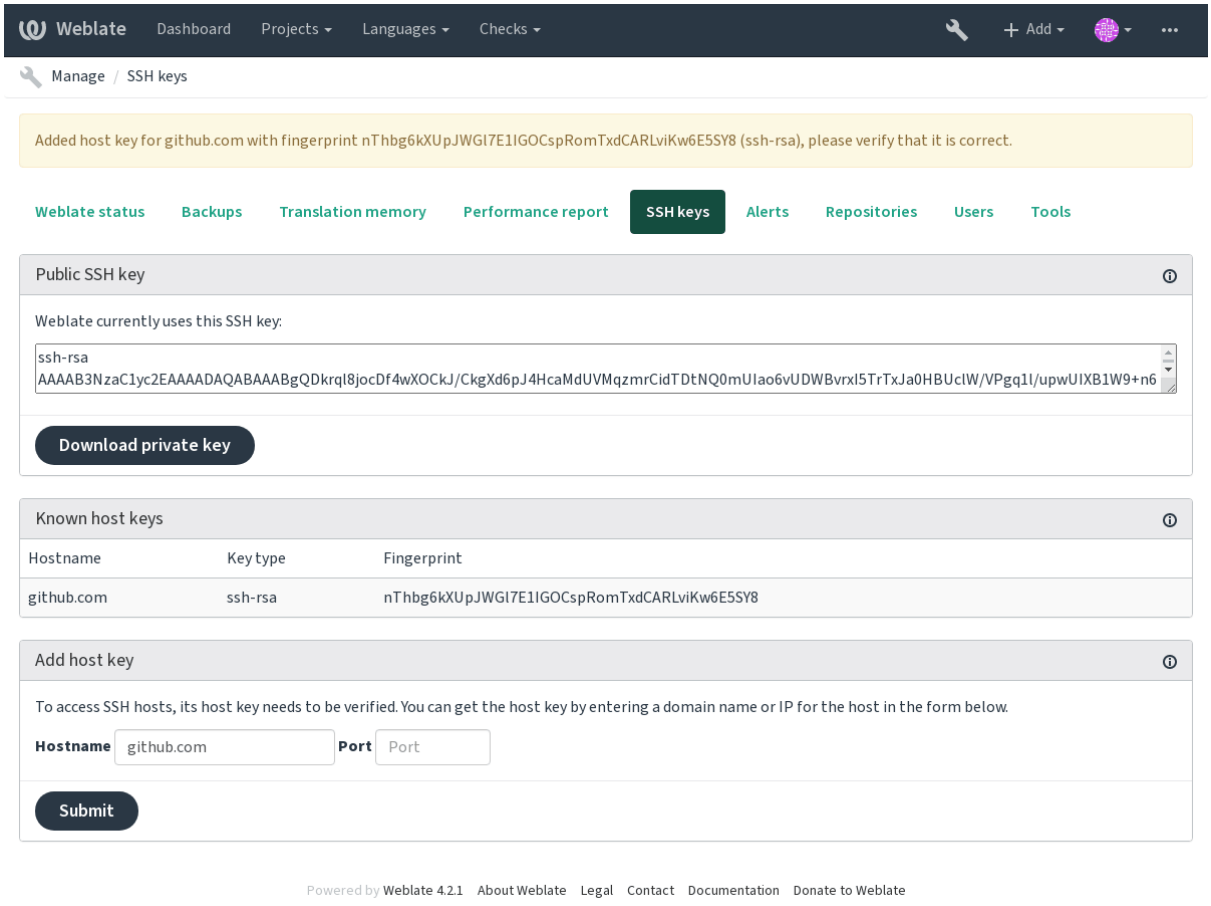
注解： 相应的私有 SSH 密钥当前无法使用密码，因此请确保已受到良好的保护。

提示： 对生成的私有 Weblate SSH 密钥进行备份。

验证 SSH 主机密钥

Weblate 会在第一次访问时自动记住 SSH 主机密钥，并记住它们以备将来使用。

如果要在连接到仓库之前对其进行验证，请在管理界面的同一部分中的 *Add host key* 中验证要访问的服务器的 SSH 主机密钥。输入您要访问的主机名（例如 `gitlab.com`），然后按 *Submit*。验证其指纹与您添加的服务器匹配。它们显示在确认消息中：



GitHub repositories

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

In case the 推送分支 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 密钥*) 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.

参见：

Accessing repositories from Hosted Weblate

Weblate internal URLs

To share one repository between different components you can use a special URL like `weblate://project/component`. This way, the component will share the VCS repository configuration with the referenced component (`project/component` in the example).

Weblate automatically adjusts repository URL when creating component when it finds component with matching repository setup. You can override this in last step of 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 more components sharing single repository, for example 压缩 *Git* 提交.

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`.

注解： If you 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
```

注解： 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.

参见：

The cURL manpage, Git config documentation

1.11.2 Git

参见：

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

Git 强制推送

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.

警告： 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
```

警告： 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

2.3 新版功能.

This adds a thin layer atop *Git* using the hub tool 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.

参见：

推送 Weblate 的更改

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.

参见:

GITHUB_USERNAME, *Setting up hub* for configuration instructions

Setting up hub

Pushing changes to GitHub as pull requests requires a configured hub installation on your server. Follow the installation instructions at https://hub.github.com/ use hub to finish the configuration, for example:

```
# Use DATA_DIR as configured in Weblate settings.py, it is /app/data in the Docker
HOME=${DATA_DIR}/home hub clone octocat/Spoon-Knife
```

The hub will ask you for your GitHub credentials, retrieve a token and store it in `~/.config/hub`. This file has to be readable by the user running Weblate.

注解: Use the username you configured *hub* with, as *GITHUB_USERNAME* (*WEBLATE_GITHUB_USERNAME* for the Docker image).

1.11.4 GitLab

3.9 新版功能.

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

There is no need to use this 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.

参见:

推送 Weblate 的更改

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.

Configure the lab command line tool and set *GITLAB_USERNAME* for this to work.

参见:

GITLAB_USERNAME, *Setting up Lab* for configuration instructions

Setting up Lab

Pushing changes to GitLab as merge requests requires a configured lab installation on your server. Follow the installation instructions at lab and run it without any arguments to finish the configuration, for example:

```
# Use DATA_DIR as configured in Weblate settings.py, it is /app/data in the Docker
$ HOME=${DATA_DIR}/home lab
Enter GitLab host (default: https://gitlab.com):
Create a token here: https://gitlab.com/profile/personal_access_tokens
Enter default GitLab token (scope: api):
(Config is saved to ~/.config/lab.hcl)
```

The lab will ask you for your GitLab access token, retrieve it and store it in `~/.config/lab.hcl`. The file has to be readable by the user running Weblate.

注解： Use the username you configured *lab* with, as *GITLAB_USERNAME* (*WEBLATE_GITLAB_USERNAME* for the Docker image).

1.11.5 Gerrit

2.2 新版功能.

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

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

1.11.6 Mercurial

2.1 新版功能.

Mercurial is another VCS you can use directly in Weblate.

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

参见：

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

1.11.7 Subversion

2.8 新版功能.

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.

注解： 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 foud 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.

在 2.19 版更改: Before this, there was only support for standard layout repositories.

Subversion credentials

Weblate expects you to have accepted the certificate up-front and if needed, your credentials. 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
```

参见：

DATA_DIR

1.11.8 Local files

3.8 新版功能.

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 的 REST API

2.6 新版功能: 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 客户端*.

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.

Request Headers

- Accept –the response content type depends on Accept header
- Authorization –optional token to authenticate

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 failure (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

- 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
```

(下页继续)

(续上页)

```
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/
```

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

在 4.1 版更改: Added ratelimiting status headers.

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
```

(下页继续)

(续上页)

```
{
    "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 用户

4.0 新版功能.

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.

参见:

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

POST /api/users/
Creates a new user.

Parameters

- **username** (*string*) –用户名
- **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/"
    ],
```

(下页继续)

(续上页)

```
    "is_superuser": true,
    "is_active": true,
    "date_joined": "2020-03-29T18:42:42.617681Z",
    "url": "http://example.com/api/users/exampleusername/"
}
```

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*)**/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 群组

4.0 新版功能.

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.

参见:

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

POST /api/groups/
Creates a new group.

Parameters

- **name** (*string*) –组名
- **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/",
```

(下页继续)

(续上页)

```
    "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 角色

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

参见:

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 语言

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

参见:

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

POST /api/languages/
Creates a new language.

Parameters

- **code** (*string*) –语言名称
- **name** (*string*) –语言名称
- **direction** (*string*) –Language direction
- **plural** (*object*) –Language plural formula and number

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

Parameters

- **language** (*string*) –语言代码

Response JSON Object

- **code** (*string*) –语言代码
- **direction** (*string*) –文字方向
- **plural** (*object*) –Object of language plural information
- **aliases** (*array*) –Array of aliases for language

Example JSON data:

```
{
    "code": "en",
    "direction": "ltr",
    "name": "English",
    "plural": {
        "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*) –语言名称
- **direction** (*string*) –Language direction
- **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*) –语言名称
- **direction** (*string*) –Language direction
- **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*) –语言代码

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 strings
- **fuzzy_percent** (*int*) –percentage of fuzzy strings
- **failing** (*int*) –number of failing strings
- **failing** –percentage of failing strings

1.12.7 项目

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

参见:

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

POST /api/projects/
3.9 新版功能.

Creates a new project.

Parameters

- **name** (*string*) –项目名称
- **slug** (*string*) –Project slug
- **web** (*string*) –项目网站
- **source_language** (*string*) –Project source language code (optional)

GET /api/projects/(string: *project*)/
Returns information about a project.

Parameters

- **project** (*string*) –项目 URL slug

Response JSON Object

- **name** (*string*) –project name
- **slug** (*string*) –project slug
- **source_language** (*object*) –source language object; see *GET /api/languages/(string:language)/*
- **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/*

Example JSON data:

```
{
    "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/"
}
```

DELETE /api/projects/(string: *project*)/
3.9 新版功能.

Deletes a project.

Parameters

- **project** (*string*) –项目 URL slug

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*) –项目 URL slug

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*) –项目 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

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*) –项目 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/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*) –项目 URL slug

Response JSON Object

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

POST /api/projects/(string: *project*)/components/
3.9 新版功能.

Creates translation components in the given project.

Parameters

- **project** (*string*) –项目 URL slug

Response JSON Object

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

CURL example:

```
curl \
    --data-binary '{
        "branch": "master",
        "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"
    }' \
```

(下页继续)

(续上页)

```
    -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": "master",
    "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": "master",
    "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/",
        "web": "https://weblate.org/",
        "web_url": "http://example.com/projects/hello/"
```

(下页继续)

(续上页)

```
    },
    "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.

3.8 新版功能.

Parameters

- **project** (*string*) –项目 URL slug

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.

3.8 新版功能.

Parameters

- **project** (*string*) –项目 URL slug

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 组件

GET /api/components/
Returns a list of translation components.

参见:

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*) –项目 URL slug
- **component** (*string*) –组件 URL slug

Response JSON Object

- **project** (*object*) –the translation project; see *GET /api/projects/(string:project)/*
- **name** (*string*) –组件名称
- **slug** (*string*) –*Component slug*
- **vcs** (*string*) –版本控制系统
- **repo** (*string*) –源代码库
- **git_export** (*string*) –已导出代码库 *URL*
- **branch** (*string*) –仓库分支
- **push_branch** (*string*) –推送分支
- **filemask** (*string*) –文件掩码
- **template** (*string*) –单语种译文模版语言文件
- **edit_template** (*string*) –编辑译文模版文件
- **intermediate** (*string*) –中间语言文件
- **new_base** (*string*) –新翻译的译文模版
- **file_format** (*string*) –文件格式
- **license** (*string*) –翻译许可证
- **agreement** (*string*) –贡献者协议
- **new_lang** (*string*) –添加新翻译
- **language_code_style** (*string*) –语言代码风格
- **push** (*string*) –代码库推送 *URL*
- **check_flags** (*string*) –翻译标记
- **priority** (*string*) –优先权
- **enforced_checks** (*string*) –强制检查
- **restricted** (*string*) –*Restricted access*
- **repoweb** (*string*) –代码库浏览器
- **report_source_bugs** (*string*) –*Source string bug report address*
- **merge_style** (*string*) –合并方式
- **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*) –允许同步翻译
- **enable_suggestions** (*string*) –启用建议
- **suggestion_voting** (*string*) –建议投票
- **suggestion_autoaccept** (*string*) –自动接受建议
- **push_on_commit** (*string*) –提交时推送
- **commit_pending_age** (*string*) –对变更进行提交的延时时间
- **auto_lock_error** (*string*) –出错时锁定
- **language_regex** (*string*) –语言筛选
- **variant_regex** (*string*) –正则表达式变体
- **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/*

Example JSON data:


```
{
    "branch": "master",
    "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/",
        "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/",
```


(下页继续)

(续上页)

```
    "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*) –项目 URL slug
- **component** (*string*) –组件 URL slug

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
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{
    "branch": "master",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "new name",
    "slug": "weblate",
    "project": {
        "name": "Hello",
```

(下页继续)

(续上页)


```
        "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*) –项目 URL slug
- **component** (*string*) –组件 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*/ 3.9 新版功能.

Deletes a component.

Parameters

- **project** (*string*) –项目 URL slug
- **component** (*string*) –组件 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*) –项目 URL slug
- **component** (*string*) –组件 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*) –项目 URL slug
- **component** (*string*) –组件 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*) –项目 URL slug
- **component** (*string*) –组件 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*) –项目 URL slug
- **component** (*string*) –组件 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
```

(下页继续)

(续上页)

```
{"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*) –项目 URL slug
- **component** (*string*) –组件 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*) –项目 URL slug
- **component** (*string*) –组件 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*) –项目 URL slug
- **component** (*string*) –组件 URL slug

GET /api/components/(string: *project*)/
string: *component*/new_template/ Downloads template file for new translations.

Parameters

- **project** (*string*) –项目 URL slug
- **component** (*string*) –组件 URL slug

GET /api/components/(string: *project*)/
string: *component*/translations/ Returns a list of translation objects in the given component.

Parameters

- **project** (*string*) –项目 URL slug
- **component** (*string*) –组件 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*) –项目 URL slug
- **component** (*string*) –组件 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,
    "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/"
```

(下页继续)

(续上页)

```
}
```

GET /api/components/(string: *project*)/(string: *component*)/statistics/ Returns paginated statistics for all translations within component.

2.7 新版功能.

Parameters

- **project** (*string*) –项目 URL slug
- **component** (*string*) –组件 URL slug

Response JSON Object

- **results** (*array*) –array of translation statistics objects; see *GET /api/translations/(string:project)/(string:component)/(string:language)/statistics/*

1.12.9 翻译

GET /api/translations/
Returns a list of translations.

参见:

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*) –项目 URL slug
- **component** (*string*) –组件 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*) –未通过检查的字符串数目
- **failing_checks_percent** (*float*) –未通过检查的字符串比重
- **failing_checks_words** (*int*) –未通过检查的单词数目
- **filename** (*string*) –translation filename
- **fuzzy** (*int*) –number of strings marked for review
- **fuzzy_percent** (*float*) –percentage of strings marked for review
- **fuzzy_words** (*int*) –number of words marked for review
- **have_comment** (*int*) –number of strings with comment
- **have_suggestion** (*int*) –number of strings with suggestion
- **is_template** (*boolean*) –译文是否有单语基础
- **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": "master",
        "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",
```


(下页继续)

(续上页)


```
        "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": {
        "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***/** 3.9 新版功能.

Deletes a translation.

Parameters

- **project** (*string*) –项目 URL slug
- **component** (*string*) –组件 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*) –项目 URL slug
- **component** (*string*) –组件 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*) –项目 URL slug
- **component** (*string*) –组件 URL slug
- **language** (*string*) –Translation language code

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*) –项目 URL slug
- **component** (*string*) –组件 URL slug
- **language** (*string*) –Translation language code

Request JSON Object

- **key** (*string*) –Name of translation unit
- **value** (*string*) –The translation unit value

POST /api/translations/(string: *project*)/
string: *component*/string: *language*/autotranslate/ Trigger automatic translation.

Parameters

- **project** (*string*) –项目 URL slug
- **component** (*string*) –组件 URL slug
- **language** (*string*) –Translation language code

Request JSON Object

- **mode** (*string*) –自动翻译模式
- **filter_type** (*string*) –Automatic translation filter type
- **auto_source** (*string*) –自动翻译来源
- **component** (*string*) –为项目打开对共享翻译记忆库的贡献，以访问其他组件。
- **engines** (*string*) –机器翻译引擎
- **threshold** (*string*) –匹配分数阈值

GET /api/translations/(string: *project*)/
string: *component*/string: *language*/file/ Download current translation file as stored in VCS (without `format` parameter) or as converted to a standard format (currently supported: Gettext PO, MO, XLIFF and TBX).

注解： 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*) –项目 URL slug
- **component** (*string*) –组件 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*) –项目 URL slug
- **component** (*string*) –组件 URL slug
- **language** (*string*) –Translation language code

Form Parameters

- **string conflicts** –How to deal with conflicts (`ignore`, `replace-translated` or `replace-approved`)
- **file file** –Uploaded file
- **string email** –作者邮箱
- **string author** –作者姓名
- **string method** –Upload method (`translate`, `approve`, `suggest`, `fuzzy`, `replace`, `source`), see *Import methods*
- **string fuzzy** –Fuzzy 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*) –项目 URL slug
- **component** (*string*) –组件 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*) –项目 URL slug
- **component** (*string*) –组件 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.

2.7 新版功能.

Parameters

- **project** (*string*) –项目 URL slug
- **component** (*string*) –组件 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 strings needing review
- **fuzzy_percent** (*float*) –percentage of strings needing review
- **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

2.10 新版功能.

GET /api/units/
Returns list of translation units.

参见:

Unit object attributes are documented at *GET /api/units/(int:id)/*.

GET /api/units/(int: *id*)/
Returns information about translation unit.

Parameters

- **id** (*int*) –Unit ID

Response JSON Object

- **translation** (*string*) –URL of a related translation object
- **source** (*string*) –source string
- **previous_source** (*string*) –previous source string used for fuzzy matching
- **target** (*string*) –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
- **fuzzy** (*boolean*) –是否该单元是模糊的或标记为需要检查
- **translated** (*boolean*) –单元是否被翻译
- **approved** (*boolean*) –翻译是否被核准
- **position** (*int*) –unit position in translation file
- **has_suggestion** (*boolean*) –单元是否有翻译建议
- **has_comment** (*boolean*) –单元是否有评论
- **has_failing_check** (*boolean*) –单元是否有未通过检查的翻译
- **num_words** (*int*) –number of source words
- **priority** (*int*) –translation priority; 100 is default
- **id** (*int*) –unit identifier
- **web_url** (*string*) –URL where the unit can be edited
- **souce_info** (*string*) –Source string information link; see *GET /api/units/(int:id)/*

1.12.11 修改

2.10 新版功能.

GET /api/changes/

在 4.1 版更改: Filtering of changes was introduced in the 4.1 release.

Returns a list of translation changes.

参见:

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
- **glossary_term** (*string*) –URL of a related glossary term 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 截图

2.14 新版功能.

GET /api/screenshots/
Returns a list of screenshot string information.

参见:

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
- **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*)/units/
int: *unit_id* 删除与截图关联的源字符串。

Parameters

- **id** (*int*) –Screenshot ID
- **unit_id** –源字符串单元 ID

POST /api/screenshots/
Creates a new screenshot.

Form Parameters

- **file image** –Uploaded file
- **string name** –截图名称
- **string project_slug** –Project Slug
- **string component_slug** –Component Slug

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*)/
编辑截屏的部分信息。

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*)/
编辑截屏的完整信息。

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*)/
删除截图

Parameters

- **id** (*int*) –Screenshot ID

1.12.13 组件列表

4.0 新版功能.

GET /api/component-lists/
Returns a list of component lists.

参见:

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

1.12.14 通知钩子

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***/** 2.6 版后已移除: 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*)**/**
2.6 版后已移除: 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.

注解: 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.

参见:

从 *GitHub* 自动接收更改 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.

参见:

从 *GitLab* 自动接收更改 For instruction on setting up GitLab integration

https://docs.gitlab.com/ce/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.

参见:

从 *Bitbucket* 自动接收更改 For instruction on setting up Bitbucket integration

https://confluence.atlassian.com/bitbucket/manage-webhooks-735643732.html Generic information about Bitbucket Webhooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/pagure/
3.3 新版功能.

Special hook for handling Pagure notifications and automatically updating matching components.

参见:

从 *Pagure* 自动接受更改 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/
3.8 新版功能.

Special hook for handling Azure Repos notifications and automatically updating matching components.

参见:

从 *Azure Repos* 自动接收更改 For instruction on setting up Azure integration

https://docs.microsoft.com/en-us/azure/devops/service-hooks/services/webhooks Generic information about Azure Repos Web Hooks

ENABLE_HOOKS For enabling hooks for whole Weblate

POST /hooks/gitea/

3.9 新版功能.

Special hook for handling Gitea Webhook notifications and automatically updating matching components.

参见:

从 *Gitea Repos* 自动接收更改 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/

3.9 新版功能.

Special hook for handling Gitee Webhook notifications and automatically updating matching components.

参见:

从 *Gitee Repos* 自动接收更改 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

1.12.15 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`

2.6 版后已移除: 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/master/ HTTP/1.1
Host: example.com
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

[
    {
```

(下页继续)

(续上页)


```
        "code": "cs",
        "failing": 0,
        "failing_percent": 0.0,
        "fuzzy": 0,
        "fuzzy_percent": 0.0,
        "last_author": "Michal \u010ciha\u0159",
        "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/master/cs/
↪"
    },
    {
        "code": "nl",
        "failing": 21,
        "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/master/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/master/el/
↪"
    },
]
```

1.12.16 RSS 频道

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.

参见:

RSS on wikipedia

1.13 Weblate 客户端

2.7 新版功能: 自从 Weblate 2.7 以来，已经有完整的 wlc 实用程序支持。如果您使用的是旧版本，则可能会与 API 发生某些不兼容。

1.13.1 安装

Weblate 客户端是单独提供的，包括 Python 模块。要使用下面的命令，您需要安装*wlc*:

```
pip3 install wlc
```

1.13.2 入门

wlc 配置存储在 `~/.config/weblate` 中，请创建它以匹配您的环境:

```
[weblate]
url = https://hosted.weblate.org/api/

[keys]
https://hosted.weblate.org/api/ = APIKEY
```

然后，您可以在默认服务器上调用命令:

```
wlc ls
wlc commit sandbox/hello-world
```

参见:

配置文件

1.13.3 概要

```
wlc [parameter] <command> [options]
```

命令实际上指示应该执行哪个操作。

1.13.4 描述

Weblate 客户端是一个 Python 库和命令行实用程序，可使用 API 远程管理 Weblate。命令行实用程序可以作为 **wlc** 调用，并且内置在*wlc* 上。

网站范围内的选项

程序为整个实例接受以下选项，必须在任何子命令之前输入这些选项。

--format {csv,json,text,html}
指定输出格式。

--url URL
指定 API URL。覆盖在配置文件中找到的任何值，请参阅配置文件。该网址应以 /api/ 结尾，例如 https://hosted.weblate.org/api/。

--key KEY
指定要使用的 API 用户密钥。覆盖在配置文件中找到的任何值，请参阅配置文件。您可以在 Weblate 的个人资料中找到密钥。

--config PATH
覆盖配置文件路径，请参阅配置文件。

--config-section SECTION
覆盖正在使用的配置文件部分，请参阅配置文件。

子命令

以下子命令可用：

version
打印当前版本。

list-languages
列出 Weblate 中使用的语言。

list-projects
列出 Weblate 中的项目。

list-components
列出 Weblate 中的组件。

list-translations
列出 Weblate 中的翻译。

show
显示 Weblate 对象（翻译，组件或项目）。

ls
列出 Weblate 对象（翻译，组件或项目）。

commit
提交在 Weblate 对象（翻译，组件或项目）中所做的更改。

pull
将远程仓库更改拉入 Weblate 对象（翻译，组件或项目）。

push
将 Weblate 对象更改推送到远程仓库（翻译，组件或项目）。

reset
0.7 新版功能: 自 wlc 0.7 起受支持。

重置 Weblate 对象中的更改以匹配远程存储库（翻译，组件或项目）。

cleanup
0.9 新版功能: 从 wlc 0.9 开始受支持。

删除 Weblate 对象中所有未跟踪的更改以匹配远程仓库（翻译，组件或项目）。

repo
显示给定 Weblate 对象（翻译，组件或项目）的仓库状态。

statistics
显示给定 Weblate 对象（翻译，组件或项目）的详细统计信息。

lock-status
0.5 新版功能: 自 wlc 0.5 起受支持。

显示锁定状态。

lock
0.5 新版功能: 自 wlc 0.5 起受支持。

锁定组件以防止在 Weblate 中进一步翻译。

unlock
0.5 新版功能: 自 wlc 0.5 起受支持。

解锁 Weblate 组件的翻译。

changes
0.7 新版功能: 从 wlc 0.7 和 Weblate 2.10 开始受支持。

显示给定对象的更改。

download
0.7 新版功能: 自 wlc 0.7 起受支持。

下载翻译文件。

--convert
转换文件格式，如果未指定，则在服务器上不进行任何转换，并且将文件原样下载到仓库中。

--output
指定要保存输出的文件，如果未指定，则将其打印到 stdout。

upload
0.9 新版功能: 从 wlc 0.9 开始受支持。

上传翻译文件。

--overwrite
上传时覆盖现有翻译。

--input
从中读取内容的文件，如果未指定，则从 stdin 中读取。

1.13.5 配置文件

.weblate 每个项目的配置文件

~/.config/weblate 用户配置文件

/etc/xdg/weblate 系统范围的配置文件

该程序遵循 XDG 规范，因此您可以通过环境变量 XDG_CONFIG_HOME 或 XDG_CONFIG_DIRS 来调整配置文件的位置。

可以在 [weblate] 部分中配置以下设置（您可以通过*--config-section*进行自定义）：

key
: 用于访问 Weblate 的 API KEY。

url
: API 服务器网址，默认为 http://127.0.0.1:8000/api/。

translation
: 默认翻译的路径——组件或项目。

配置文件是一个 INI 文件，例如：

```
[weblate]
url = https://hosted.weblate.org/api/
key = APIKEY
translation = weblate/master
```

另外，API 密钥可以存储在 [keys] 部分中：

```
[keys]
https://hosted.weblate.org/api/ = APIKEY
```

这样，您就可以在 VCS 仓库中使用 .weblate 配置时，将密钥存储在个人设置中，以便 wlc 知道它应该与哪个服务器通信。

1.13.6 示例

打印当前程序版本：

```
$ wlc version
version: 0.1
```

列出所有项目：

```
$ wlc list-projects
name: Hello
slug: hello
source_language: en
url: http://example.com/api/projects/hello/
web: https://weblate.org/
web_url: http://example.com/projects/hello/
```

您还可以指定 wlc 应该从事的项目：

```
$ cat .weblate
[weblate]
url = https://hosted.weblate.org/api/
translation = weblate/master

$ wlc show
branch: master
```

(下页继续)

(续上页)

```
file_format: po
filemask: weblate/locale/*/LC_MESSAGES/django.po
git_export: https://hosted.weblate.org/git/weblate/master/
license: GPL-3.0+
license_url: https://spdx.org/licenses/GPL-3.0+
name: master
new_base: weblate/locale/django.pot
project: weblate
repo: git://github.com/WeblateOrg/weblate.git
slug: master
template:
url: https://hosted.weblate.org/api/components/weblate/master/
vcs: git
web_url: https://hosted.weblate.org/projects/weblate/master/
```

通过此设置，可以轻松地提交当前项目中的未决更改：

```
$ wlc commit
```

1.14 Weblate’s Python API

1.14.1 安装

The Python API is shipped separately, you need to install the *Weblate 客户端*: (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*)

参数

- **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*)

参数 **path** (*str*) –Request path

返回类型 object

Performs a single API GET call.

post(*path*, ***kwargs*)

参数 **path** (*str*) –Request path

返回类型 object

Performs a single API GET call.

1.14.3 wlc.config

WeblateConfig

class wlc.config.**WeblateConfig**(*section='wlc'*)

参数 **section** (*str*) –Configuration section to use

Configuration file parser following XDG specification.

load(*path=None*)

参数 **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*)

参数

- **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.

CHAPTER 2

Administrator docs

2.1 配置手册

2.1.1 安装 Weblate

使用 Docker 安装

通过 dockerized Weblate 部署，您可以在几秒钟内启动并运行您的个人 Weblate 实例。Weblate 的所有依赖项已包含在内。PostgreSQL 被设置为默认数据库。

硬件要求

Weblate 应该可以在所有现代硬件上正常运行，以下是在单个主机（Weblate，数据库和 Web 服务器）上运行 Weblate 所需的最低配置：

- 2 GB 的内存
- 2 个 CPU 核心
- 1 GB 的存储空间

内存越多越好——用于所有级别的缓存（文件系统，数据库和 Weblate）。

许多并发用户会增加所需的 CPU 内核数量。对于数百个翻译组件，建议至少有 4 GB 的内存。

注解： 根据 Weblate 中管理的翻译大小，安装 Weblate 的实际要求差异很大。

安装

以下示例假设您拥有一个工作正常的 Docker 环境，并安装了 `docker-compose` 。请查看 Docker 文档以获取说明。

1. 克隆 weblate-docker 存储库：

```
git clone https://github.com/WeblateOrg/docker-compose.git weblate-docker
cd weblate-docker
```

2. 使用您的设置创建一个 `docker-compose.override.yml` 文件。请参阅*Docker 环境变量* 以获取环境变量的完整列表。

```
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
```

> **注解：** 如果未设置 *WEBLATE_ADMIN_PASSWORD*，则使用首次启动时显示的随机密码创建管理员用户。
>
> 提供的示例使 Weblate 侦听端口 80，在 `docker-compose-override.yml` 文件中编辑端口映射以进行更改。

3. 启动 Weblate 容器：

```
docker-compose up
```

享受您的 Weblate 部署，可以在 `weblate` 容器的端口 80 上进行访问。

在 2.15-2 版更改: 最近更改了设置，以前有单独的 web 服务器容器，因为 2.15-2 开始，web 服务器已嵌入 Weblate 容器中。

在 3.7.1-6 版更改: 在 2019 年 7 月（从 3.7.1-6 标签开始）中，容器未以 root 用户身份运行。这已将裸露端口从 80 更改为 8080。

参见：

Invoking management commands

具有 HTTPS 支持的 Docker 容器

请参阅安装 以获取常规部署说明，本节仅提及与之相比的差异。

使用自己的 SSL 证书

3.8-3 新版功能.

如果您要使用自己的 SSL 证书，只需将文件放入 Weblate 数据卷中（请参阅*Docker 容器卷*）：

- `ssl/fullchain.pem` 包含证书，包括任何需要的 CA 证书
- `ssl/privkey.pem` 包含有私钥

拥有这两个文件的用户必须与启动 docker 容器并将文件掩码设置为 `600`（仅拥有用户可读可写）的用户为同一用户。

此外，Weblate 容器现在将在端口 4443 上接受 SSL 连接，您将要在 docker compose override 中包括 HTTPS 的端口转发：

```
version: '3'
services:
  weblate:
    ports:
      - 80:8080
      - 443:4443
```

如果您已经在同一服务器上托管其他站点，则反向代理（例如 NGINX ）可能会使用端口 `80` 和 `443`。要将 HTTPS 连接从 NGINX 传递到 docker 容器，可以使用以下配置：

```
server {
    listen 443;
    listen [::]:443;

    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>;
    }
}
```

将 `<SITE_URL>`，`<SITE>` 和 `<EXPOSED_DOCKER_PORT>` 替换为您环境中的实际值。

使用 Let’s Encrypt 自动生成 SSL 证书

如果要在公共安装中使用 Let’s Encrypt 自动生成的 SSL 证书，则需要在其他 Docker 容器中添加反向 HTTPS 代理，这将使用 https-portal。这是在 `docker-compose-https.yml` 文件中使用的。然后使用您的设置创建一个 `docker-compose-https.override.yml` 文件：

```
version: '3'
services:
  weblate:
    environment:
```

(下页继续)

(续上页)

```
      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'
```

每当调用 **docker-compose** 时，您都需要将两个文件都传递给它，然后执行以下操作：

```
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
```

升级 Docker 容器

通常，只更新 Weblate 容器并保持 PostgreSQL 容器为您的版本是一个好主意，因为升级 PostgreSQL 会很痛苦，并且在大多数情况下不会带来很多好处。

您可以通过坚持使用现有的 docker-compose 并仅提取最新镜像然后重新启动来执行此操作：

```
docker-compose stop
docker-compose pull
docker-compose up
```

Weblate 数据库应在首次启动时自动迁移，并且不需要其他手动操作。

注解： Weblate 不支持跨 3.0 的升级。如果您使用的是 2.x 系列，并且要升级到 3.x，请首先升级到最新的 3.0.1-x（在撰写本文时为 `3.0.1-7`）镜像，它将进行迁移和然后继续升级到较新版本。

您可能还想更新 `docker-compose` 仓库，尽管在大多数情况下并不需要。在这种情况下，请注意 PostgreSQL 版本的更改，因为升级数据库并不容易，请参见 GitHub issue 以获取更多信息。

管理员登录

设置容器之后，您可以使用 *WEBLATE_ADMIN_PASSWORD* 中提供的密码以 管理员 用户身份登录，或者如果未设置该密码，则在首次启动时生成随机密码。

要重置 管理员 密码，请在 *WEBLATE_ADMIN_PASSWORD* 设置为新密码的情况下重启容器。

参见：

WEBLATE_ADMIN_PASSWORD, *WEBLATE_ADMIN_NAME*, *WEBLATE_ADMIN_EMAIL*

Docker 环境变量

可以使用环境变量在 Docker 容器中设置许多 Weblate 的配置：

通用设置

WEBLATE_DEBUG
使用*DEBUG*配置 Django 调试模式。

示例：

```
environment:
  WEBLATE_DEBUG: 1
```

参见：

Disable debug mode.

WEBLATE_LOGLEVEL
配置日志记录的详细程度。

WEBLATE_SITE_TITLE
更改所有页面页眉上显示的站点标题。

WEBLATE_SITE_DOMAIN
配置网站域名.

提示： In case it is not set, the first item from *WEBLATE_ALLOWED_HOSTS* is used.

参见：

Set correct site domain, *SITE_DOMAIN*

WEBLATE_ADMIN_NAME

WEBLATE_ADMIN_EMAIL
配置站点管理员的姓名和电子邮件。它用于*ADMINS*设置和创建 管理员用户（有关此信息，请参阅*WEBLATE_ADMIN_PASSWORD*）。

示例：

```
environment:
  WEBLATE_ADMIN_NAME: Weblate admin
  WEBLATE_ADMIN_EMAIL: noreply@example.com
```

参见：

管理员登录, *Properly configure admins*, *ADMINS*

WEBLATE_ADMIN_PASSWORD
设置 管理员用户的密码。

- 如果未设置并且 管理员用户不存在，则会使用首次启动容器时显示的随机密码来创建它。
- 如果未设置并且 管理员用户存在，则不执行任何操作。
- 如果设置，则在每次容器启动时都会对 管理员用户进行调整，以匹配*WEBLATE_ADMIN_PASSWORD*，*WEBLATE_ADMIN_NAME*和*WEBLATE_ADMIN_EMAIL*。

警告： 将密码存储在配置文件中可能会带来安全风险。考虑仅将此变量用于初始设置（或让 Weblate 在初始启动时生成随机密码）或用于密码恢复。

参见：

管理员登录, *WEBLATE_ADMIN_PASSWORD*, *WEBLATE_ADMIN_NAME*, *WEBLATE_ADMIN_EMAIL*

WEBLATE_SERVER_EMAIL

WEBLATE_DEFAULT_FROM_EMAIL
配置外发电子邮件的地址。

参见:

Configure e-mail sending

WEBLATE_ALLOWED_HOSTS
使用*ALLOWED_HOSTS*配置允许的 HTTP 主机名。

Defaults to * which allows all hostnames.

示例:

```
environment:
  WEBLATE_ALLOWED_HOSTS: weblate.example.com,example.com
```

参见:

ALLOWED_HOSTS, *Allowed hosts setup*, *Set correct site domain*

WEBLATE_REGISTRATION_OPEN
通过切换*REGISTRATION_OPEN*配置是否打开注册。

示例:

```
environment:
  WEBLATE_REGISTRATION_OPEN: 0
```

WEBLATE_REGISTRATION_ALLOW_BACKENDS
配置可用于通过*REGISTRATION_ALLOW_BACKENDS*创建新帐户的身份验证方法。

示例:

```
environment:
  WEBLATE_REGISTRATION_OPEN: 0
  WEBLATE_REGISTRATION_ALLOW_BACKENDS: azuread-oauth2,azuread-tenant-
↪oauth2
```

WEBLATE_TIME_ZONE
在 Weblate 中配置使用的时区，请参阅 TIME_ZONE。

注解: To change the time zone of the Docker container itself, use the TZ environment variable.

示例:

```
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.

注解: This does not make the Weblate container accept HTTPS connections, you need to configure that as well, see *具有 HTTPS 支持的 Docker 容器* for examples.

示例:

```
environment:
  WEBLATE_ENABLE_HTTPS: 1
```

参见：

Set correct site domain

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*.

注解： 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`.

示例：

```
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.

示例：

```
environment:
  WEBLATE_SECURE_PROXY_SSL_HEADER: HTTP_X_FORWARDED_PROTO,https
```

参见：

SECURE_PROXY_SSL_HEADER

WEBLATE_REQUIRE_LOGIN
Configures login required for the whole of the Weblate installation using *LOGIN_REQUIRED_URLS*.

示例：

```
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 login 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*.

参见：

GitHub, *Setting up hub*

WEBLATE_GITLAB_USERNAME
Configures GitLab username for GitLab merge-requests by changing *GITLAB_USERNAME*

参见:

GitLab Setting up Lab

WEBLATE_GITLAB_HOST
Configures GitLab Host for GitLab merge-requests

参见:

GitLab Setting up Lab

WEBLATE_GITLAB_TOKEN
Configures GitLab access token for GitLab merge-requests

参见:

GitLab Setting up Lab

WEBLATE_SIMPLIFY_LANGUAGES
Configures the language simplification policy, see *SIMPLIFY_LANGUAGES*.

WEBLATE_DEFAULT_ACCESS_CONTROL
Configures the default 访问控制 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 允许同步翻译 for new components, see *DEFAULT_TRANSLATION_PROPAGATION*.

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*.

参见:

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.

参见:

Content security policy, *CSP_SCRIPT_SRC*, *CSP_IMG_SRC*, *CSP_CONNECT_SRC*, *CSP_STYLE_SRC*, *CSP_FONT_SRC*

Machine translation settings

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_VERSION
Configures *DeepL* API version to use, see *MT_DEEPL_API_VERSION*.

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*.

示例：

```
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.
  # 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: CN=Users,DC=example,DC=com
  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)
```

参见:

LDAP 身份验证

GitHub

WEBLATE_SOCIAL_AUTH_GITHUB_KEY

WEBLATE_SOCIAL_AUTH_GITHUB_SECRET
Enables *GitHub 身份验证*.

Bitbucket

WEBLATE_SOCIAL_AUTH_BITBUCKET_KEY

WEBLATE_SOCIAL_AUTH_BITBUCKET_SECRET
Enables *Bitbucket 身份验证*.

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 微软 *Azure Active Directory*.

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 微软 *Azure Active Directory*.

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 *SAML 身份验证*.

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.

参见:

Database setup for Weblate

POSTGRES_PASSWORD
PostgreSQL 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

Database backup settings

参见:

下载的数据用于备份

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.

参见:

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.

电子邮件服务器设置

要使外发电子邮件正常工作，您需要提供一个邮件服务器。

TLS 配置示例：

```
environment:
    WEBLATE_EMAIL_HOST: smtp.example.com
    WEBLATE_EMAIL_HOST_USER: user
    WEBLATE_EMAIL_HOST_PASSWORD: pass
```

SSL 配置示例：

```
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
```

参见:

Configuring outgoing e-mail

WEBLATE_EMAIL_HOST
邮件服务器主机名或 IP 地址。

参见:

WEBLATE_EMAIL_PORT, *WEBLATE_EMAIL_USE_SSL*, *WEBLATE_EMAIL_USE_TLS*, EMAIL_HOST

WEBLATE_EMAIL_PORT
邮件服务器端口，默认为 25。

参见:

EMAIL_PORT

WEBLATE_EMAIL_HOST_USER
电子邮件身份验证用户。

参见:

EMAIL_HOST_USER

WEBLATE_EMAIL_HOST_PASSWORD
电子邮件验证密码。

参见:

EMAIL_HOST_PASSWORD

WEBLATE_EMAIL_USE_SSL
与 SMTP 服务器通信时是否使用隐式 TLS（安全）连接。在大多数电子邮件文档中，这种 TLS 连接类型称为 SSL。通常在端口 465 上使用。如果遇到问题，请参阅显式 TLS 设置 *WEBLATE_EMAIL_USE_TLS*。

参见:

WEBLATE_EMAIL_PORT, *WEBLATE_EMAIL_USE_TLS*, EMAIL_USE_SSL

WEBLATE_EMAIL_USE_TLS
与 SMTP 服务器通讯时是否使用 TLS（安全）连接。这用于显式 TLS 连接，通常在端口 587 或 25 上。如果您遇到挂起的连接，请参见隐式 TLS 设置 *WEBLATE_EMAIL_USE_SSL*。

参见:

WEBLATE_EMAIL_PORT, *WEBLATE_EMAIL_USE_SSL*, EMAIL_USE_TLS

WEBLATE_EMAIL_BACKEND
将 Django 后端配置为用于发送电子邮件。

参见:

Configure e-mail sending, EMAIL_BACKEND

错误报告

建议系统地从安装中收集错误，请参阅收集错误报告。

要启用对 Rollbar 的支持，请进行以下设置：

ROLLBAR_KEY
您的 Rollbar 发布服务器访问令牌。

ROLLBAR_ENVIRONMENT
您的 Rollbar 环境，默认为 `production`。

要启用对 Sentry 的支持，请进行以下设置：

SENTRY_DSN
您的 Sentry DSN。

SENTRY_ENVIRONMENT
您的 Sentry 环境（可选）。

Localization CDN

WEBLATE_LOCALIZE_CDN_URL

WEBLATE_LOCALIZE_CDN_PATH

4.2.1 新版功能.

Configuration for *JavaScript 本地化 CDN*.

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
```

注解： You are responsible for seting up serving of the files generated by Weblate, it only does stores the files in configured location.

参见：

Translating HTML and JavaScript using Weblate CDN, *LOCALIZE_CDN_URL*, *LOCALIZE_CDN_PATH*

Changing enabled apps, checks, addons or autofixes

3.8-5 新版功能.

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

示例：

```
environment:
  WEBLATE_REMOVE_AUTOFIX: weblate.trans.autofixes.whitespace.
↪SameBookendingWhitespace
  WEBLATE_ADD_ADDONS: customize.addons.MyAddon,customize.addons.OtherAddon
```

参见：

CHECK_LIST, *AUTOFIX_LIST*, *WEBLATE_ADDONS*, INSTALLED_APPS

容器设置

CELERY_MAIN_OPTIONS

CELERY_NOTIFY_OPTIONS

CELERY_TRANSLATE_OPTIONS

CELERY_MEMORY_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 matches the number of processors (except the backup worker, which is supposed to run only once).

示例：

```
environment:
  CELERY_MAIN_OPTIONS: --concurrency 16
```

参见：

Celery worker options, *Background tasks using Celery*

UWSGI_WORKERS

Configure how many uWSGI workers should be executed.

It defaults to number of processors + 1.

示例：

```
environment:
  UWSGI_WORKERS: 32
```

Docker 容器卷

Weblate 容器导出单个数据卷。其他服务容器（PostgreSQL 或 Redis）也具有其数据卷，但本文档未涵盖这些数据卷。

数据卷用于存储 Weblate 持久数据（例如克隆的仓库）或自定义 Weblate 安装。

Docker 卷在主机系统上的位置取决于您的 Docker 配置，但通常存储在 `/var/lib/docker/volumes/weblate-docker_weblate-data/_data/` 中。在容器中，它挂载为 `/app/data`。

参见：

Docker 卷文档

进一步的配置定制

您可以在数据卷中进一步自定义 Weblate 安装，请参阅*Docker 容器卷*。

定制配置文件

您还可以在 `/app/data/settings-override.py` 中覆盖配置（请参阅*Docker 容器卷*）。加载所有环境设置后将执行此操作，因此将完全设置它，并可用于自定义任何内容。

替换 logo 和其他静态文件

3.8-5 新版功能.

Weblate 附带的静态文件可以通过放置到 `/app/data/python/customize/static` 中来覆盖（请参阅*Docker 容器卷*）。例如，创建 `/app/data/python/customize/static/favicon.ico` 将替换 favicon。

提示： 在容器启动时，这些文件被复制到相应的位置，因此需要在更改卷的内容后重新启动 Weblate 。

或者，您也可以包括自己的模块（请参阅*定制 Weblate*），并将其作为单独的卷添加到 Docker 容器中，例如：

```
weblate:
  volumes:
    - weblate-data:/app/data
    - ./weblate_customization/weblate_customization:/app/data/python/weblate_
↪customization
  environment:
    WEBLATE_ADD_APPS: weblate_customization
```

添加自己的 Python 模块

3.8-5 新版功能.

您可以将自己的 Python 模块放置在 `/app/data/python/` 中（请参阅*Docker 容器卷*），然后可以由 Weblate 加载它们，很可能是使用*定制配置文件*。

参见：

定制 Weblate

Hub 设置

为了使用 GitHub 的 pull-request 功能，您必须通过进入 Weblate 容器并执行任意 Hub 命令来初始化 Hub 配置。例如：

```
docker-compose exec --user weblate weblate bash
cd
HOME=/app/data/home hub clone octocat/Spoon-Knife
```

为凭据传递的用户名必须与*GITHUB_USERNAME* 相同。

参见：

GitHub, *Setting up hub*

Lab 设置

为了使用 GitLab 的 merge-request 功能，您必须通过进入 Weblate 容器并执行 `lab` 命令来初始化 `lab` 配置。例如：

```
docker-compose exec --user weblate weblate bash
cd
HOME=/app/data/home lab
```

您还可以使用环境变量在每个容器启动时配置 `lab`。只需将 `WEBLATE_GITLAB_USERNAME`，`WEBLATE_GITLAB_HOST` 和 `WEBLATE_GITLAB_TOKEN` 添加到您的环境配置中即可。

```
weblate:
  environment:
    WEBLATE_GITLAB_USERNAME: translations_bot
    WEBLATE_GITLAB_HOST: https://gitlab.example.com
    WEBLATE_GITLAB_TOKEN: personal_access_token_of_translations_bot
```

为 lab 配置传递的 `access_token` 必须与*GITLAB_USERNAME* 相同。

参见：

GitLab Setting up Lab

选择您的机器——本地或云提供商

使用 Docker Machine，您可以在本地计算机上或在任何数量的基于云的部署，例如 Amazon AWS，Greenhost 和许多其他提供商上创建 Weblate 部署。

Installing on Debian and Ubuntu

硬件要求

Weblate 应该可以在所有现代硬件上正常运行，以下是在单个主机（Weblate ，数据库和 Web 服务器）上运行 Weblate 所需的最低配置：

- 2 GB 的内存
- 2 个 CPU 核心
- 1 GB 的存储空间

内存越多越好——用于所有级别的缓存（文件系统，数据库和 Weblate ）。

许多并发用户会增加所需的 CPU 内核数量。对于数百个翻译组件，建议至少有 4 GB 的内存。

注解： 根据 Weblate 中管理的翻译大小，安装 Weblate 的实际要求差异很大。

安装

System requirements

Install the dependencies needed to build the Python modules (see 软件要求):

```
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 可选依赖性):

```
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

# Caching backend: Redis
apt install redis-server

# Database server: PostgreSQL
apt install postgresql postgresql-contrib

# SMTP server
apt install exim4
```

Python modules

提示： 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 可选依赖性):

```
pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
```

Configuring Weblate

注解: 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 can stick with shipped example for testing purposes, but you will want 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.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See 项目配置 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 支持的文件格式 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

硬件要求

Weblate 应该可以在所有现代硬件上正常运行，以下是在单个主机（Weblate ，数据库和 Web 服务器）上运行 Weblate 所需的最低配置：

- 2 GB 的内存
- 2 个 CPU 核心
- 1 GB 的存储空间

内存越多越好——用于所有级别的缓存（文件系统，数据库和 Weblate ）。

许多并发用户会增加所需的 CPU 内核数量。对于数百个翻译组件，建议至少有 4 GB 的内存。

注解： 根据 Weblate 中管理的翻译大小，安装 Weblate 的实际要求差异很大。

安装

System requirements

Install the dependencies needed to build the Python modules (see 软件要求):

```
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 可选依赖性):

```
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

提示： 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 可选依赖性):

   ```
   pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
   ```

Configuring Weblate

注解： 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 can stick with shipped example for testing purposes, but you will want 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.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See 项目配置 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 支持的文件格式 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

硬件要求

Weblate 应该可以在所有现代硬件上正常运行，以下是在单个主机（Weblate ，数据库和 Web 服务器）上运行 Weblate 所需的最低配置：

- 2 GB 的内存
- 2 个 CPU 核心
- 1 GB 的存储空间

内存越多越好——用于所有级别的缓存（文件系统，数据库和 Weblate ）。

许多并发用户会增加所需的 CPU 内核数量。对于数百个翻译组件，建议至少有 4 GB 的内存。

注解： 根据 Weblate 中管理的翻译大小，安装 Weblate 的实际要求差异很大。

安装

System requirements

Install the dependencies needed to build the Python modules (see 软件要求):

```
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 可选依赖性):

```
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

提示： 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 可选依赖性):

```
pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
```

Configuring Weblate

注解： 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 can stick with shipped example for testing purposes, but you will want 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.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See 项目配置 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 支持的文件格式 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

硬件要求

Weblate 应该可以在所有现代硬件上正常运行，以下是在单个主机（Weblate ，数据库和 Web 服务器）上运行 Weblate 所需的最低配置：

- 2 GB 的内存
- 2 个 CPU 核心
- 1 GB 的存储空间

内存越多越好——用于所有级别的缓存（文件系统，数据库和 Weblate ）。

许多并发用户会增加所需的 CPU 内核数量。对于数百个翻译组件，建议至少有 4 GB 的内存。

注解： 根据 Weblate 中管理的翻译大小，安装 Weblate 的实际要求差异很大。

安装

System requirements

Install the dependencies needed to build the Python modules (see 软件要求):

```
brew install pango libjpeg python git libyaml gobject-introspection
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 可选依赖性):

```
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

提示： 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 可选依赖性):

```
pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
```

Configuring Weblate

注解: 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 can stick with shipped example for testing purposes, but you will want 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.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See 项目配置 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 支持的文件格式 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 can stick with shipped example for testing purposes, but you will want 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
   ```

   注解： This step should be repeated whenever you update the repository.

Installing on OpenShift

注解： This guide is looking for contributors experienced with OpenShift, see <https://github.com/WeblateOrg/weblate/issues/2889>.

Weblate supports OpenShift, the needed integration files are in main repository in the `openshift3` directory.

根据您的设置和经验，为您选择适当的安装方法：

- 使用 *Docker* 安装，建议用于产品设置。
- Virtualenv 安装，建议用于产品设置：
  - *Installing on Debian and Ubuntu*
  - *Installing on SUSE and openSUSE*
  - *Installing on RedHat, Fedora and CentOS*
  - *Installing on macOS*
- *Installing from sources*，建议用于开发。
- *Installing on OpenShift* 。

2.1.2 软件要求

操作系统

Weblate 已知运行在 Linux 、FreeBSD 和 macOS 上。其他类 Unix 的系统也很可能支持运行。

Windows 不支持 Weblate 。但仍然可能工作，并愿意接受补丁。

其他服务

Weblate 为其操作使用其他服务。至少需要后面的服务运行：

- PostgreSQL 数据库服务器，请见*Database setup for Weblate* 。
- Redis 服务器，用于缓存和任务队列，请见*Background tasks using Celery* 。
- SMTP 服务器，用于发送电子邮件，请见*Configuring outgoing e-mail* 。

Python 依赖性

Weblate 用 Python 编写，并且支持 Python 3.6 或更新版本。您可以使用 pip 或您的发布包来安装依赖性，完全列表可在 `requirements.txt` 中找到。

最重要的依赖性：

Django https://www.djangoproject.com/

Celery https://docs.celeryproject.org/

Translate Toolkit （翻译工具包） https://toolkit.translatehouse.org/

translation-finder https://github.com/WeblateOrg/translation-finder

Python Social Auth https://python-social-auth.readthedocs.io/

Django REST 框架 https://www.django-rest-framework.org/

可选依赖性

后面的模块对 Weblate 的一些特性是必须的。可以在 `requirements-optional.txt` 中找到。

`Mercurial` （对 Mercurial 仓库支持是可选的） https://www.mercurial-scm.org/

`phply` （对 PHP 支持是可选的） https://github.com/viraptor/phply

`tesserocr` （对截屏 OCR 是可选的） https://github.com/sirfz/tesserocr

`akismet` （对建议垃圾邮件保护是可选的） https://github.com/ubernostrum/akismet

`ruamel.yaml` （对*YAML files* 是可选的） https://pypi.org/project/ruamel.yaml/

`Zeep` （对*Microsoft Terminology Service* 是可选的） https://docs.python-zeep.org/

`aeidon` （对*Subtitle files* 是可选的） https://pypi.org/project/aeidon/

数据库后端依赖性

Weblate 支持 PostgreSQL 、MySQL 和 MariaDB 数据库，更多细节请见*Database setup for Weblate* 和后端文件。

其他系统要求

后面的依赖性必须安装在系统上:

`Git` https://git-scm.com/

Pango 、Cairo 和相关的头文件与 gir introspection 数据 https://cairographics.org/ , https://pango.gnome.org/ ，请见*Pango 和 Cairo*

`hub` （将拉取请求发送给 GitHub 是可选的） https://hub.github.com/

`git-review` （对 Gerrit 支持是可选的） https://pypi.org/project/git-review/

`git-svn` （对 Subversion 支持是可选的） https://git-scm.com/docs/git-svn

`tesseract` 及其数据（对截屏 OCR 是可选的） https://github.com/tesseract-ocr/tesseract

编译时间依赖性

为了编译一些*Python 依赖性* ，会需要安装其依赖性。这依赖于如何安装它们，因此请每个独立包的文件。如果使用 `pip` 安装或使用发布包而使用预编译的 `Wheels` 时不会需要那些。

Pango 和 Cairo

在 3.7 版更改.

Weblate uses Pango and Cairo for rendering bitmap widgets (see *Promoting the translation*) and rendering checks (see 管理字型). 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 contains 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!
```

(下页继续)

(续上页)

```
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 Weblate.

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.

参见:

Serving static files

2.1.5 Database setup for Weblate

It is recommended to run Weblate with a PostgreSQL database server.

参见:

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.

注解: Weblate uses trigram extension which has to be installed separately in some cases. Look for `postgresql-contrib` or a similarly named package.

参见:

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 -O weblate weblate
```

提示: 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',
        # Database password
        'PASSWORD': 'password',
        # Set to empty string for localhost
        'HOST': 'database.example.com',
        # Set to empty string for default
        'PORT': '',
    }
}
```

MySQL and MariaDB

Weblate can be also used with MySQL or MariaDB, please see MySQL notes and MariaDB notes for caveats using Django with those.

提示： 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.

Because of this it is recommended to use *PostgreSQL* for new installations.

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`.

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_HOST_USER` and `EMAIL_PORT`. Their names are quite self-explanatory, but you can find more info in the Django documentation.

注解： 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).

Running behind reverse proxy

Several features in Weblate rely on being able to get client IP address. This includes *Rate limiting*, *Spam protection* or 审计日志.

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.

参见：

Spam protection, *Rate limiting*, 审计日志, *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"
```

参见:

Proxy Environment Variables

2.1.7 Adjusting configuration

参见:

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.

参见:

`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 = ['*']
```

参见:

`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'
```

参见:

Configuring the session engine, `SESSION_ENGINE`

`DATABASES`

Connectivity to database server, please check Django's documentation for more details.

参见:

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.

参见:

`DEBUG`

`DEFAULT_FROM_EMAIL`

E-mail sender address for outgoing e-mail, for example registration e-mails.

参见:

`DEFAULT_FROM_EMAIL`

`SECRET_KEY`

Key used by Django to sign some info in cookies, see *Django secret key* for more info.

参见:

`SECRET_KEY`

`SERVER_EMAIL`

E-mail used as sender address for sending e-mails to the administrator, for example notifications on failed merges.

参见:

`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.

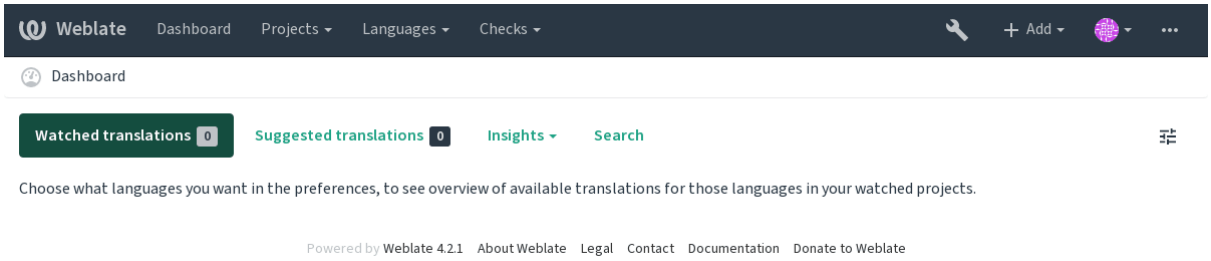
Once you are done, you should also check the *Performance report* in the admin interface, which will give you hints of potential non optimal configuration on your site.

参见:

配置, 访问控制

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
```

参见:

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.

参见:

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'),
)
```

参见:

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.

在 4.2 版更改: Prior to the 4.2 release the Django sites framework was used instead, please see The "sites"framework.

参见:

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
```

提示: You might want to set up HSTS as well, see SSL/HTTPS for more details.

参见:

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.

警告： 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.

参见：

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',
        }
    }
}
```

参见：

头像缓存, Django's cache framework

头像缓存

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',
        }
```

(下页继续)

(续上页)

```
    },
    'avatar': {
        'BACKEND': 'django.core.cache.backends.filebased.FileBasedCache',
        'LOCATION': os.path.join(DATA_DIR, 'avatar-cache'),
        'TIMEOUT': 604800,
        'OPTIONS': {
            'MAX_ENTRIES': 1000,
        },
    }
```

参见:

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'
```

注解: 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.

参见:

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.`

提示: On Docker container, this is available as *WEBLATE_ALLOWED_HOSTS*.

参见:

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.

参见:

SECRET_KEY

Home 目录

在 2.1 版更改: 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')
```

注解: 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.

参见:

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': [
                ('django.template.loaders.cached.Loader', [
                    'django.template.loaders.filesystem.Loader',
                    'django.template.loaders.app_directories.Loader',
                ]),
```

(下页继续)

(续上页)

```
            ],
        },
    },
]
```

参见:

`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 惰性提交 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*.

在 3.2 版更改: 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. For example when using Apache you might want to set it in `/etc/apache2/envvars`:

```
export LANG='en_US.UTF-8'
export LC_ALL='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
```

提示： The official Docker image has this feature already enabled.

参见：

Common Deployment Scenarios, *Serving static files*

2.1.10 Running server

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*)

注解： 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*.

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
```

警告： 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`.

提示： 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

在 2.4 版更改: 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`.

参见:

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.

参见:

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;
```

(下页继续)

(续上页)

```
        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/python3.7/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
```

(下页继续)

(续上页)

```
# 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
```

参见:

How to use Django with uWSGI

Sample configuration for Apache

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
    WSGIProcessGroup weblate
    WSGIApplicationGroup %{GLOBAL}
```

(下页继续)

(续上页)

```
    WSGIScriptAlias / /home/weblate/weblate-env/lib/python3.7/site-packages/
↪weblate/wsgi.py process-group=weblate
    WSGIPassAuthorization On

    <Directory /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/>
        <Files wsgi.py>
        Require all granted
        </Files>
    </Directory>

</VirtualHost>
```

注解： 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`.

参见：

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 !
```

(下页继续)

(续上页)

```
    ProxyPass /static/ !
    ProxyPass /media/ !

    ProxyPass / http://localhost:8000/
    ProxyPassReverse / http://localhost:8000/
    ProxyPreserveHost On
</VirtualHost>
```

参见:

How to use Django with Gunicorn

Running Weblate under path

在 1.3 版更改: This is supported since Weblate 1.3.

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
#
# 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
    WSGIProcessGroup weblate
    WSGIApplicationGroup %{GLOBAL}

    WSGIScriptAlias /weblate /home/weblate/weblate-env/lib/python3.7/site-packages/
↪weblate/wsgi.py process-group=weblate
    WSGIPassAuthorization On

    <Directory /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/>
        <Files wsgi.py>
        Require all granted
        </Files>
    </Directory>
```

(下页继续)

(续上页)

```
</VirtualHost>
```

Additionally, you will have to adjust `weblate/settings.py`:

```
URL_PREFIX = '/weblate'
```

2.1.11 Background tasks using Celery

3.2 新版功能.

Weblate uses Celery to process background tasks. The example settings come with eager configuration, which does process all tasks in place, but you want to change this to something more reasonable for a production setup.

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
```

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
```

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 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"

# Extra command-line arguments to the worker,
# increase concurency if you get weblate.E019
CELERYD_OPTS="--beat:celery --concurrency:celery=4 --queues:celery=celery --
↪prefetch-multiplier:celery=4 \
    --concurrency:notify=4 --queues:notify=notify --prefetch-multiplier:notify=10 \
    --concurrency:translate=4 --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="/var/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"
```

Logrotate configuration to be placed as `/etc/logrotate.d/celery`:

```
/var/log/celery/*.log {
        weekly
        missingok
        rotate 12
        compress
        notifempty
}
```

注解： The Celery process has to be executed under the same user as Weblate and the WSGI process, otherwise files in the *DATA_DIR* will be stored with mixed ownership, leading to runtime issues.

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 惰性提交.

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.

警告： 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 收集错误报告.

参见：

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.

2.1.13 收集错误报告

与其他任何软件一样，Weblate 可能会失败。为了收集有用的故障状态，我们建议使用第三方服务来收集此类信息。这在 Celery 任务失败的情况下尤其有用，否则将只会向日志报告错误，而您不会收到有关它们的通知。Weblate 支持以下服务：

Sentry

Weblate 内置了对 Sentry 的支持。要使用它，只需在 `settings.py` 中设置*SENTRY_DSN*：

```
SENTRY_DSN = "https://id@your.sentry.example.com/"
```

Rollbar

Weblate 具有对 Rollbar 的内置支持。要使用它，只需遵循 Rollbar notifier for Python 的说明即可。

简而言之，您需要调整 `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': 'master',
    'root': '/absolute/path/to/code/root',
}
```

其他所有内容都是自动集成的，您现在将同时收集服务器和客户端错误。

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 部署

Weblate 可以容易地安装到您的云中。请针对您的平台找到具体的指南：

- *使用 Docker 安装*
- *Installing on OpenShift*

2.2.1 Helm Chart

可以使用 Helm 将 Weblate 安装到 Kubernetes 上。具体的手册请见 <https://github.com/WeblateOrg/helm/tree/master/charts/weblate> 。

2.2.2 Bitnami Weblate 栈

Bitnami 为很多平台提供 Weblate 栈 <https://bitnami.com/stack/weblate> 。设置在安装过程中调整，更多文件请见 <https://bitnami.com/stack/weblate/README.txt> 。

2.2.3 YunoHost 中的 Weblate

自托管项目 YunoHost 为 Weblate 提供了包。一旦安装了 YunoHost，就可以同其它应用一样安装 Weblate。它还为你提供带有备份和恢复的完全工作栈，但您必须为特定应用编辑设置文件。

您可以使用管理界面，或这个按钮（它将带您到您的服务器）：

还能够使用命令行界面：

```
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 *使用 Docker 安装*) 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 *软件要求* 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.

注解： 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 *备份和移动 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 the continue upgrading in the 3.x series. Upgrades skipping this step are not supported and will break.

参见:

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 and the continue upgrading in the 4.x series. Upgrades skipping this step are not supported and will break.

参见:

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 *DeepL* machine translation now defaults to v2 API, you might need to adjust *MT_DEEPL_API_VERSION* in case your current DeepL subscription does not support that.

参见:

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.

参见:

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 migrate to PostgreSQL as that will be the only supported database backend in the 4.0 release. 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 -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 PotsgreSQL using pgloader*.

1. Add PostgeSQL 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
        '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 PotsgreSQL 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 PostgeSQL 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

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 备份和移动 Weblate

2.4.1 自动备份

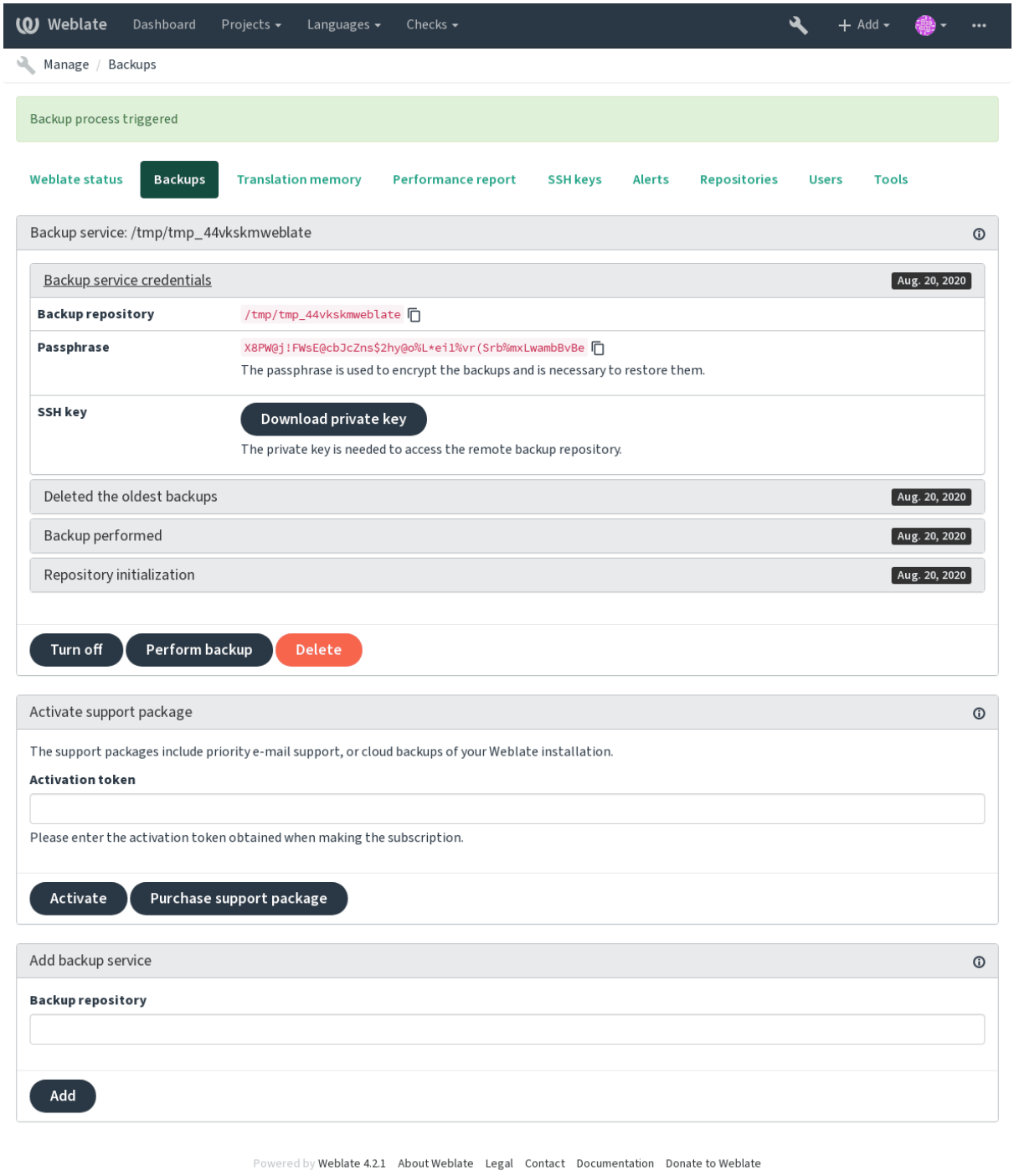
3.9 新版功能.

Weblate 内置了对使用 BorgBackup 创建服务备份的支持。Borg 创建了节省空间的加密备份，可以安全地存储在云中。可以在管理界面中的 *Backups* 选项卡上控制备份。

警告： 自动备份仅包含 PostgreSQL 数据库。其他数据库引擎必须手动备份。建议您迁移到 PostgreSQL，请见*Database setup for Weblate* 和*Migrating from other databases to PostgreSQL* 。

使用 Borg 的备份是递增的，Weblate 配置为保留后面的备份：

- 14 个每天备份
- 8 个每周备份
- 6 个每月备份



使用 Weblate 配置的备份存储

备份 Weblate 事例最简单的方法是购买 backup service at weblate.org 。激活过程可以分几步来执行:

1. 在 https://weblate.org/support/#backup 上购买备份服务。
2. 在管理界面输入得到的密钥，请见*Integrating support* 。
3. Weblate 将连接到云服务，并得到访问信息来备份。
4. 在 *Backups* 标签打开新的备份配置。
5. 备份 Borg 凭据，以便能够恢复备份，请见*Borg 加密密钥* 。

提示： 为了安全起见，有打开的手动步骤。没有你的同意，就不会有数据发送到通过注册步骤得到的备份仓库。

使用客户的备份存储

也可以使用自己的存储来备份。SSH 可以用于在远程目的地存储备份，目标服务器需要安装 BorgBackup 。

参见：

General 在 Borg 文件中

Borg 加密密钥

BorgBackup 生成加密的备份，没有密码的话就不能恢复备份。当添加备份服务时产生密码，并且应该将其复制并保存在安全的地方。

在使用使用 *Weblate* 配置的备份存储 的情况下，请同样备份私有 SSH 密钥——它用于访问你的备份。

参见：

borg init

从 BorgBackup 恢复

1. 恢复功能会访问你的备份仓库，并准备备份密码。
2. 使用 `borg list REPOSITORY` 列出服务器上存在的备份。
3. 使用 `borg extract REPOSITORY::ARCHIVE` 将所需备份恢复到当前目录。
4. 从放置在 Weblate 数据目录下 `backup` 目录中的 SQL 备份中恢复数据库（请见下载的数据用于备份）。
5. 将 Weblate 配置（`backups/settings.py`，请见下载的数据用于备份 ）复制到正确的位置，请见*Adjusting configuration* 。
6. 将整个存储的数据目录复制到在*DATA_DIR* 中配置的位置。

Borg 会话会是这个样子的：

```
$ 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:
```

参见：

borg list, borg extract

2.4.2 手动备份

依赖于您想存储什么，Weblate 存储的类型数据备份在各自的位置。

提示: 在进行手动备份时，会想要通过将 `weblate.I028` 添加到 `settings.py` 的 `SILENCED_SYSTEM_CHECKS` 中，或 Docker 的 *`WEBLATE_SILENCED_SYSTEM_CHECKS`* 中，来关闭 Weblate 的缺少备份警告。

```
SILENCED_SYSTEM_CHECKS.append("weblate.I028")
```

数据库

实际存储位置依赖于数据库的设置。

数据库是最重要的存储。要设置数据库的常规备份，没有的话翻译设置就丢失了。

本地数据库备份

建议的方式是使用数据库的本地工具如 **`pg_dump`** 或 **`mysqldump`** 来备份数据库。这通常比 Django 备份执行得好，并且恢复所有数据的完整表格。

可以在更新版的 Weblate 中恢复备份，当运行 `migrate` 时执行任何必要的迁移。如何在两个版本之间执行升级的更多细节信息请咨询*Upgrading Weblate* 。

Django 数据库备份

另外，可以使用 Django 的 `dumpdata` 命令备份数据库。那种方式是不依托数据库的，并且可以用于先要改变数据库后端的情况。

在恢复前，需要运行与进行备份使用的 Weblate 完全一致的版本。这是必须的，因为数据库结构在发布版本之间有变化，并且会导致某种方式的数据损坏。在安装相同版本后，使用 `migrate` 来运行所有的数据库迁移。

一旦完成，数据库中就已经建立了一些入口，并且同样建立在数据库备份中。建议的方法是使用管理 shell （请见*Invoking management commands* ）手动删除这样的入口：

```
weblate shell
>>> from weblate.auth.models import User
>>> User.objects.get(username='anonymous').delete()
```

文件

如果有充足的备份空间，则简单地备份整个*`DATA_DIR`* 。即使包括一些不想要的文件，这样做也是安全的。后面的部分具体描述了什么应该备份，什么应该跳过。

下载的数据用于备份

存储在*DATA_DIR* /backups 中。

Weblate 这里备份各种数据，可以包括这些文件用于更完整的备份。文件每日更新（需要运行 Celery beat 服务器，请见*Background tasks using Celery* ）。当前，这包括：

- Weblate 设置为 settings.py （还有扩展版，在 settings-expanded.py ）。
- PostgreSQL 数据库备份为 database.sql 。

数据库备份默认存储为纯文本，但也可以通过使用:setting:‘DATABASE_BACKUP‘来压缩的或整个跳过。

版本控制仓库

存储在*DATA_DIR* “/vcs“中。

版本控制仓库包含了带有 Weblate 更改的上游仓库的复制备份。如果对所有翻译组件推进了同意允许，那么所有 Weblate 更改都包括在上游中，并且不必备份 Weblate 侧的仓库。它们可以从上游位置再次克隆，而不会丢失数据。

SSH 和 GPG 密钥

存储在*DATA_DIR* /ssh 和*DATA_DIR* /home 中。

如果您正在使用 Weblate 生成的 SSH 或 GPG 密钥，您应该备份这些位置，否则您将丢失私有密钥，并且您将不得不重新生成新的密钥。

用户上传的文件

存储在*DATA_DIR* /media 中。

可以备份用户上传的文件（例如*Visual context for strings* ）。

手动备份的命令行

使用 crom 任务，可以设置批处理命令，以每天为单位来执行，例如：

```
$ XZ_OPT="-9" tar -Jcf ~/backup/weblate-backup-$(date -u +%Y-%m-%d_%H%M%S).xz␣
↪backups vcs ssh home media fonts secret
```

XZ_OPT 后面引号之间的字符串允许您选择自己的 xz 选项，例如用于压缩的内存量；请见 https://linux.die.net/man/1/xz

可以根据需要调整文件夹和文件的列表。例如，为了节省翻译内存（在备份文件夹中），可以使用：

```
$ 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
```

2.4.3 Celery 任务

Celery 任务队列会包含一些信息，但备份通常不需要它。最多就是丢失了翻译内存还没有处理的更新。建议恢复时随便执行全文或仓库更新，这样不会有丢失的问题。

参见：

Background tasks using Celery

2.4.4 恢复手动备份

1. 将已经备份的所有数据恢复。
2. 使用`updategit`更新所有仓库。

```
weblate updategit --all
```

2.4.5 移动 Weblate 安装

按照上面备份与恢复的说明，将安装重定位到不同系统。

参见：

Upgrading from Python 2 to Python 3, *Migrating from other databases to PostgreSQL*

2.5 身份验证

2.5.1 注册用户

Weblate 的默认设置使用 python-social-auth，网站上处理新用户注册的一种形式。确定电子邮箱后，新用户可以通过使用一种第三方服务来贡献或证实。

还可以使用`REGISTRATION_OPEN`关闭新用户注册。

身份验证尝试服从于*Rate limiting*。

2.5.2 身份验证后台

Django 的内置解决方案用途是进行身份验证，包括用各种社交登录选项进行验证。使用它意味着可以导入基于 Django 其他项目的用户数据库（请见*Migrating from Pootle*）。

也可以另外设置 Django，相对于其他方式进行身份验证。

参见：

Authentication settings 描述了如何配置官方 Docker 镜像的身份验证。

2.5.3 社交身份验证

由于 Welcome to Python Social Auth's documentation!，Weblate 支持很多使用第三方服务的身份验证，如 GitLab 、Ubuntu 、Fedora 等。

请检查 Django Framework 中的通用配置指示的文件。

注解： Weblate 默认依赖于第三方身份验证服务来提供合法的电子邮箱地址。如果想要使用的一些服务不支持，请通过为其配置 FORCE_EMAIL_VALIDATION ，来强制 Weblate 网站上的电子邮箱验证：

```
SOCIAL_AUTH_OPENSUSE_FORCE_EMAIL_VALIDATION = True
```

参见：

Pipeline

允许各自的后端非常容易，只是添加 AUTHENTICATION_BACKENDS 入口设置，并且可能为给定的身份验证方法添加所需的密钥。请注意，一些后端不默认提供用户邮箱，必须另外请求，否则 Weblate 不能确认用户做出的贡献。

参见：

Python Social Auth backend

OpenID 身份验证

对于基于 OpenID 的服务，通常只要启用它们就行了。后面的部分关于对于 OpenSUSE 、Fedora 和 Ubuntu 允许 OpenID 身份验证：

```
# 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',
)
```

参见：

OpenID

GitHub 身份验证

需要在 GitHub 上注册应用，然后告诉 Weblate 所有的秘密：

```
# 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']
```

应该配置 GitHub 具有回调 URL 作为 `https://example.com/accounts/complete/github/` 。

注解： Weblate 在身份验证时提供的回调 URL 。在得到 URL 不匹配的错误时，可以根据需要来修改，请见*Set correct site domain* 。

参见：

GitHub

Bitbucket 身份验证

需要在 Bitbucket 上注册应用，然后告诉 Weblate 所有的秘密：

```
# 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
```

注解： Weblate 在身份验证时提供的回调 URL 。在得到 URL 不匹配的错误时，可以根据需要来修改，请见*Set correct site domain* 。

参见：

Bitbucket

Google OAuth 2

为了使用 Google OAuth 2 ， 可以在 <https://console.developers.google.com/> 上注册应用，并允许 Google+ API 。

重定向 URL 为 `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'
```

注解： Weblate 在身份验证时提供的回调 URL 。在得到 URL 不匹配的错误时，可以根据需要来修改，请见*Set correct site domain* 。

参见：

Google

Facebook OAuth 2

通常根据 OAuth2 服务，需要注册 Facebook 应用。一旦完成，就可以设置 Weblate 来使用了：

重定向 URL 为 `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']
```

注解： Weblate 在身份验证时提供的回调 URL 。在得到 URL 不匹配的错误时，可以根据需要来修改，请见*Set correct site domain* 。

参见：

Facebook

GitLab OAuth 2

为了使用 GitLab OAuth 2 ，需要在 <https://gitlab.com/profile/applications> 上注册应用。

重定向 URL 为 `https://WEBLATE SERVER/accounts/complete/gitlab/` ，并确保你标记 *read_user* 范围。

```
# 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/'
```

注解： Weblate 在身份验证时提供的回调 URL 。在得到 URL 不匹配的错误时，可以根据需要来修改，请见*Set correct site domain* 。

参见：

GitLab

微软 Azure Active Directory

可以配置 Weblate，使用一般或特定租户进行身份验证。

常见的重定向 URL 为 `https://WEBLATE SERVER/accounts/complete/azuread-oauth2/` ，`https://WEBLATE SERVER/accounts/complete/azuread-tenant-oauth2/` 用于租户特定身份验证。

```
# 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
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY = ""
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECRET = ""
# Tenant ID
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENANT_ID = ""
```

注解： Weblate 在身份验证时提供的回调 URL 。在得到 URL 不匹配的错误时，可以根据需要来修改，请见*Set correct site domain* 。

参见：

Microsoft Azure Active Directory

Slack

为了使用 Slack OAuth 2 ，需要在 <https://api.slack.com/apps> 上注册应用。

重定向 URL 为 `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 = ''
```

注解： Weblate 在身份验证时提供的回调 URL 。在得到 URL 不匹配的错误时，可以根据需要来修改，请见*Set correct site domain* 。

参见：

Slack

关闭密码身份验证

通过从 `AUTHENTICATION_BACKENDS` 删除 `social_core.backends.email.EmailAuth` ，可以关闭电子邮箱和密码身份验证。总是将 `weblate.accounts.auth.WeblateUserBackend` 保留在那里，它用于 Weblate 核心功能。

小技巧： 对于手动建立的用户，可以仍然在管理界面使用密码身份验证。只需导航到 `/admin/` 。

例如，使用后面的设置可以实现只是用 openSUSE Open ID 的身份验证：

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.suse.OpenSUSEOpenId',
    'weblate.accounts.auth.WeblateUserBackend',
)
```

2.5.4 密码身份验证

默认 `settings.py` 与一组合理的设置 `AUTH_PASSWORD_VALIDATORS` 在一起：

- 密码不能与其它个人信息太相似。
- 密码必须包含 10 个字符。
- 密码不能是通常使用的密码。
- 密码不能完全是数字。
- 密码不能包括单个字符或只有空格。
- 密码与您过去使用的密码不匹配。

可以自定义这个设置来匹配密码政策。

可以另外安装 django-zxcvbn-password 这会非常实际地估计密码的难度，并允许拒绝低于下面适当阈值的密码。

2.5.5 SAML 身份验证

4.1.1 新版功能.

请遵守 Python Social Auth 的指示来配置。显著的差异有：

- Weblate 支持单一 IDP ，在 `SOCIAL_AUTH_SAML_ENABLED_IDPS` 中被称为 `weblate` 。
- SAML XML 元数据 URL 为 `/accounts/metadata/saml/` 。
- 后面的设置自动填入：`SOCIAL_AUTH_SAML_SP_ENTITY_ID` 、`SOCIAL_AUTH_SAML_TECHNICAL_CONTACT` 、`SOCIAL_AUTH_SAML_SUPPORT_CONTACT`

配置的例子：

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.email.EmailAuth",
    "social_core.backends.saml.SAMLAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Social auth backends setup
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",
    }
}
```

参见:

Configuring SAML in Docker, SAML

2.5.6 LDAP 身份验证

LDAP 身份验证可以使用 *django-auth-ldap* 软件包而最好地实现。可以使用通常的方式安装:

```
# Using PyPI
pip install django-auth-ldap>=1.3.0

# Using apt-get
apt-get install python-django-auth-ldap
```

警告: 早于 1.3.0 版的 django-auth-ldap，自动分配组 对新建立的用户无法正常工作。

注解: 在 Python LDAP 3.1.0 模块中有一些不兼容，导致可能无法使用那个版本。如果得到错误信息 AttributeError: ‘module’ object has no attribute ‘_trace_level’ ，将 python-ldap 降回到 3.0.0 版可能会有帮助。

一旦安装了软件包，就可以钩入 Django 身份验证了:

```
# Add LDAP backed, keep Django one if you want to be able to login
# 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
```

(下页继续)

(续上页)

```
# like:
# AUTH_LDAP_USER_DN_TEMPLATE = 'ou=users,dc=example,dc=com'

# List of attributes to import from LDAP upon login
# 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
```

注解: 应该从 AUTHENTICATION_BACKENDS 设置中删除 'social_core.backends.email.EmailAuth'，否则用户将能够在 Weblate 中设置自己的密码，并且用来进行身份验证。仍然需要保留 'weblate.accounts.auth.WeblateUserBackend' 来生成权限，并实现匿名用户。这也会让你使用本地管理账户来登录，如果有的话（例如通过使用*createadmin*）。

使用绑定密码

如果可以为身份验证使用直接绑定，那么需要使用搜索，并为用户搜索提供绑定，例如：

```
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)")
```

活动目录集成

```
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)")
```

(下页继续)

(续上页)

```
AUTH_LDAP_GROUP_TYPE = NestedActiveDirectoryGroupType()
AUTH_LDAP_FIND_GROUP_PERMS = True

# Optionally enable group mirroring from LDAP to Weblate
# AUTH_LDAP_MIRROR_GROUPS = True
```

参见:

Django Authentication Using LDAP, Authentication

2.5.7 CAS 身份验证

可以使用软件包如 *django-cas-ng* 来实现 CAS 身份验证。

第一步通过 CAS 揭示了用户的电子邮箱域。这必须在 CAS 服务器自身来配置，并需要至少运行 CAS v2，因为 CAS v1 不支持属性。

第二步更新 Weblate，来使用 CAS 服务器和属性。

为了安装 *django-cas-ng*：

```
pip install django-cas-ng
```

一旦安装了软件包，就可以通过修改 `settings.py` 文件将其钩到 Django 身份验证系统：

```
# 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'
)
```

最后，可以使用信号将电子邮箱域投射到用户对象上。为了生效，必须将信号从 *django-cas-ng* 软件包导入，并将你的代码与这个信号连接。在设置文件中这样做可能产生问题，这样建议将它放进去：

- 在你的 app 配置的 `django.apps.AppConfig.ready()` 方法
- 在项目的 `urls.py` 文件中（当没有模块存在时）

```
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()
```

参见:

Django CAS NG

2.5.8 配置第三方 Django 身份验证

一般地，任何 Django 身份认证插件应该可以在 Weblate 上工作。只需要按照插件的说明，只记住安装了 Weblate 用户后台。

参见:

LDAP 身份验证, *CAS 身份验证*

典型的安装包括，将身份验证后台添加到 `AUTHENTICATION_BACKENDS`，并将身份验证 app （如果有的话）安装到 `INSTALLED_APPS`：

```
AUTHENTICATION_BACKENDS = (
    # Add authentication backend here
    'weblate.accounts.auth.WeblateUserBackend',
)

INSTALLED_APPS = (
    ...
    'weblate',
    # Install authentication app here
)
```

2.6 访问控制

在 3.0 版更改: 在 Weblate 3.0 之前，特权系统基于 Django，但现在是专门为 Weblate 构建的。如果您使用的是旧版本，请查阅该版本的文档，此处的信息将不适用。

Weblate 带有细粒度的特权系统，可以为整个实例或在有限范围内分配用户权限。

基于组和角色的权限系统，其中角色定义了一组权限，组将它们分配给用户和翻译，请参阅*用户，角色，用户组和权限* 以获取更多详细信息。

安装后，将创建一组默认的组，您可以使用这些组为整个实例分配用户角色（请参阅*默认群组和角色*）。此外，启用*根据项目的访问控制* 后，您可以将用户分配给特定的翻译项目。使用*客户访问控制* 可以实现更细粒度的配置

2.6.1 通用设置

锁定 Weblate

为了完全锁定 Weblate 安装，可以使用*LOGIN_REQUIRED_URLS* 来强制用户登录并*REGISTRATION_OPEN* 来防止新注册。

全网站范围的权限

为了管理整个事例的权限，只通过将用户添加到 *Users*（这通常使用*自动分配组* 默认实现）、*Reviewers* 和 *Managers* 群组中即可。将所有的项目保持为 *Public*（请参考*根据项目的访问控制*）。

各项目的权限

将项目设置为 *Protected* 或 *Private* ，并在 Weblate 的界面上根据项目来管理用户。

为语言、组件或项目添加语言

可以根据项目、组件或语言向任何用户另外授予权限。为了实现这一目的，对给定的资源建立新的群组（例如“捷克语译者”）并配置。对于所选的资源，任何指定的权限可以授予群组内的成员。

如果根据项目权限使用的话，无需另外设置即可正常工作。对于整个事例的权限，您可能还想从 *Users* 群组中去掉这些权限，或者改变将所有用户自动指定给那个群组（请参考自动分配组 ）。

参见：

权限检查

2.6.2 根据项目的访问控制

注解： 通过允许 ACL ，所有用户禁止访问给定项目的任何内容，除非可以将权限授予他们去做。

可以限制用户访问独立的项目。这个特性通过每个单独项目的配置中 *Access control* 来打开。这会为这个项目自动建立几个群组，请参考预定义的群组 。

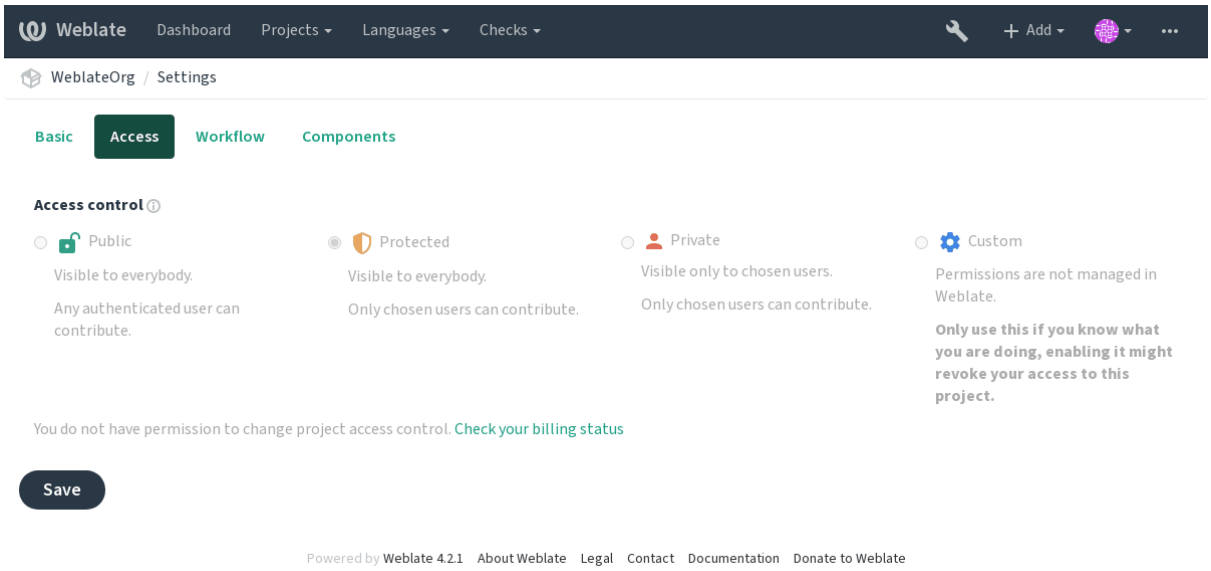
后面的选择是为了 *Access control* ：

公开的 公开可见、可翻译

受保护的 公开可见，但只允许被选择的用户翻译

私有的 公开可见，但只允许被选择的用户翻译

自定义 Weblate 不允许管理用户，请参考客户访问控制 。



为了允许访问这个项目，必须将权限直接添加给特定用户，或在 Django 管理界面上添加给用户所在的群组，或者使用项目页面上的用户管理，如下面的描述根据项目访问控制来管理 。

注解： 即使打开 ACL ，也可以看到项目的一些概要信息：

- 整个事例的统计，包括所有项目计数。
- 整个事例的语言概要，包括所有项目的计数。

2.6.3 自动分配组

可以设置 Weblate 根据用户的邮件地址自动将用户添加到用户组中。只有在建立账户时这一自动指定才进行。

可以对每个用户组在 Django 管理界面中设置（在 *Authentication* 部分）。

注解： 对于 *Users* 和 *Viewers* 用户组的自动用户组指定，总是通过 Weblate 在合并时建立，在想将其关闭的时候，简单地设置正则表达式为 `^$` 即可，因为永远不匹配。

2.6.4 用户，角色，用户组和权限

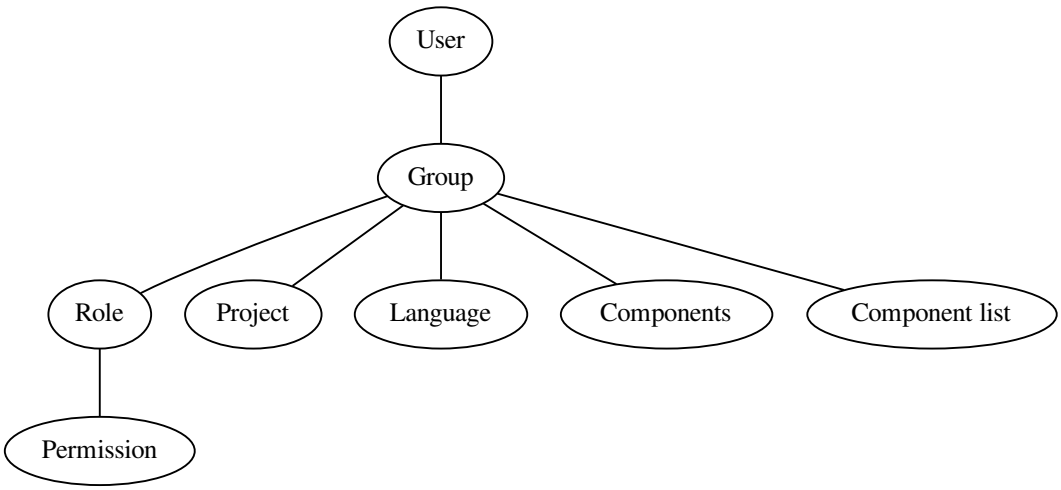
身份验证模型包括几个对象：

权限 Weblate 定义的各自权限。可以不指定各自权限，这可以只通过角色指定实现。

Role 角色定义为一组权限。这能够在几个地方重复使用这些组，并使管理更容易。

User 用户可以使几个用户组的成员。

群组 用户组与角色、用户和身份验证对象（项目、语言和组件列表）联系。



权限检查

任何时候检查权限来决定是否执行给定的动作时，根据范围来执行检查，并且后面的检查依次执行：

1. *Component list* 与组件或项目匹配。
2. *Component* 与组件或项目匹配。
3. *Projects* 与项目匹配。

可以看到，为组件授予访问权限同样也会自动地授予用户所包括项目的访问权限。

注解： 只使用第一条规则。所以如果设置所有的 *Component list* 、*Components* 和 *Project* , 只会应用 *Component list* 。

如果检查翻译许可，则会执行另外的步骤：

4. *Languages* are matched against the scope of translations if set, if not set, this does not match any language.

提示： 可以使用 *Language selection* 或 *Project selection* 来自动包括所有语言或项目。

检查项目的访问权限

用于可以成为与项目或其中任何组件相连接的用户组的成员。只有成员即可，不需要特别许可来访问项目（这在默认的 *Viewers* 用户组中使用，请见默认群组和角色 ）。

检查对组件的访问

一旦用户可以访问包含的项目，就能访问不受限制的组件。通过允许*Restricted access* ，访问组件需要组件（或包含组件列表）的明确授权。

2.6.5 管理用户和群组

可以使用 `/admin/` URL 下面的 Django 管理界面，来管理所有用户和用户组。

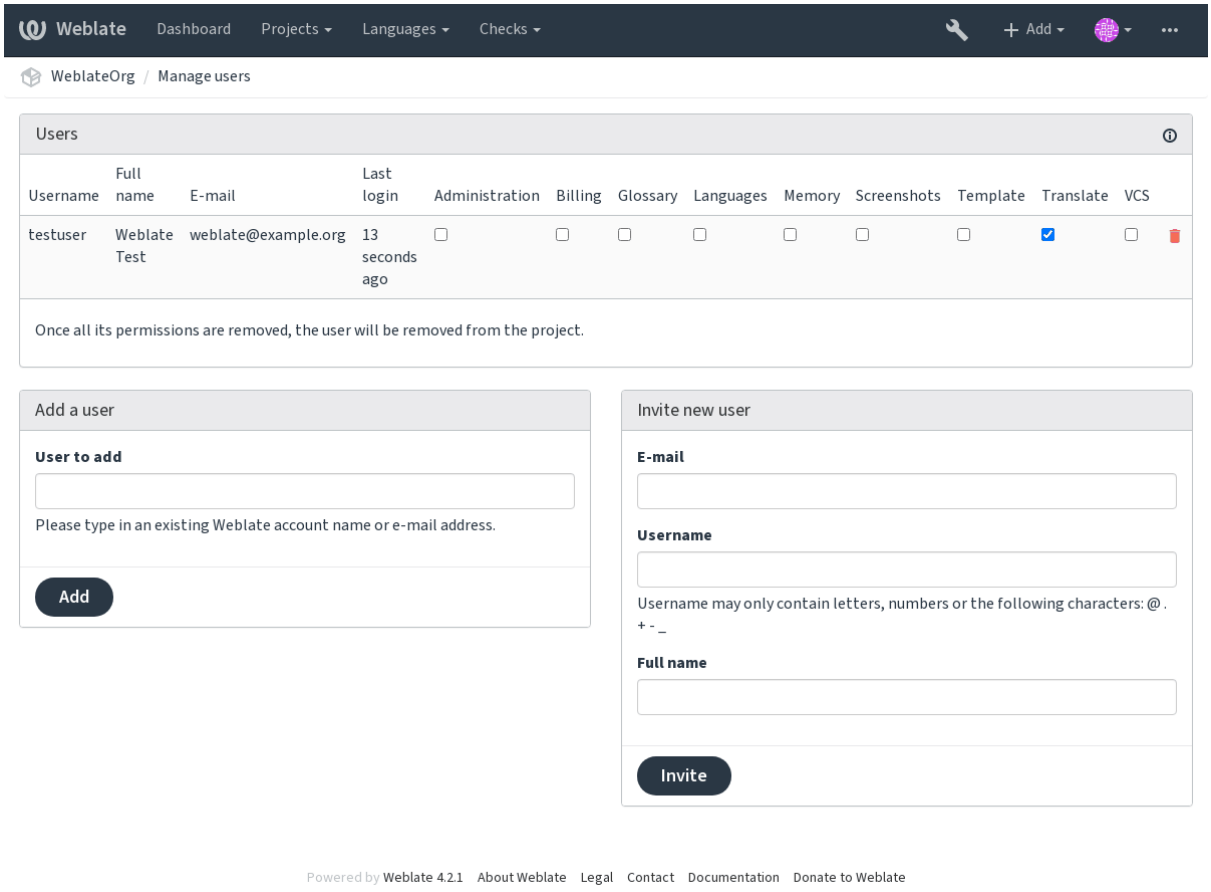
根据项目访问控制来管理

注解： 这个特性只对 ACL 控制的项目起作用，请见根据项目的访问控制 。

具有 *Can manage ACL rules for a project* 特权的用户（请见访问控制 ）还可以通过项目页面上的打开访问控制来管理用户。这个界面允许您：

- 将现有用户添加到项目中
- 邀请新用户参加到项目中
- 更改用户的权限
- 取消用户的权限

可以在项目菜单的 *Manage* 进行用户管理：



参见:

根据项目的访问控制

预定义的群组

Weblate 的项目带有预定义的群组，您可以为之指定用户。

Administration
在项目中可以有所有的权限。

Glossary
可以管理词汇表（添加或删除权限，或上传）。

Languages
可以管理翻译语言——添加或删除翻译。

Screenshots
可以管理截屏——添加或删除截屏，并将其与源字符串联系起来。

Template
可以在单语言组件 编辑翻译模板和源字符串信息。

Translate
可以翻译项目，并将离线的翻译上传。

VCS
可以管理 VCS 版本控制系统并访问导出的仓库。

Review
可以在复查时批准翻译。

Billing
可以访问账单信息（请见账单）。

2.6.6 客户访问控制

通过选择 *Custom* 作为 *Access control*，Weblate 会对停止管理给定项目的访问权限，使用 Django 管理界面可以管理所有用户和群组。这可以用于确定更富在的访问控制，或在单一的 Weblate 界面中对所有项目设置可分享的访问策略。如果想要对所有项目默认打开这个功能，请配置*DEFAULT_ACCESS_CONTROL*。

警告： 通过将其打开，Weblate 会删除所有为这个项目建立的根据项目的访问控制。如果没有事件的管理权限却去做的话，会立即丢失管理项目的访问权限。

2.6.7 默认群组和角色

特权列表

账单（请见账单） 查看账单信息 [*Administration*, *Billing*]

修改 下载更改 [*Administration*]

注释 发表评论 [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

删除评论 [*Administration*]

组件 编辑组件设置 [*Administration*]

锁定组件，防止被翻译 [*Administration*]

词汇表 添加词汇表入口 [*Administration*, *Manage glossary*, *Power user*]

编辑词汇表入口 [*Administration*, *Manage glossary*, *Power user*]

删除词汇表入口 [*Administration*, *Manage glossary*, *Power user*]

上传词汇表入口 [*Administration*, *Manage glossary*, *Power user*]

机器翻译 使用机器翻译 [*Administration*, *Power user*]

项目 编辑项目设置 [*Administration*]

更改项目访问权限 [*Administration*]

报告 下载报告 [*Administration*]

截图 添加截屏 [*Administration*, *Manage screenshots*]

编辑截屏 [*Administration*, *Manage screenshots*]

删除截屏 [*Administration*, *Manage screenshots*]

源字符串 编辑源字符串信息 [*Administration*, *Edit source*]

字符串 添加新字符串 [管理组]

忽略失败的复查 [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

编辑字符串 [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

复查字符串 [*Administration*, *Review strings*]

当建议被强制执行时需要编辑字符串 [*Administration*, *Review strings*]

编辑源字符串 [*Administration*, *Edit source*, *Power user*]

建议 接受建议 [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

添加建议 [*Add suggestion*, *Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

删除建议 [*Administration*]

为建议投票 [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

翻译 新建翻译 [*Administration*, *Manage languages*, *Power user*]

执行自动翻译 [*Administration*, *Manage languages*]

删除现有的翻译 [*Administration*, *Manage languages*]

开始另一门语言的翻译 [*Administration*, *Manage languages*]

上传 定义翻译上传的作者 [*Administration*]

使用上传的内容覆盖现有的字符串 [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

上传翻译的字符串 [*Administration*, *Edit source*, *Power user*, *Review strings*, *Translate*]

版本控制系统（VCS） 访问内部仓库 [*Access repository*, *Administration*, *Manage repository*, *Power user*]

将更改提交到内部代码库 [*Administration*, *Manage repository*]

从内部代码库推送更改 [*Administration*, *Manage repository*]

重置内部代码库的更改 [*Administration*, *Manage repository*]

查看上游仓库位置 [*Access repository*, *Administration*, *Manage repository*, *Power user*]

更新内部代码库 [*Administration*, *Manage repository*]

全网站范围的特权 使用管理界面

添加语言定义

管理语言定义

添加群组

管理群组

添加用户

管理用户

管理公告

管理翻译记忆库

注解： 全网站特权不会授予任何默认角色。这些特权特别强大，非常接近超级用户状态——多数特权会影响 Weblate 安装的所有项目。

群组列表

下面的群组在安装时建立（或在执行后`setupgroups`）：

访客 对非授权用户确定权限。

这个群组只包括匿名用户（请见`ANONYMOUS_USER_NAME`）。

可以从群组中去掉角色，来限制非授权用户的权限。

默认角色: *Add suggestion*, *Access repository*

Viewers 这一角色确保公开项目对所有用户可见。所有用户默认是这个群组的成员。

所有用户默认为这个群组的成员，使用`自动分配组`。

默认角色：无

用户 所有用户的默认群组。

所有用户默认为这个群组的成员，使用自动分配组 。

默认角色: *Power user*

校对 复核员的群组（参见翻译工作流 ）。

默认角色: *Review strings*

管理人员 管理员的群组。

默认角色: *Administration*

警告: 永远不要删除预先定义的 Weblate 群组和用户，因为这会导致意外的错误。如果不想要使用这些特性，只去掉他们的特权即可。

2.7 翻译项目

2.7.1 翻译组织

Weblate 将项目/组件的可翻译 VCS 内容组织成树状结构。

- 底层对象是项目配置，该项目配置应将所有翻译归在一起（例如，多个版本的应用程序翻译和/或随附的文档）。
- 在上面的级别上，:ref:‘component‘（实际上是要翻译的组件），您定义要使用的 VCS 仓库以及要翻译的文件的掩码。
- 在*Component configuration* 上方，有单独的翻译，当 VCS 仓库中出现翻译文件（与*Component configuration* 中定义的掩码匹配）时，Weblate 会自动处理这些翻译。

Weblate 支持 Translate Toolkit 支持的多种翻译格式（双语和单语），请参阅支持的文件格式。

注解: 您可以使用*Weblate internal URLs* 共享克隆的 VCS 仓库。当您有许多共享同一 VCS 的组件时，强烈建议使用此功能。它提高了性能并减少了所需的磁盘空间。

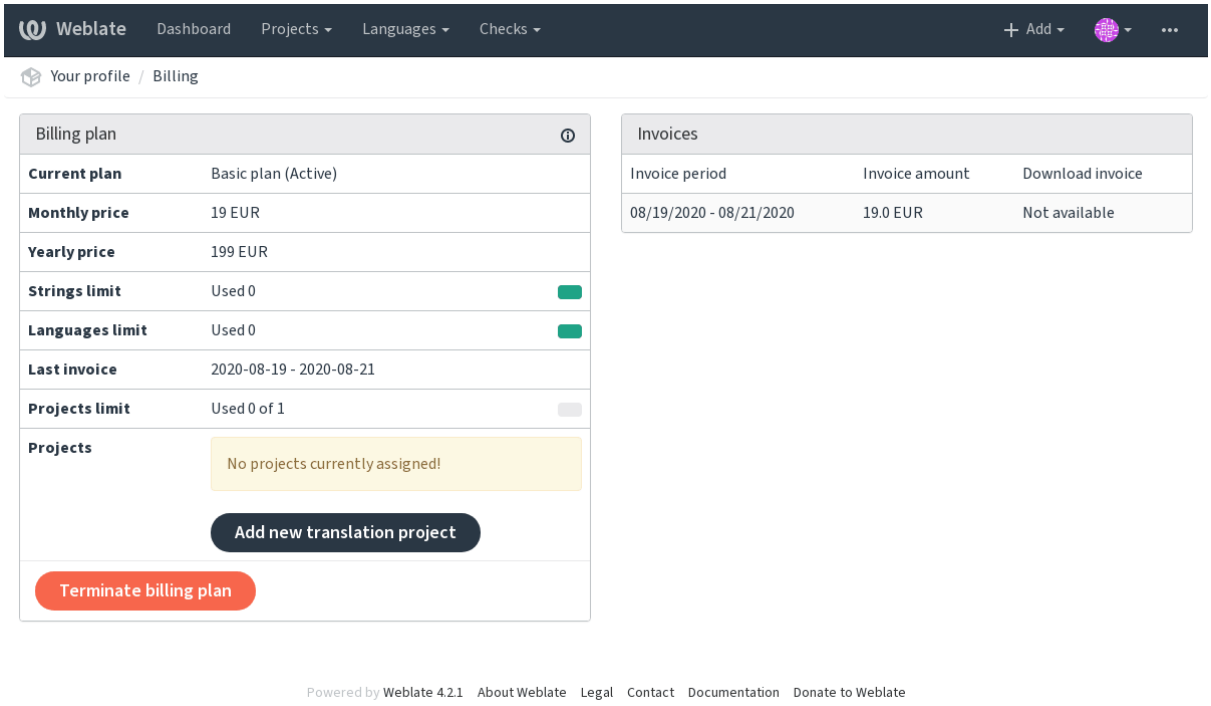
2.7.2 添加翻译项目和组件

在 3.2 版更改: 已包含用于添加项目和组件的界面，您不再需要使用*Django 管理界面*。

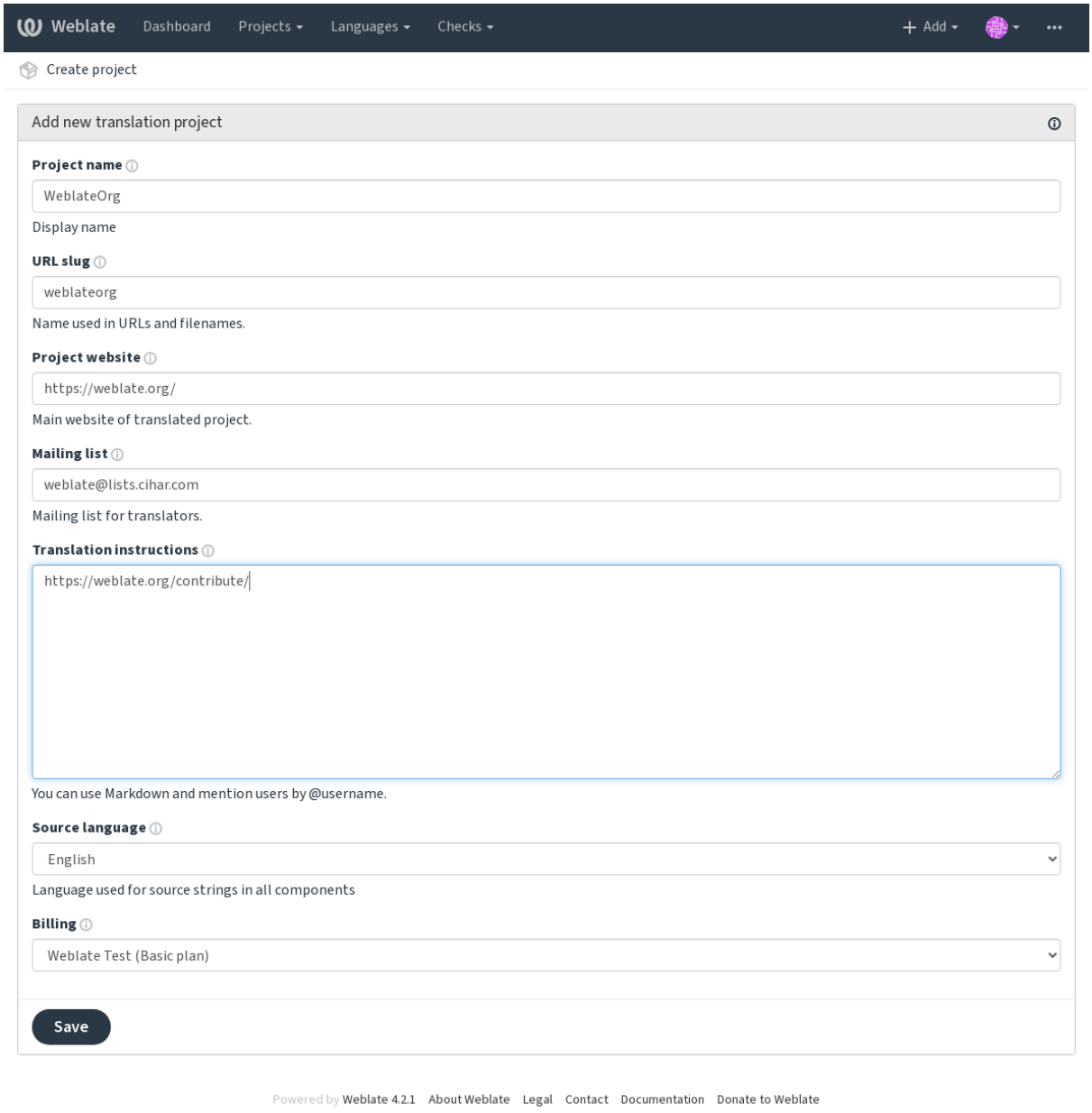
在 3.4 版更改: 现在，添加组件的过程是多阶段的，可以自动发现大多数参数。

根据您的许可，可以创建新的翻译项目和组件。超级用户始终允许使用此功能，并且如果您的实例使用计费方式（例如 https://hosted.weblate.org/ 参见账单），您还可以根据管理计费的用户帐户中的计划额度来创建计费方式。

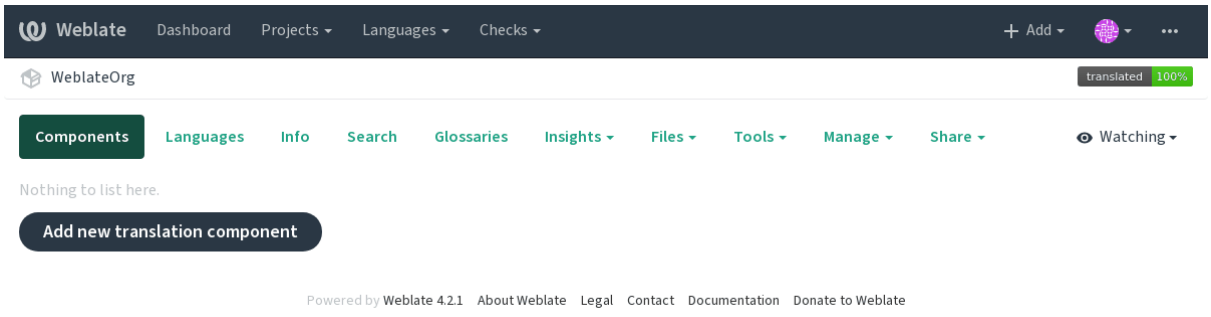
您可以在单独的页面上查看当前的结算方案：



您可以从此处开始创建项目，也可以使用导航栏中的菜单来填写翻译项目的基本信息以完成添加：



创建项目后，您将直接进入项目页面：



只需单击一次即可启动创建新翻译组件的操作。创建组件的过程是多阶段的，并自动检测大多数翻译参数。有几种创建组件的方法：

从版本控制 从远程版本控制仓库创建组件。

从现有组件 通过选择不同的文件为现有组件创建其他组件。

其他分支 仅针对不同分支，为现有组件创建其他组件。

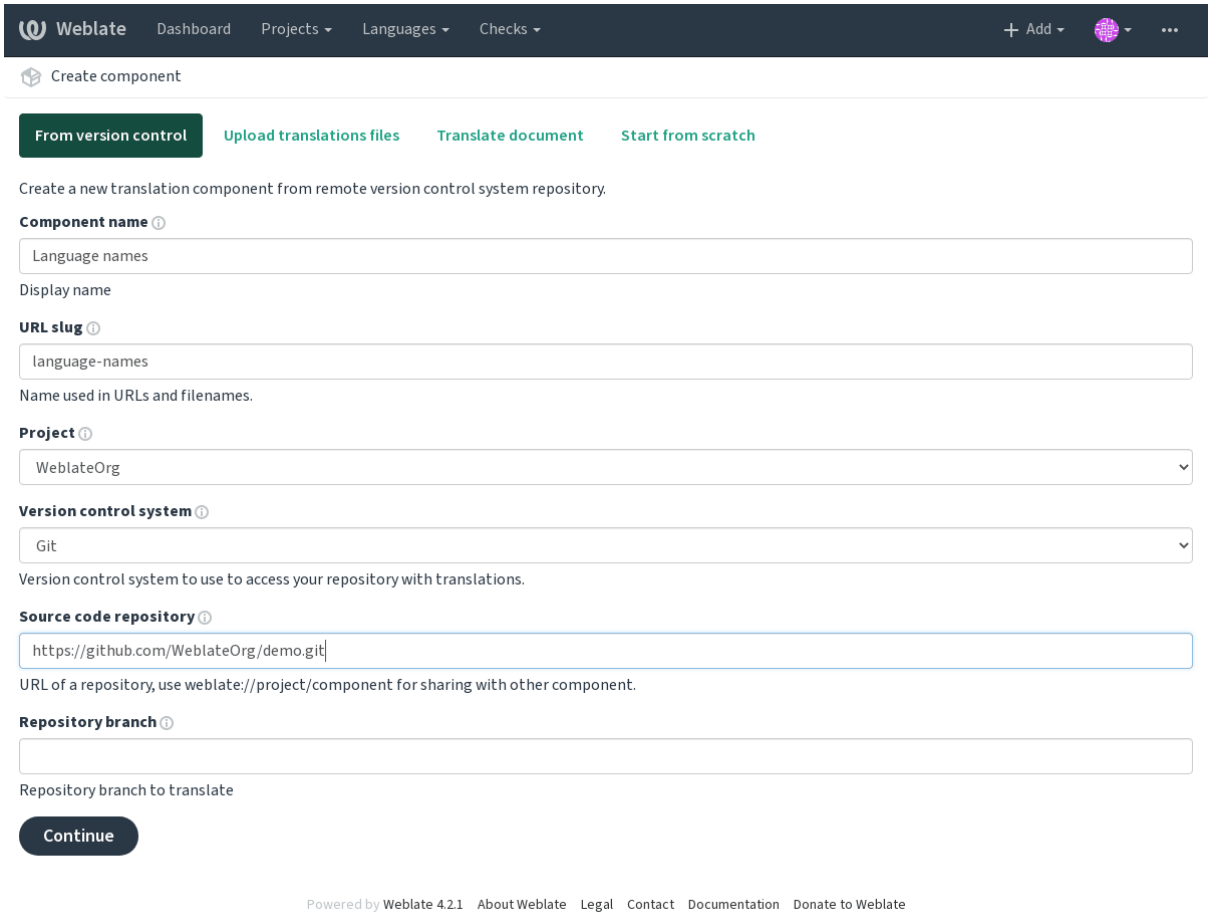
上传翻译文件 如果您没有版本控制或不想将其与 Weblate 集成，则将翻译文件上传到 Weblate。您以后可以使用网络界面或 API 更新内容。

翻译文档 上传单个文档并进行翻译。

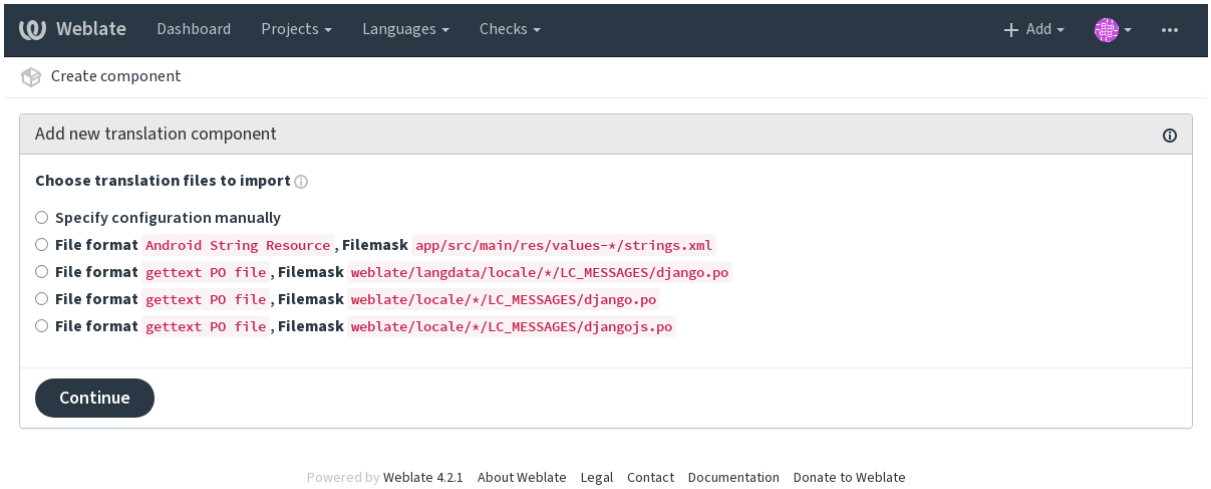
从头开始 创建空白翻译项目并手动添加字符串。

一旦有了现有的翻译组件，就可以使用同一仓库轻松地为其他文件或分支添加新的组件。

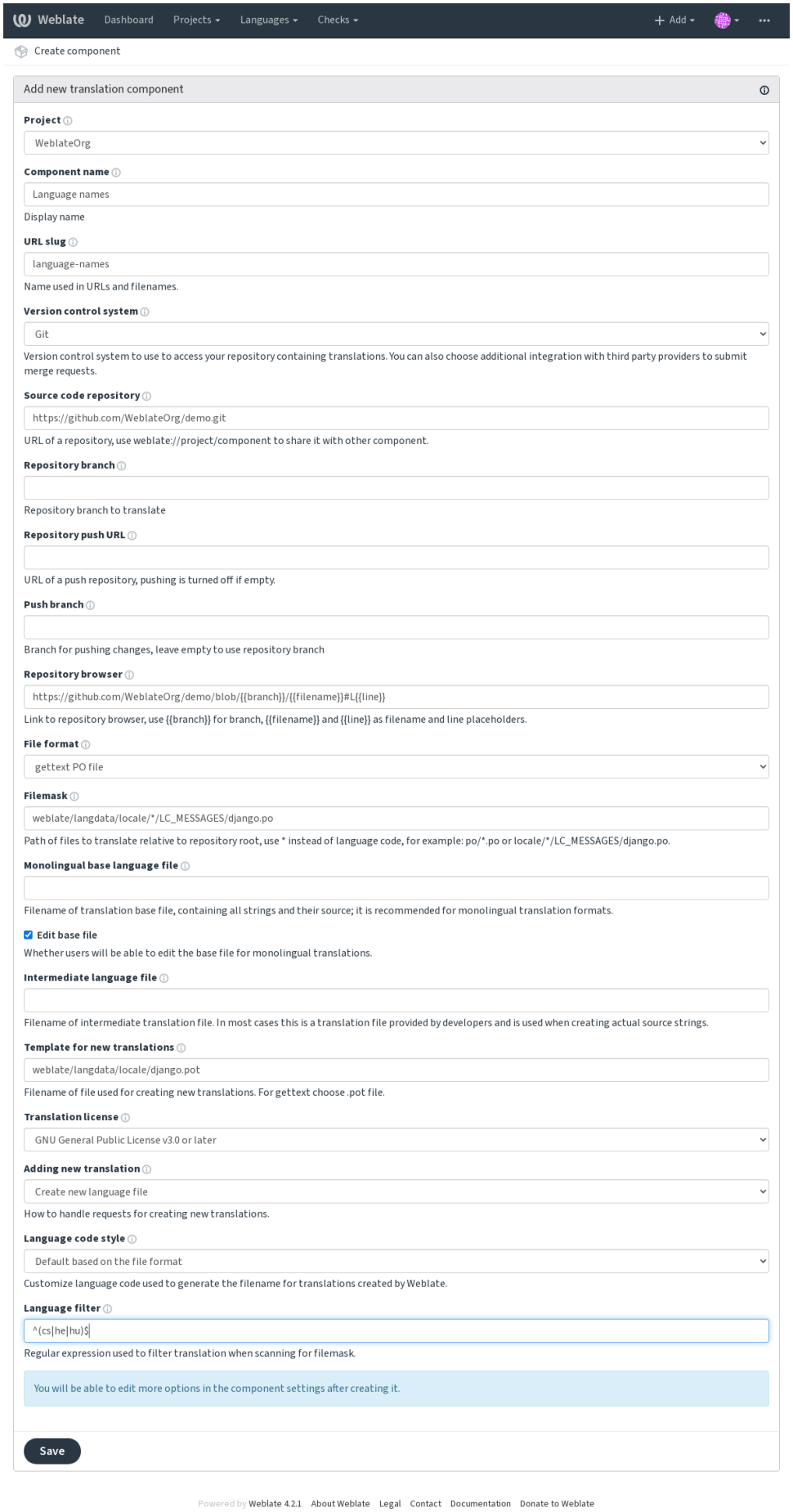
首先，您需要填写名称和仓库位置：



在下一页上，将显示已发现的可翻译资源的列表：



最后，您检查翻译组件信息并填写可选详细信息：


Weblate
Dashboard
Projects
Languages
Checks
Add
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.
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 when scanning for filemask.
You will be able to edit more options in the component settings after creating it.
Save
Powered by Weblate 4.2.1 About Weblate Legal Contact Documentation Donate to Weblate

参见:

Django 管理界面, 项目配置, *Component configuration*

2.7.3 项目配置

创建一个翻译项目，然后在其中添加一个新的翻译组件。这个项目就像一个架子，里面堆放着真正的翻译。同一项目中的所有组件共享建议及其字典；翻译也将自动传播到单个项目中的所有组件（除非在组件配置中关闭），请参见 Memory Management。

这些基本属性设置并通知翻译人员项目：

项目名称

详细的项目名称，用于显示项目名称。

Project slug

适用于 URL 的项目名称。

项目网站

译者可以在其中找到有关该项目的更多信息的 URL。

邮件列表

译者可以在其中讨论或评论翻译的邮件列表。

翻译说明

指向更多网站的 URL，为翻译人员提供了更详细的说明。

设置 Language-Team 头

Weblate 是否应管理 `Language-Team` 头（目前这是仅*GNU gettext* 功能）。

使用共享的翻译记忆库

是否使用共享翻译记忆库，有关更多详细信息，请参见共享翻译记忆库。

贡献到共享的翻译记忆库

是否贡献到共享翻译记忆库，请参阅共享翻译记忆库 以获取更多详细信息。

访问控制

配置每个项目的访问控制，请参阅根据项目的访问控制 以获取更多详细信息。

可以通过*DEFAULT_ACCESS_CONTROL* 更改默认值。

启用复查

Enable review workflow for translations, see 专门的审核者.

启用来源评论

Enable review workflow for source strings, see 源字符串复查.

启用 hooks

是否将未经身份验证的通知钩子 用于此仓库。

源语言

所有组件中用于源字符串的语言。如果您要翻译的不是英语，请更改此选项。

提示： In case you are translating bilingual files from English, but want to be able to do fixes in the English translation as well, you might want to choose *English (Developer)* as a source language. To avoid conflict between name of the source language and existing translation.

For monolingual translations, you can use intermediate translation in this case, see 中间语言文件.

参见：

中间语言文件, *Quality gateway for the source strings*, 双语和单语格式, *Language definitions*

语言别名

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.

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`

提示： 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.

参见：

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.

You can find some examples of typical configurations in the 支持的文件格式.

注解： 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:

组件名称

Verbose component name, used to display the component name.

Component slug

Component name suitable for URLs.

Component project

项目配置 where the component belongs.

版本控制系统

VCS to use, see 版本控制集成 for details.

源代码库

VCS repository used to pull changes.

参见：

See *Accessing repositories* for more details on specifying URLs.

提示： 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.

代码库推送 URL

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.

参见:

See *Accessing repositories* for more details on how to specify a repository URL and *推送 Weblate 的更改* for more details on pushing changes from Weblate.

代码库浏览器

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, you might want to strip leading directory by `parentdir` filter (see *Template markup*): `https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename|parentdir}}#L{{line}}`

已导出代码库 URL

URL where changes made by Weblate are exported. This is important when *持续本地化集成* is not used, or when there is a need to manually merge changes. You can use *Git exporter* to automate this for Git repositories.

仓库分支

Which branch to checkout from the VCS, and where to look for translations.

推送分支

Branch for pushing changes, leave empty to use *仓库分支*.

注解: This is currently only supported for Git and GitHub, it is ignored for other VCS integrations.

文件掩码

要翻译的文件的掩码，包括路径。它应包含一个 "*" 替换语言代码（有关处理方式的信息，请参阅*Language definitions*）。如果您的仓库包含多个翻译文件（例如，多个 gettext 域），则您需要为每个文件创建一个组件。

例如 "po/*.po*" 或 *"locale/*LC_MESSAGES/django.po"。

如果文件名包含特殊字符（例如 "[，]"），则需要将这些特殊字符转义为 `[[]` 或 `[]]`。

参见:

双语和单语格式, *What does mean "There are more files for the single language (en)"?*

单语种译文模版语言文件

Base file containing string definitions for 单语言组件.

参见：

双语和单语格式, *What does mean “There are more files for the single language (en)”?*

编辑译文模版文件

Whether to allow editing the base file for 单语言组件.

中间语言文件

Intermediate language file for 单语言组件. In most cases this is a translation file provided by developers and is used when creating actual source strings.

When set, the source translation is based on this file, but all others are based on 单语种译文模版语言文件. In case the string is not translated in source translation, translating to other languages is prohibited. This provides *Quality gateway for the source strings*.

参见：

Quality gateway for the source strings, 双语和单语格式, *What does mean “There are more files for the single language (en)”?*

新翻译的译文模版

Base file used to generate new translations, e.g. `.pot` file with gettext.

提示： 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.

参见：

Adding new translations, 添加新翻译, 双语和单语格式, *What does mean “There are more files for the single language (en)”?*

文件格式

Translation file format, see also 支持的文件格式.

Source string bug report address

Email address used for reporting upstream bugs. This address will also receive notification about any source string comments made in Weblate.

允许同步翻译

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*.

启用建议

Whether translation suggestions are accepted for this component.

建议投票

Turns on votecasting for suggestions, see 建议投票.

自动接受建议

Automatically accept voted suggestions, see 建议投票.

翻译标记

Customization of quality checks and other Weblate behavior, see 定制行为.

强制检查

List of checks which can not be ignored, see 强制检查.

翻译许可证

License of the translation (does not need to be the same as the source code license).

贡献者协议

翻译此内容前需同意的协议。

添加新翻译

如何处理创建新语言的请求。可用选项：

联系维护者 用户可以选择所需的语言，项目维护者将收到有关该语言的通知。由他们决定是否向仓库添加（或不添加）语言。

显示翻译介绍 向用户显示的页面链接描述了开始新翻译的过程。如果需要更正式的流程（例如，在开始实际翻译之前组成人员团队），请使用此选项。

创建新语言文件 用户可以选择语言，然后 Weblate 会自动为其创建文件并开始翻译。

禁用添加新翻译 用户将无法选择开始新的翻译。

参见：

Adding new translations.

语言代码风格

Customize language code used to generate the filename for translations created by Weblate, see *Adding new translations* for more details.

合并方式

You can configure how updates from the upstream repository are handled. This might not be supported for some VCSs. See 结合或变基 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*.

提交者姓名

Name of the committer used for Weblate commits, the author will always be the real translator. On some VCSs this might be not supported.

Default value can be changed by *DEFAULT_COMMITER_NAME*.

提交者邮箱

Email of committer used for Weblate commits, the author will always be the real translator. On some VCSs this might be not supported. The default value can be changed in *DEFAULT_COMMITER_EMAIL*.

提交时推送

Whether committed changes should be automatically pushed to the upstream repository. When enabled, the push is initiated once Weblate commits changes to its internal repository (see 惰性提交). To actually enable pushing *Repository push URL* has to be configured as well.

对变更进行提交的延时时间

Sets how old changes (in hours) are to get before they are committed by background task or *commit_pending* management command. All changes in a component are committed once there is at least one older than this period.

Default value can be changed by *COMMIT_PENDING_HOURS*.

出错时锁定

Enables locking the component on repository error (failed pull, push or merge). Locking in this situation avoids adding another conflict which would have to be resolved manually.

The component will be automatically unlocked once there are no repository errors left.

语言筛选

Regular expression used to filter the translation when scanning for filemask. This can be used to limit the list of languages managed by Weblate.

注解： You need to list language codes as they appear in the filename.

Some examples of filtering:

Filter description	正则表达式
Selected languages only	`^(cs|de|es)$`
Exclude languages	`^(?!(it|fr)$).+$`
Exclude non language files	`^(?!(blank)$).+$`
Include all files (default)	`^[^.]+$`

正则表达式变体

Regular expression used to determine the variants of a string, see *String variants*.

注解： Most of the fields can be edited by project owners or managers, in the Weblate interface.

参见：

Does Weblate support other VCSes than Git and Mercurial?, *Translation component alerts*

优先权

高优先级的组件将最先提供给翻译者。

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.

Enable this in case you want to grant access to this component explicitly - the project level permissions will not apply and you will have to specify component or component list level permission in order to grant access.

Default value can be changed by *DEFAULT_RESTRICTED_COMPONENT*.

提示： This applies to project managers as well - please make sure you will not loose access to the component after toggling the status.

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**
  - 组件发现
  - 统计数据生成器
  - 从附加组件执行脚本

There following variables are available in the component templates:

{{ language_code }} 语言代码

{{ language_name }} 语言名称

{{ component_name }} 组件名称

{{ component_slug }} Component slug

{{ project_name }} 项目名称

{{ project_slug }} Project slug

{{ url }} Translation URL

{{ filename }} 翻译文件名

{{ 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 组件发现 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*.

参见:

Management commands, 组件发现

2.8 Language definitions

To present different translations properly, info about language name, text direction, plural definitions and language code is needed. Definitions for about 350 languages are included.

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 语言别名.

If no exact match can be found, an attempt will be made to best fit it into an existing language (e.g. 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) and naming of the language 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*).

提示: In case you see something unwanted as a language, you might want to adjust 语言筛选 to ignore such file when parsing 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 Language definitions

Each language consists of following fields:

语言代码

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.

参见:

Parsing language codes

语言名称

Visible name of the language. The language names included in Weblate are also being localized depending on user interface language.

文字方向

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.

复数式

Gettext compatible plural formula used to determine which plural form is used for given count.

参见:

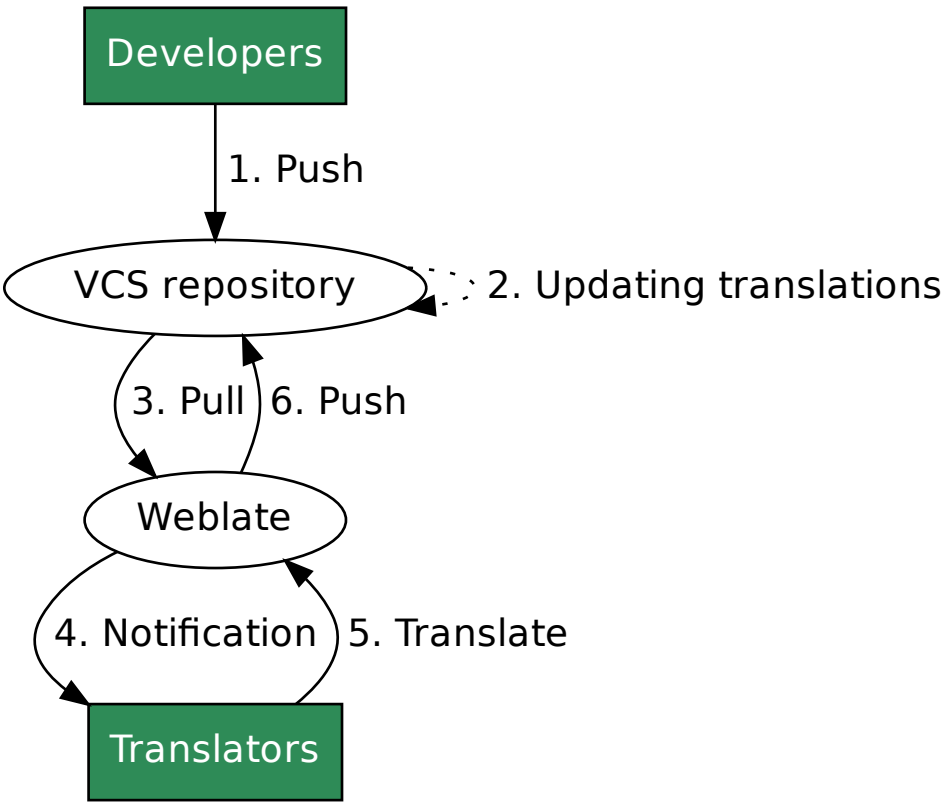
复数形式, GNU gettext utilities: Plural forms, Language Plural Rules by the Unicode Consortium

2.9 持续本地化集成

有适当的基础结构，因此您的翻译紧随开发。这样，翻译人员可以一直进行翻译，而不必在发布之前处理大量的新文本。

这是过程：

1. 开发人员进行更改并将其推送到 VCS 仓库。
2. 可以选择更新翻译文件（这取决于文件格式，请参阅*Why does Weblate still show old translation strings when I' ve updated the template?*）。
3. Weblate 从 VCS 仓库中拉取更改，请参阅更新仓库。
4. 一旦 Weblate 检测到翻译更改，便会根据翻译者的订阅设置通知他们。
5. 翻译者使用 Weblate Web 界面提交翻译，或上传离线更改。
6. 翻译者完成后，Weblate 会将更改提交到本地仓库（请参阅惰性提交），如果有权限将其推回（请参阅推送 *Weblate* 的更改）。



2.9.1 更新仓库

应该设置一些方式来从他们的源更新后端仓库。

- 使用通知钩子 来与多数常见的代码托管服务集成
- 在仓库管理中或使用 API 或 *Weblate 客户端* 来手动触发更新
- 允许 *AUTO_UPDATE* 在您的 Weblate 事例上自动更新所有组件
- 执行 `updategit` （选择项目，或 *–all* 来更新全部）

无论何时 Weblate 更新仓库，都将出发以前更新插件，请见： 附加组件 。

避免合并冲突

当相同的文件在 Weblate 之内与之外都更改时导致 Weblate 的合并冲突。有两种方法来处理——避免在 Weblate 之外编辑，或者将 Weblate 集成到您的更新过程中，从而在更新 weblate 之外的文件之前刷新更改。

第一种方法容易用于单语种文件——可以添加 Weblate 之内的新字符串，并将文件的整个编辑留在那里。对于双语种文件，通常存在某种消息提取过程而从源代码产生翻译文件。在一些情况下，这可以分成两部分——一部分用于提取过程产生模板（例如使用 **xgettext** 产生 gettex POT ），然后下一步过程将它合并到真正的翻译中（例如使用 **msgmerge** 更新 gettext PO 文件）。可以在 Weblate 中执行第二步，它将确认在这个操作前所有挂起的更改都包括进去了。

第二种方法可以这样实现，使用 API ，当您在自己一侧进行更改时，强制 Weblate 推送所有挂起的更改，并锁定翻译。

进行更新的脚本看起来像这样：

```
# 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
```

如果多个组件分享相同的仓库，您需要分别将他们全部锁定：

```
wlc lock foo/bar
wlc lock foo/baz
wlc lock foo/baj
```

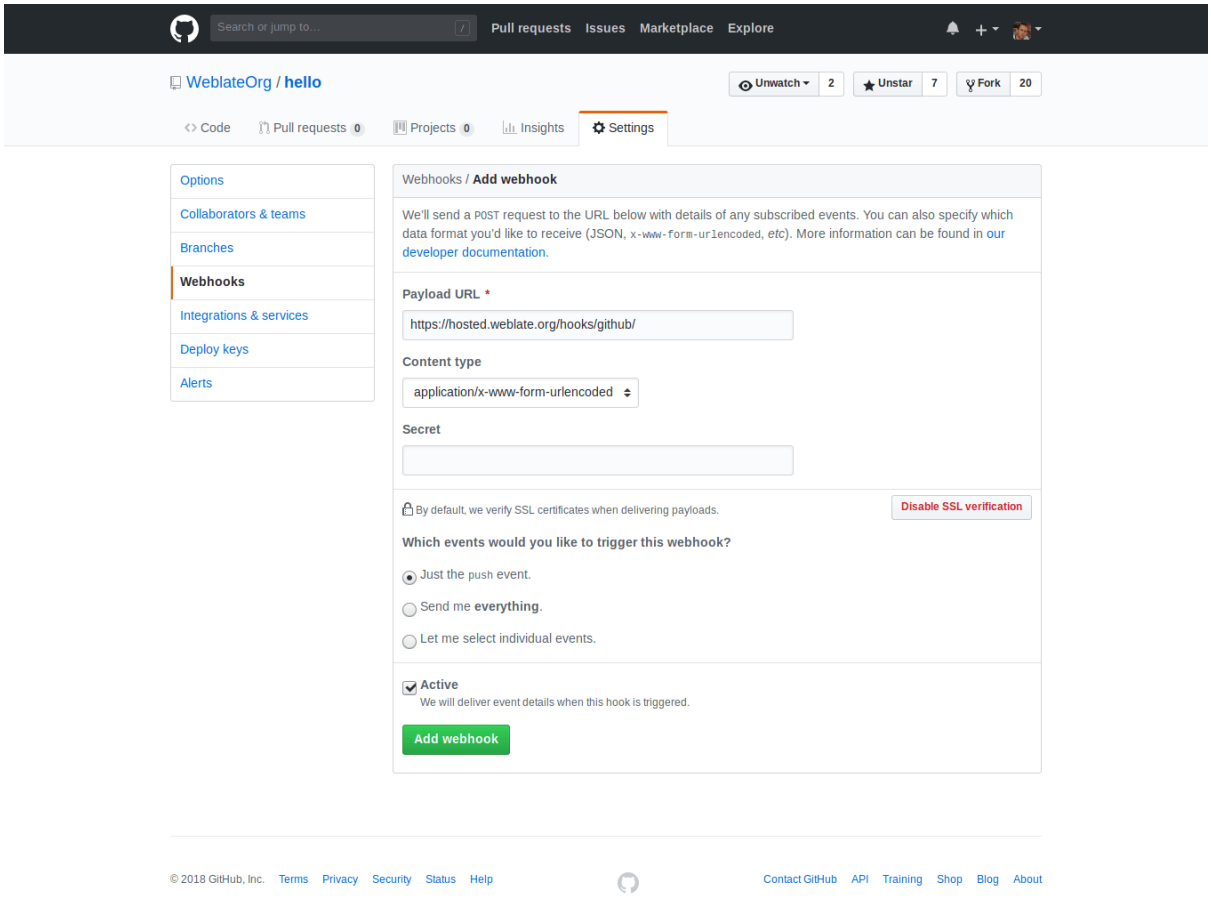
注解： 例子使用了*Weblate 客户端*，这需要配置（API 密钥）来远程控制 Weblate 。可以通过使用 HTTP 客户端代替 wlc ，例如 curl 来实现，请见 API 。

从 GitHub 自动接收更改

Weblate 伴随 GitHub 本地支持。

如果使用 Hosted Weblate ，建议的方法是安装 Weblate app，该方法能够得到正确的设置，而不必设置很多东西。它还可以用于将更改推送回来。

为了在每次推送到 GitHub 仓库时接收通知，将 Weblate Webhook 添加到仓库设置（*Webhooks* ）中，如下图所示：



对于负载 URL，将 `/hooks/github/` 增补到您的 Weblate URL 中，例如对于 Hosted Weblate 服务，这是 `https://hosted.weblate.org/hooks/github/` 。

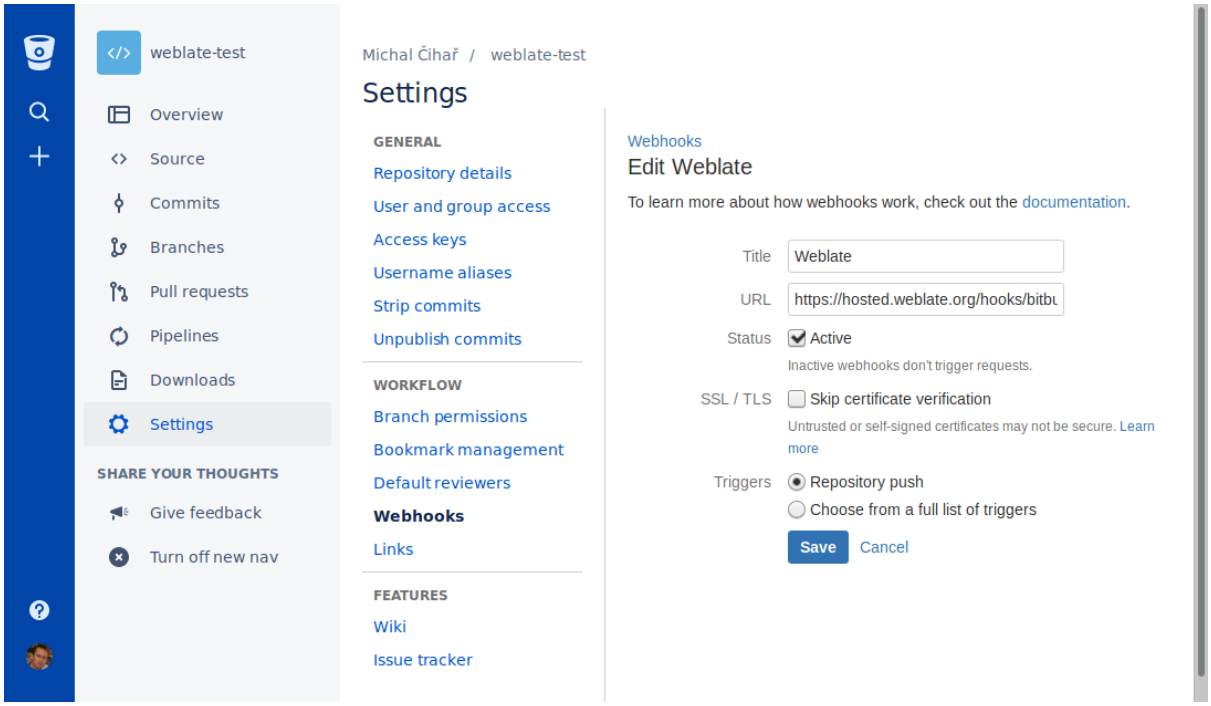
可以将其他设置保留为默认值（Weblate 可以处理内容类型，并只消费 *push* 事件）。

参见:

`POST /hooks/github/`, *Accessing repositories from Hosted Weblate*

从 Bitbucket 自动接收更改

Weblate 已经支持 Bitbucket webhooks，添加仓库推送时触发的 webhook，目的地为您 Weblate 安装上的 `/hooks/bitbucket/` （例如 `https://hosted.weblate.org/hooks/bitbucket/` ）。



参见:

`POST /hooks/bitbucket/`, *Accessing repositories from Hosted Weblate*

从 GitLab 自动接收更改

Weblate 已经支持 GitLab hooks，添加项目的 webhook，目的地为您 Weblate 安装上的 `/hooks/gitlab/`（例如 `https://hosted.weblate.org/hooks/gitlab/`）。

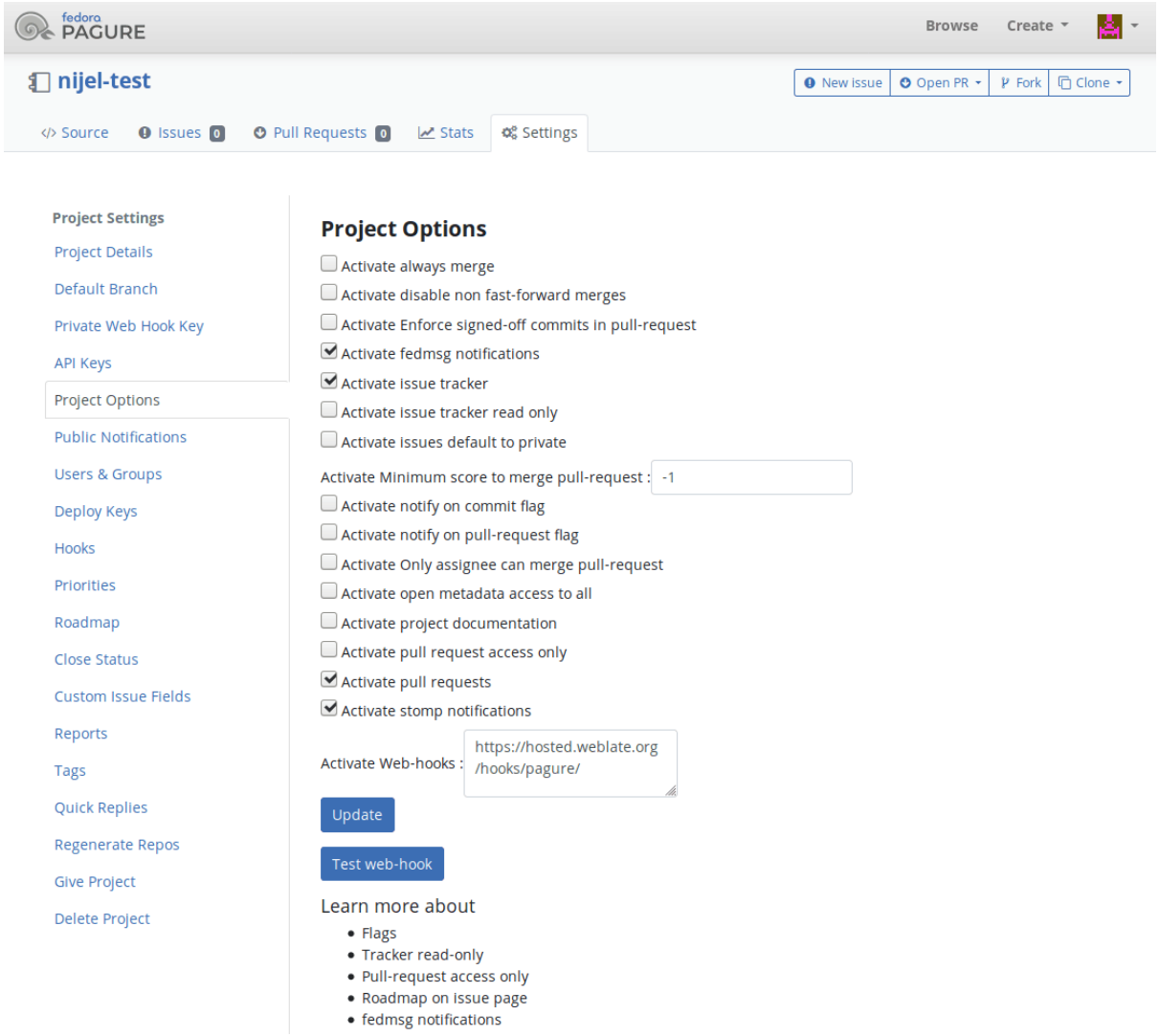
参见:

`POST /hooks/gitlab/`, *Accessing repositories from Hosted Weblate*

从 Pagure 自动接受更改

3.3 新版功能.

Weblate 已经支持 Pagure hooks，添加项目的 webhook，目的地为您 Weblate 安装上的 `/hooks/pagure/`（例如 `https://hosted.weblate.org/hooks/pagure/`）。这可以在 *Project options* 之下的 *Activate Web-hooks* 中完成：



参见:

`POST /hooks/pagure/`, *Accessing repositories from Hosted Weblate*

从 Azure Repos 自动接收更改

3.8 新版功能.

Weblate 已经支持 Azure Repos web hooks，为 *Code pushed* 事件添加 webhook，目的地为您 Weblate 安装上的 `/hooks/azure/` URL （例如 `https://hosted.weblate.org/hooks/azure/` ）。这可以在 *Project settings* 之下的 *Service hooks* 中完成。

参见:

Web hooks in Azure DevOps manual, `POST /hooks/azure/`, *Accessing repositories from Hosted Weblate*

从 Gitea Repos 自动接收更改

3.9 新版功能.

Weblate 已经支持 Gitea webhooks，为 *Push events* 事件添加 *Gitea Webhook*，目的地为您 Weblate 安装上的 `/hooks/gitea/` URL（例如 `https://hosted.weblate.org/hooks/gitea/`）。这可以在 *Settings* 之下的 *Webhooks* 中完成。

参见：

Webhooks in Gitea manual, `POST /hooks/gitea/`, *Accessing repositories from Hosted Weblate*

从 Gitee Repos 自动接收更改

3.9 新版功能.

Weblate 已经支持 Gitee webhooks，为 *Push* 事件添加 *Webhook*，目的地为您 Weblate 安装上的 `/hooks/gitee/` URL（例如 `https://hosted.weblate.org/hooks/gitee/`）。这可以在 *Management* 之下的 *Webhooks* 中完成。

参见：

Webhooks in Gitee manual, `POST /hooks/gitee/`, *Accessing repositories from Hosted Weblate*

每晚自动更新仓库

Weblate 在后面合并更改时，每晚自动获取远程仓库来提高性能。您可以选择将其同样转换为进行每晚合并，通过允许`AUTO_UPDATE`。

2.9.2 推送 Weblate 的更改

每个翻译组件可以具有一个 URL 设置（请见代码库推送 *URL*），在那种情况下 Weblate 能够将更改推送到远程仓库。Weblate 还可以配置在每次提交时自动推送更改（这是默认的，请见提交时推送）。如果不想更改自动给推送，可以在 *Repository maintenance* 之下手动进行，或通过`wlc push` 使用 API。

推送选项根据使用的版本控制集成 而不同，更多细节可以在那个章节中找到。

在不想由 Weblate 直接推送的情况下，有对*GitHub* 、*GitLab* 推送请求或*Gerrit* 复查的支持，可以通过选择 *GitHub* 、*GitLab* 或 *Gerrit* 作为*Component configuration* 中的版本控制系统 来激活。

整体上，Git 、GitHub 和 GitLab 可以具有后面的选项：

需要的设置	版本控制系统	代码库推送 *URL*	推送分支
不推送	*Git*	*empty*	*empty*
直接推送	*Git*	SSH URL	*empty*
推送到单独的分支	*Git*	SSH URL	分支名称
来自叉子的 GitHub 拉取请求	*GitHub*	*empty*	*empty*
来自分支的 GitHub 拉取请求	*GitHub*	SSH URL[1]	分支名称
来自叉子的 GitLab 结合请求	*GitLab*	*empty*	*empty*
来自分支的 GitLab 结合请求	*GitLab*	SSH URL[1]	分支名称

注解： 还可以允许 Weblate 提交后更改的自动推送，这可以在提交时推送 中完成。

参见：

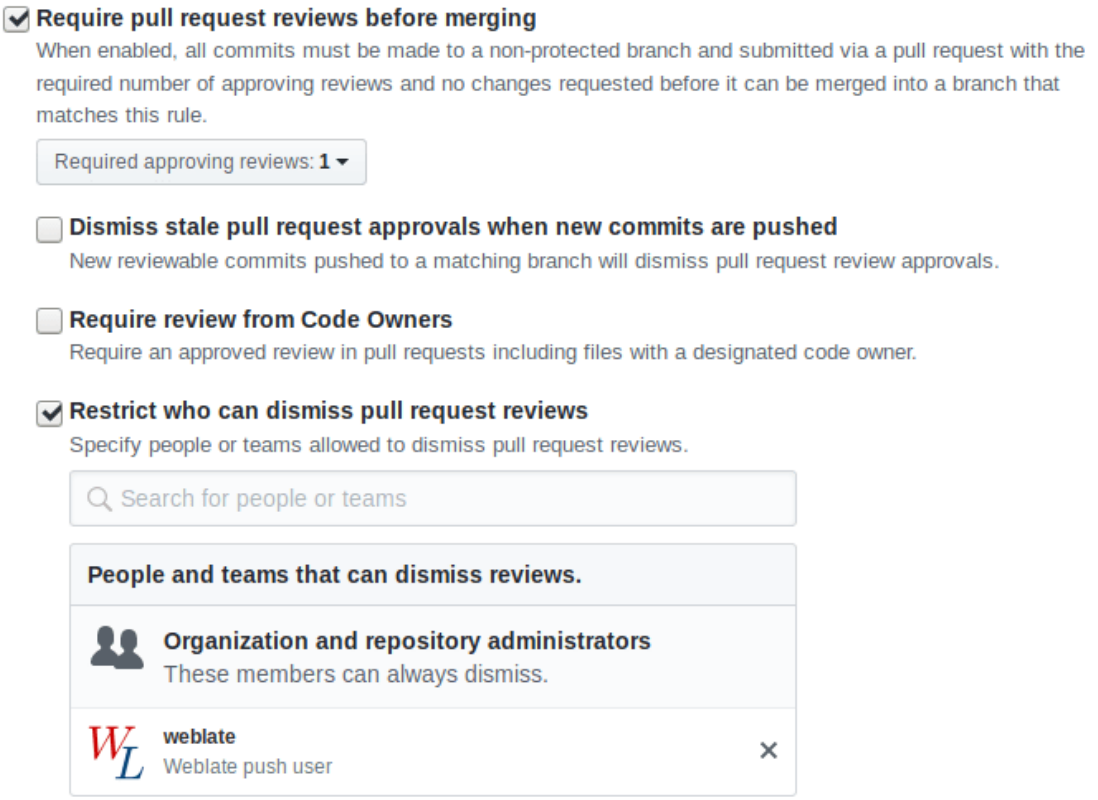
请见*Accessing repositories* 来设置 SSH 密钥，和惰性提交 获得关于 Weblate 决定提交更改的信息。

[1] 在源代码库 支持推送的情况下可以为空。

受保护的分支

如果在受保护的分支上使用 Weblate，可以配置使用推送请求，并执行翻译的实际复查（对您不知道的语言可能有问题）。另一个方法是去掉对 Weblate 推送用户的这个限制。

例如在 GitHub，这可以在仓库配置中进行：



2.9.3 结合或变基

Weblate 默认将上游仓库结合到自身。在您还通过其他方式访问下层仓库的情况下，这是最安全的方式。在不需要这个的情况下，可以允许上游更改的变基，这会产生较少的融合提交的历史。

注解： 在复杂融合的情况下，变基可能是您产生麻烦，因此请仔细考虑是否允许它们。

2.9.4 与他人交互

Weblate 通过使用它的 API，使与他人的交流更容易。

参见：

API

2.9.5 惰性提交

Weblate 的行为是，如果可能的话，将来自相同作者的提交分组成为一个提交。这极大地减少了提交的数量，然而，在您想要同步得到版本管理系统 VCS 仓库，例如用于结合的情况下（这对管理者群组是默认允许的，请见访问控制），您可能需要明确告诉它进行提交。

一旦后面的任何条件满足，这种模式的更改将被提交：

- 某人另外更改了已经被更改的字符串。
- 来自上游的结合发生了。
- 明确地请求了提交。
- 更改比*Component configuration* 上的 *Age of changes to commit* 定义的时间段更陈旧。

提示： 每个组件都建立提交。所以在具有很多组件的情况下，仍然会看到很多提交。在这种情况下您会使用压缩 *Git* 提交 插件。

如果想要更频繁地提交更改，并且不检查新旧，可以安排周期定时性任务来执行提交：

```
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 用脚本处理仓库

定制 Weblate 与仓库交互的方式是附加组件 。关于如何通过插件执行外部脚本的信息，请咨询从附加组件执行脚本 。

2.9.7 在部件之间保持翻译一致

一旦具有多个翻译组件，您会想要确保相同的字符串具有相同的翻译。这可以在几个层次实现。

翻译宣传

允许翻译宣传时（这是默认的，请见*Component configuration* ），在所有的组件中字符串匹配时，所有新的翻译自动进行。在所有的组件中这样的翻译都适当地归功于当前翻译的用户。

注解： 翻译宣传需要密钥来匹配单语种翻译格式，因此在建立翻译密钥匙请记住。

一致性检查

任何时候字符串不同时不一致的 都会检查启动。可以使用这个来手动复查这样的差异，并选择正确的翻译。

自动化翻译

基于不同组件的自动翻译，可以是在组件之间同步翻译的方式。可以或者手动触发（请见自动化翻译 ），或者使用插件（自动化翻译 ）在仓库更新时自动运行。

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.

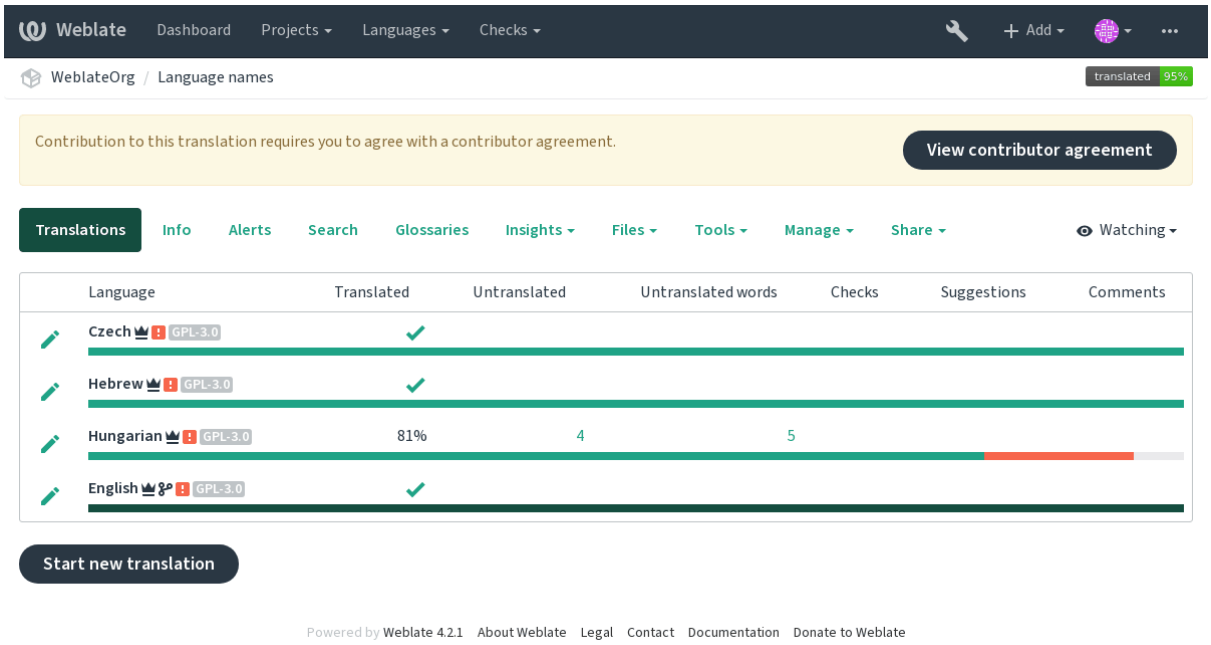
2.10.1 License info

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 贡献者协议

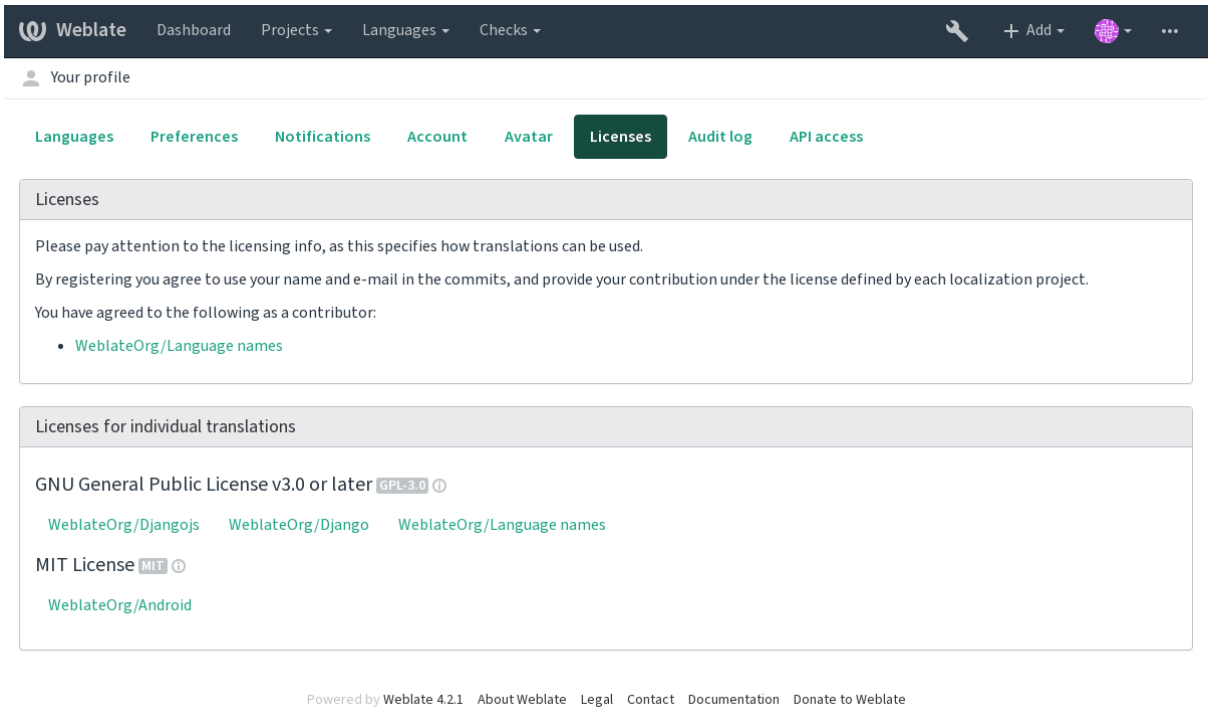
If you specify a contributor license agreement, only users who have agreed to it will be able to contribute. This is a clearly visible step when accessing the translation:



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 翻译进程

2.11.1 建议投票

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 signed in user agrees, by setting up the *Component configuration* 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).

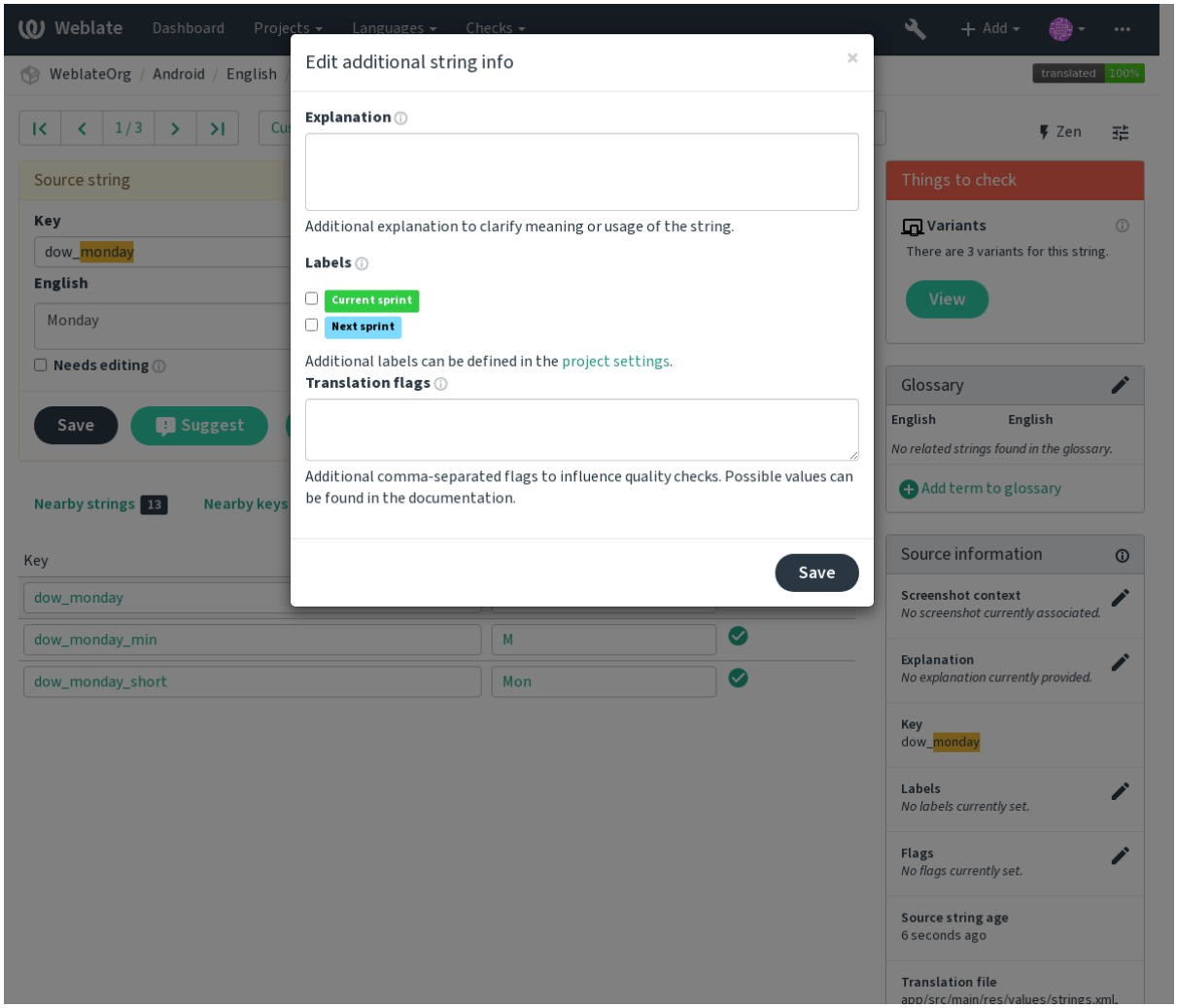
注解: Once automatic acceptance is set up, normal users lose the privilege to directly save translations or accept suggestions. This can be overridden with the *Can override suggestion state* privilege (see 访问控制).

You can combine these with 访问控制 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 with info available in the translation files. This includes explanation, string priority, check flags, or providing visual context. All these features can be set in the *Reviewing strings*:



Access this directly from the translation interface by clicking the "Edit" icon next to *Screenshot context* or *Flags*.


Weblate
Dashboard
Projects
Languages
Checks
Add
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
Clone source
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
Save
Suggest
Skip
Nearby strings 26
Comments
Machinery
Other languages
History
Context
English
Czech
State
Files
Soubory
Automatic translation
Automatický překlad
Add new translation string
Add new translation string
Translation status
Stav překladu
Singular
%(count)s word
Plural
%(count)s words
One
%(count)s slovo
Few
%(count)s slova
Other
%(count)s slov
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 správě, 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
Source information
Screenshot context
No screenshot currently associated.
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
Source string age
a second ago
Translation file
weblate/locale/cs/LC_MESSAGES/django.po, string 11
Powered by Weblate 4.2.1
About Weblate
Legal
Contact
Documentation
Donate to Weblate

Strings prioritization

2.0 新版功能.

String priority can be changed to offer higher priority strings for translation earlier by using the `priority` flag

提示： This can be used to order the flow of translation in a logical manner.

参见：

质量检查

翻译标记

2.4 新版功能.

在 3.3 版更改: 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.

参见：

质量检查

解释

在 4.1 版更改: In previous version this has been called extra context.

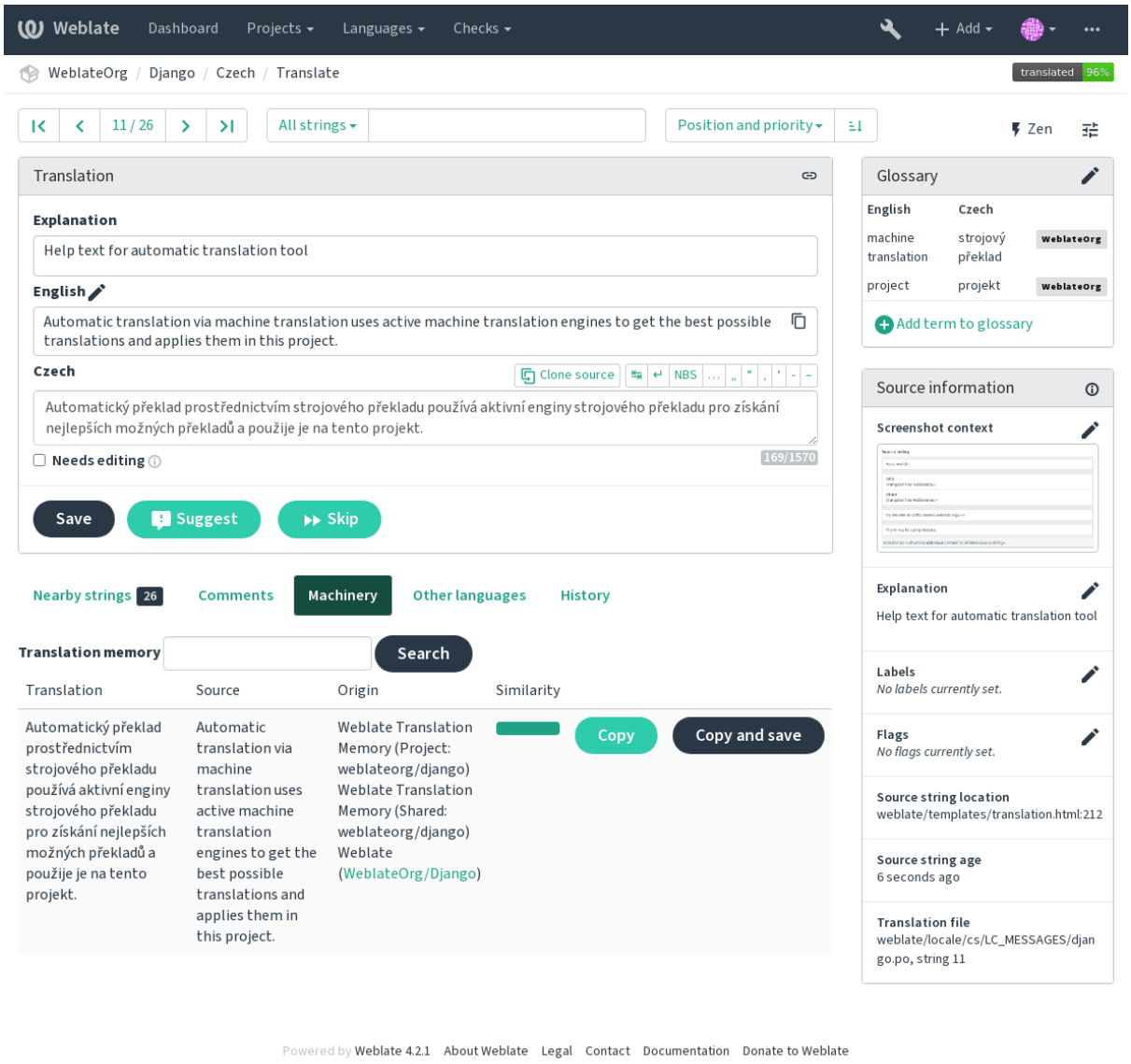
Use the explanation to clarify scope or usage of the translation. You can use Markdown to include links and other markup.

Visual context for strings

2.9 新版功能.

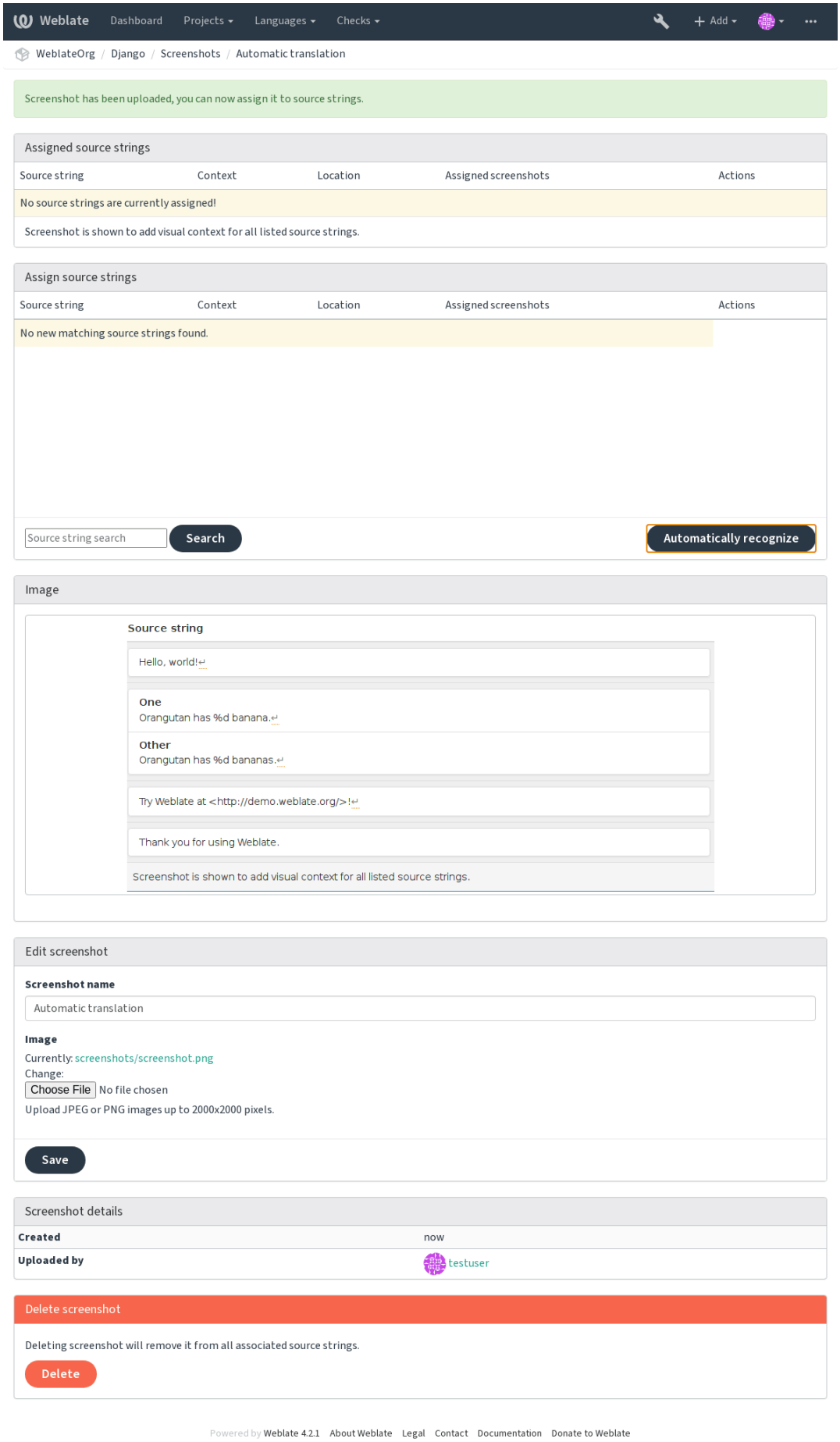
You can upload a screenshot showing a given source string in use within your program. This helps translators understand where it is used, and how it should be translated.

The uploaded screenshot is shown in the translation context sidebar:



In addition to *Reviewing 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
Add
WeblateOrg / Django / Screenshots / Automatic translation
Screenshot has been uploaded, you can now assign it to source strings.
Assigned source strings
Source string
Context
Location
Assigned screenshots
Actions
No source strings are currently assigned!
Screenshot is shown to add visual context for all listed source strings.
Assign source strings
Source string
Context
Location
Assigned screenshots
Actions
No new matching source 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
Delete screenshot
Deleting screenshot will remove it from all associated source strings.
Delete
Powered by Weblate 4.2.1
About Weblate
Legal
Contact
Documentation
Donate to Weblate

2.12 检查并修正

2.12.1 定制的自动修正

还可以应用除了自动修正以外自己的自动修正，并将它们包括到*AUTOFIX_LIST* 。

自动修复很强大，但可能导致损坏；写脚本的时候要小心。

例如，后面的自动修复会将每次出现的字符串 `foo` 在翻译中替换为 `bar`：

```
#
# Copyright © 2012 - 2020 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
```

为了安装定制的检查，在*AUTOFIX_LIST* 中为 Python 类提供完全合规的路径，请见定制的质量检查、插件和自动修复 。

2.12.2 定制行为

可以在每个源字符串（在源字符串复核中，请见*Additional info on source strings* ）或在*Component configuration* (翻译标记) 中精细调整 Weblate 的行为。一些文件格式允许直接在格式中指定标记（请见支持的文件格式 ）。

标记用逗号分隔，参数用冒号分隔。可以在字符串中使用引号来包含空白字符或特定字符。例如：

```
placeholders:"special:value":"other value", regex:.*
```

这里是现在能接受的标记的列表：

rst-text 将文本看作 RST 文件，影响未更改的翻译 。

md-text 将文本看作 Markdown 文件。

dos-eol 使用 DOS 的行末标记，而不是 Unix 的（`\r\n` instead of `\n` ）。

url 字符串应该只包括 URL 。

safe-html 字符串应该是 HTML 安全的，请见不安全的 *HTML* 网站 。

read-only 字符串应该只读，并且不能在 Weblate 中编辑。请见只读字符串 。

priority:N 字符串的优先级。高优先级的字符串首先出现被翻译。默认的优先级是 100 ，字符串的优先级越高，就会越早安排翻译。

max-length:N 将字符串的最大长度限制为 N 个字符，请见译文最大长度

xml-text 将文本看作 XML 文件，影响*XML* 语法 and *XML* 标记 。

font-family:NAME 定义 font-family 来提供检查，请见管理字型 。

font-weight:WEIGHT 定义 font-weight 来提供检查，请见管理字型 。

font-size:SIZE 定义 font-size 来提供检查，请见管理字型 。

font-spacing:SPACING 定义 font-spacing 来提供检查，请见管理字型 。

placeholders:NAME 翻译中需要的占位字符串，请见占位符 。

**replacements:FROM:TO:FROM2:TO2...`** 当检查结果文本参数时执行替代（例如在最大翻译大小或译文最大长度 中）。这一典型应用的情况拓展了非译元素，确保匹配那些即使使用了长名称的文本，例如 `replacements:%s:"John Doe"` 。

regex:REGEX 用于匹配翻译文件的正则表达式，详见正则表达式 。

python-format, c-format, php-format, python-brace-format, javascript-format, c-sharp-format, ja 将所有字符串当作格式字符串，影响格式化字符串, 格式化字符串, 格式化字符串, 格式化字符串, 格式化字符串, 格式化字符串, 格式化字符串, 格式化字符串, 格式化字符串, 格式化字符串, 未更改的翻译 。

strict-same 使用内建的单词黑名单，来避免“没有翻译”的检查提示。请见未更改的翻译 。

ignore-bbcode 跳过“BBcode 标记”质量检查。

ignore-duplicate 跳过“连续重复的单词”质量检查。

ignore-double-space 跳过“双空格”质量检查。

ignore-angularjs-format 跳过“AngularJS 插补字符串”质量检查。

ignore-c-format 跳过“C 格式”质量检查。

ignore-c-sharp-format 跳过“C# 格式”质量检查。

ignore-es-format 跳过“ECMAScript 模板文本”质量检查。

ignore-i18next-interpolation 跳过“i18next 插补”质量检查。

ignore-java-format 跳过“Java 格式”质量检查。

ignore-java-messageformat 跳过“Java MesseageFormat ”质量检查。

ignore-javascript-format 提高过“JavaScript 格式”质量检查。

ignore-percent-placeholders 跳过“百分号占位符”质量检查。

ignore-perl-format 跳过“Perl 格式”质量检查。

ignore-php-format 跳过“PHP 格式”质量检查。

ignore-python-brace-format 跳过“Python brace 格式”质量检查。

ignore-python-format 跳过“Python 格式”质量检查。

ignore-qt-format 跳过“Qt 格式”质量检查。

ignore-qt-plural-format 跳过“Qt plural 格式”质量检查。

ignore-ruby-format 跳过“Ruby 格式”质量检查。

ignore-translated 跳过“已被翻译”质量检查。

ignore-inconsistent 跳过“不一致的”质量检查。

ignore-kashida 跳过“已使用 Kashida 字符”质量检查。

ignore-md-link 跳过“Markdown 链接”质量检查。

ignore-md-reflink 跳过“Markdown 引用”质量检查。

ignore-md-syntax 跳过“Markdown 语法”质量检查。

ignore-max-length 跳过“译文最大长度”质量检查。

ignore-max-size 跳过“译文最大尺寸”质量检查。

ignore-escaped-newline 跳过“换行符 n 不匹配”质量检查。

ignore-end-colon 跳过“不匹配的冒号”质量检查。

ignore-end-ellipsis 跳过“不匹配的省略号”质量检查。

ignore-end-exclamation 跳过“不匹配的感叹号”质量检查。

ignore-end-stop 跳过“不匹配的句号”质量检查。

ignore-end-question 跳过“不匹配的问号”质量检查。

ignore-end-semicolon 跳过“不匹配的分号”质量检查。

ignore-newline-count 跳过“不匹配的断行”质量检查。

ignore-plurals 跳过“缺少复数形式”质量检查。

ignore-placeholders 跳过“占位符”质量检查。

ignore-punctuation-spacing 跳过“标点间距”质量检查。

ignore-regex 跳过“正则表达式”指令检查。

ignore-same-plurals 跳过“相同复数”质量检查。

ignore-begin-newline 跳过“换行开头”质量检查。

ignore-begin-space 跳过“空格开头”质量检查。

ignore-end-newline 跳过“换行结尾”质量检查。

ignore-end-space 跳过“空格结尾”质量检查。

ignore-same 跳过“未更改的翻译”质量检查。

ignore-safe-html 跳过“不安全的 HTML 网站”质量检查。

ignore-url 跳过“URL ”质量检查。

ignore-xml-tags 跳过“XML 标记”质量检查。

ignore-xml-invalid 跳过“XML 语法”质量检查。

ignore-zero-width-space 跳过“零宽空格”质量检查。

ignore-ellipsis 跳过“省略号”质量检查。

ignore-long-untranslated 跳过“长期未翻译”质量检查。

ignore-multiple-failures 跳过“多项检查失败”质量检查。

ignore-unnamed-format 跳过“多个未命名的变量”质量检查。

ignore-optional-plural 跳过“未复数化”质量检查。

注解： 通常规则是，任何检查都使用识别文字来命名 `ignore-*` ，所以能够将规则应用在定制检查中。

根据源字符串的设置，在*Component configuration* 设置中，并且在翻译文件自身中（例如在 GNU gettex 中），能够理解这些标记。

2.12.3 强制检查

3.11 新版功能.

可以通过设置*Component configuration* 中的 *Enforced checks* ，来配置不能省略的检查列表。列出的每个检查在用户界面中都不能省略，并且检查失败的任何字符串都被标记为 *Needs editing* （请见翻译状态）。

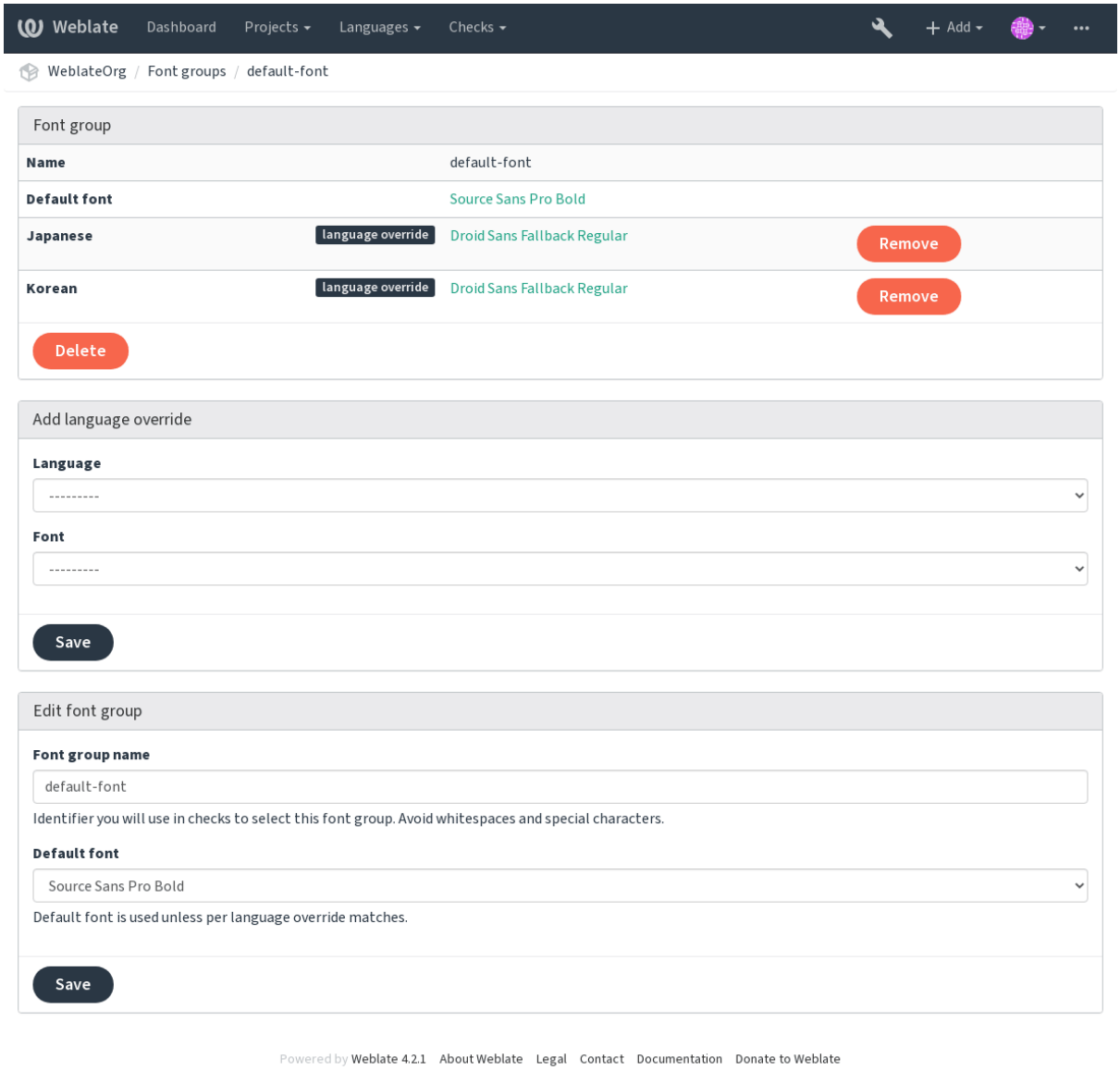
2.12.4 管理字型

3.7 新版功能.

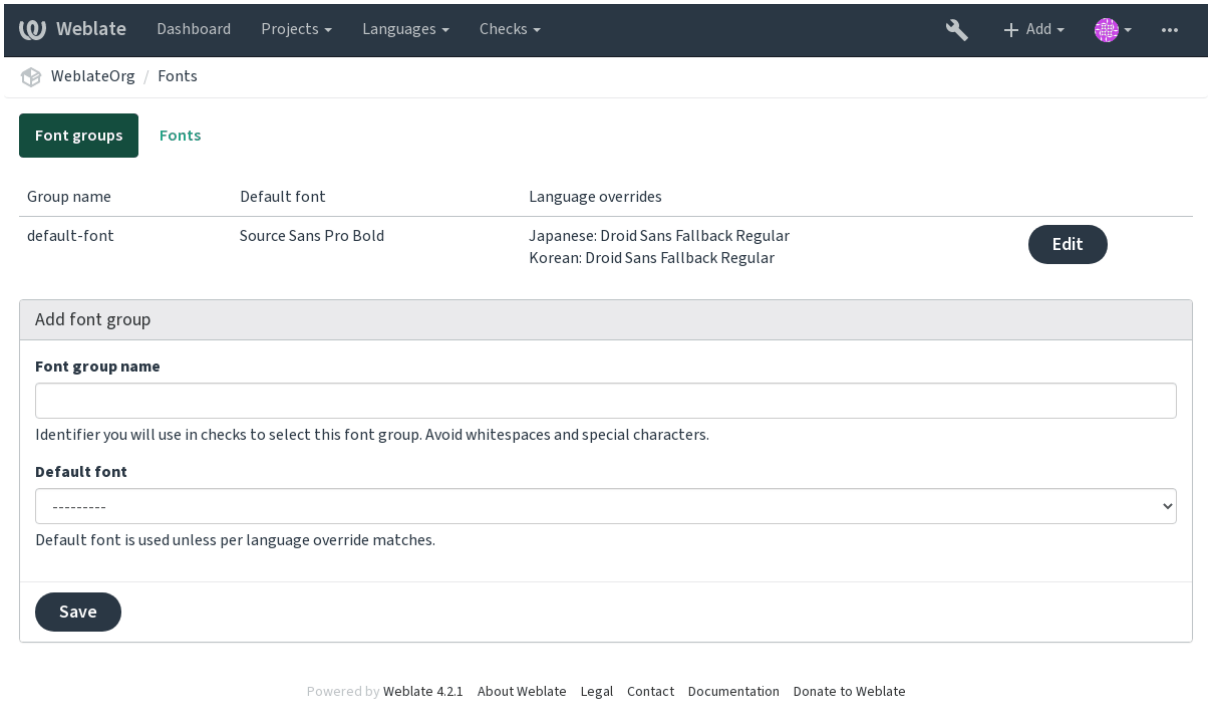
用于计算提交文本尺寸的最大翻译大小 检查，需要选择字型信息，这可以在翻译项目菜单 *Manage* 下的 Weblate 字型管理工具 *Fonts* 中实现。

可以上传 TrueType 或 OpenType 字型，设置字型组，并检查中使用。

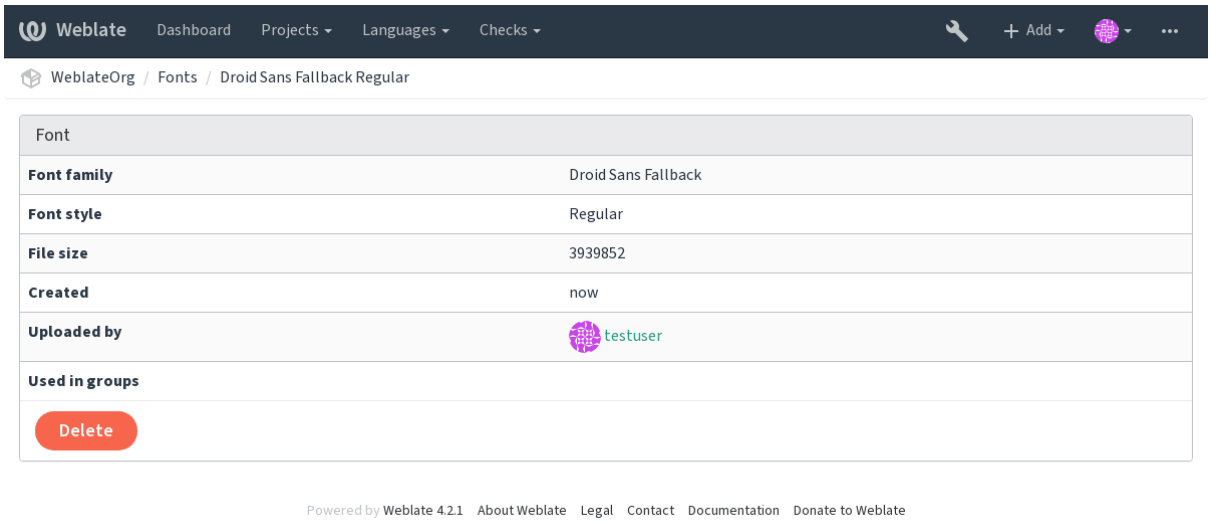
字型组允许为不同语言确定不同字型，这是非拉丁语言中典型需要的：



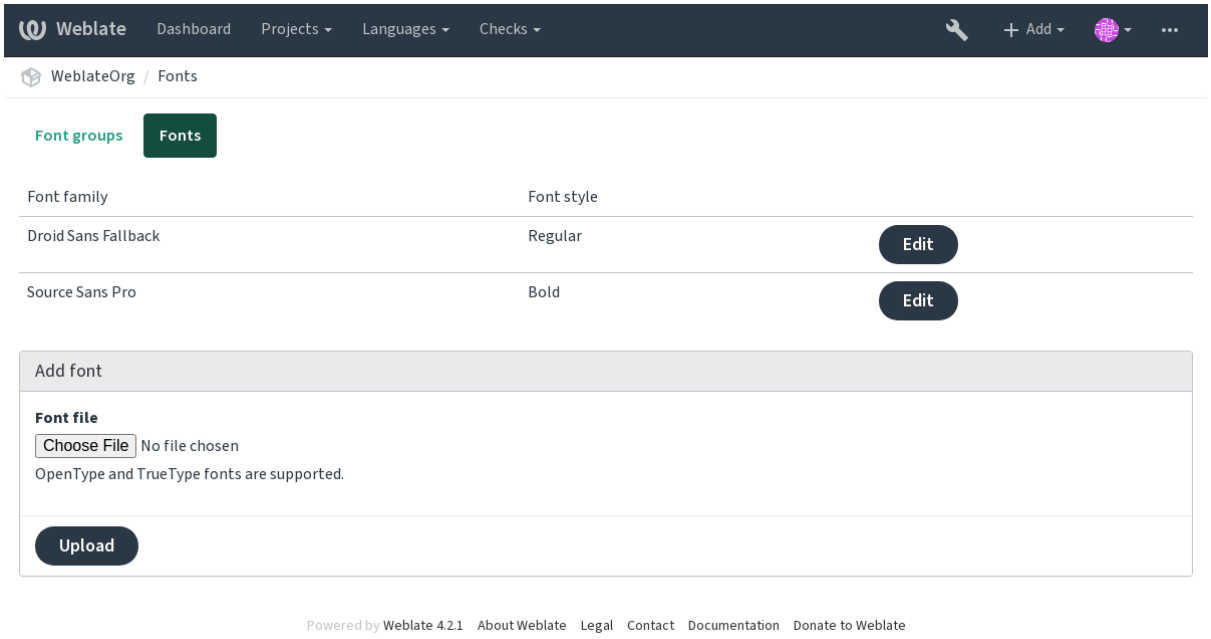
字型组通过名称识别，名称不能包含空白字符或特殊字符，这使它能够容易地用在检查定义中：



字型族和样式在上传后自动识别：



可以将集中字型上传到 Weblate 中：



为了使用字型来检查字符串长度，将适当的标记传递给它（请见定制行为）。可能会需要后面这些：

max-size:500 确定最大宽度。

font-family:ubuntu 确定字型组，通过指定其识别文字来使用。

font-size:22 确定字号。

2.12.5 编写自己的检查

Weblate 内建了很大范围的质量检查，（请见质量检查），尽管可能没有覆盖想要检查的所有内容。可以使用*CHECK_LIST* 来调整执行检查的列表，也可以添加定制的检查。

1. 子类 *weblate.checks.Check*
2. 设置一些属性。
3. 应用 `check` （如果想要处理代码中的复数的话）或 `check_single` 方法（它将为您完成）。

一些例子：

为了安装定制的检查，在*CHECK_LIST* 中为 Python 类提供完全合格的路径，请见定制的质量检查、插件和自动修复 。

检查翻译文本不含有“foo ”

这是非常简单的检查，只检查翻译中是否丢失了字符串“foo ”。

```
#
# Copyright © 2012 - 2020 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,
```

(下页继续)

(续上页)

```
# 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
```

检查捷克语翻译文本的复数差异

使用语言信息来检查，确定捷克语中的两种复数形式不同。

```
#
# Copyright © 2012 - 2020 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."""

from django.utils.translation import gettext_lazy as _

from weblate.checks.base import TargetCheck


class PluralCzechCheck(TargetCheck):
```

(下页继续)

(续上页)

```
    # 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 机器翻译

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 项目配置.

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*.

参见:

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*.

参见:

MT_APERTIUM_APY, Apertium website, Apertium APy documentation

2.13.3 AWS

3.1 新版功能.

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.

参见:

MT_AWS_REGION, *MT_AWS_ACCESS_KEY_ID*, *MT_AWS_SECRET_ACCESS_KEY* Amazon Translate Documentation

2.13.4 Baidu API machine translation

3.2 新版功能.

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*.

参见:

MT_BAIDU_ID, *MT_BAIDU_SECRET* Baidu Translate API

2.13.5 DeepL

2.20 新版功能.

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*.

提示: 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). You can toggle this by *MT_DEEPL_API_VERSION*.

参见:

MT_DEEPL_KEY, *MT_DEEPL_API_VERSION*, DeepL website, DeepL pricing, DeepL API documentation

2.13.6 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*.

参见:

Glosbe website

2.13.7 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*.

参见:

MT_GOOGLE_KEY, Google translate documentation

2.13.8 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*.

参见:

MT_GOOGLE_CREDENTIALS, *MT_GOOGLE_PROJECT*, *MT_GOOGLE_LOCATION* Google translate documentation

2.13.9 Microsoft Cognitive Services Translator

2.10 新版功能.

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.

参见:

MT_MICROSOFT_COGNITIVE_KEY, *MT_MICROSOFT_REGION*, Cognitive Services - Text Translation API, Microsoft Azure Portal

2.13.10 Microsoft Terminology Service

2.19 新版功能.

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*.

参见:

Microsoft Terminology Service API

2.13.11 ModernMT

4.2 新版功能.

Turn this service on by adding `weblate.machinery.modernmt.ModernMTTranslation` to *MT_SERVICES* and configure *MT_MODERNMT_KEY*.

参见:

ModernMT API, *MT_MODERNMT_KEY*, *MT_MODERNMT_URL*

2.13.12 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*.

参见:

MT_MYMEMORY_EMAIL, *MT_MYMEMORY_USER*, *MT_MYMEMORY_KEY*, MyMemory website

2.13.13 NetEase Sight API machine translation

3.3 新版功能.

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*.

参见:

MT_NETEASE_KEY, *MT_NETEASE_SECRET* Netease Sight Translation Platform

2.13.14 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/'
```

参见:

MT_TMSERVER, tmserver Installing amaGama, Amagama, Amagama Translation Memory

2.13.15 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*.

参见:

MT_YANDEX_KEY, Yandex Translate API, Powered by Yandex.Translate

2.13.16 Youdao Zhiyun API machine translation

3.2 新版功能.

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*.

参见:

MT_YOUDAO_ID, *MT_YOUDAO_SECRET* Youdao Zhiyun Natural Language Translation Service

2.13.17 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*.

2.13.18 Weblate Translation Memory

2.20 新版功能.

The 翻译记忆库 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.

2.13.19 SAP Translation Hub

Machine translation service provided by SAP.

You need to have a SAP account (and enabled the SAP Translation Hub 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 sandbox or the productive API.

注解： 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*.

参见：

MT_SAP_BASE_URL, *MT_SAP_SANDBOX_APIKEY*, *MT_SAP_USERNAME*, *MT_SAP_PASSWORD*, *MT_SAP_USE_MT* SAP Translation Hub API

2.13.20 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 - 2020 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
```

(下页继续)

(续上页)

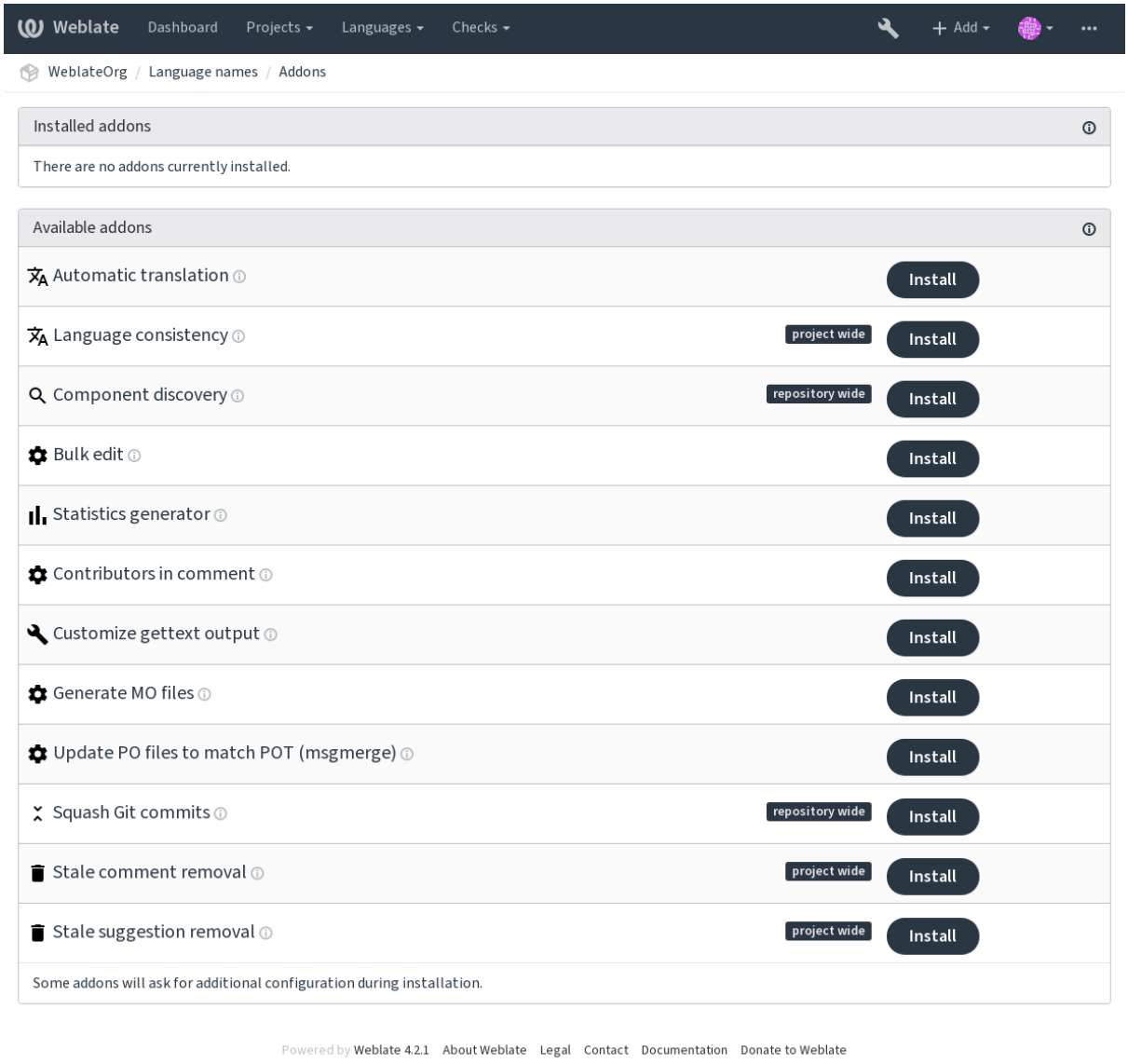
```
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Machine translation example."""

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, unit, user, search):
        """Return tuple with translations."""
        for t in dictionary.translate(text):
            yield {"text": t, "quality": 100, "service": self.name, "source": text}
```

You can list own class in *MT_SERVICES* and Weblate will start using that.

2.14 附加组件

2.19 新版功能.

附加组件提供了自定义翻译工作流的方法。它们可以被安装在翻译组件视图中，并在幕后工作。管理员可从各翻译组件的: 图形用户界面标签: ‘管理‘↓’附加组件’菜单中管理它们。



2.14.1 内置插件

自动化翻译

3.9 新版功能.

使用机器翻译或其他组件自动翻译字符串。

当组件中出现新字符串时，会自动触发此附加组件。

参见:

自动化翻译, *在部件之间保持翻译一致*

JavaScript 本地化 CDN

4.2 新版功能.

添加用于 JavaScript 或 HTML 本地化的本地化 CDN。

它可以用于将静态 HTML 网页本地化，或者用于在 JavaScript 代码中导入本地化文件。

在安装时，插件为您的组件产生唯一的 URL ，您可以将其包括在 HTML 文件中，使它们本地化。细节请见 *weblate-cdn* 。

参见:

正在配置 Weblate 内容分发网络附加组件, *Translating HTML and JavaScript using Weblate CDN*, *Weblate 内容分发网络的字符串提取*, *使用 Weblate 内容分发网络对 HTML 进行本地化操作*

清理翻译文件

更新所有翻译文件以匹配单语种译文模版文件。对于大多数文件格式来说，这意味着移除译文模版文件中不再出现的旧翻译条目。

语言一致性

确保一个项目中的所有组件都为每种要添加的语言进行翻译。

它对未添加到组件中的语言建立空的翻译。

每 24 小时，和在新的语言加入 Weblate 时，检查一次丢失的语言。

不像其他多数附加组件，这个附加组件影响整个项目。

提示: 用*自动化翻译* 自动翻译新添加的字符串。

组件发现

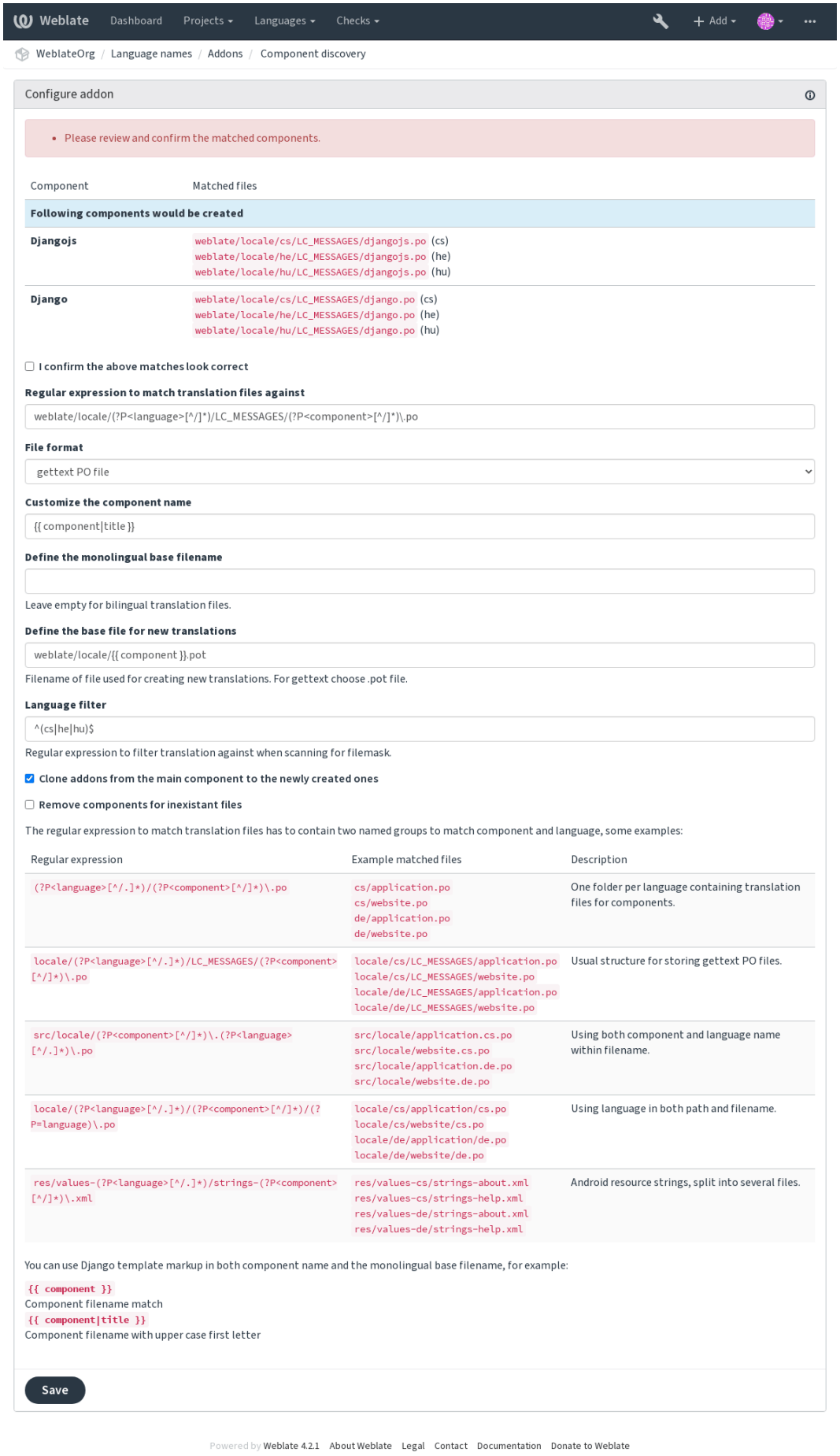
根据版本控制系统中文件更改的情况来自动添加或删除项目的组件.

每次版本控制系统（VCS ）升级时触发，否则与`import_project` 管理命令相似。这种方式可以在一个版本控制系统（VCS ）中跟踪多个翻译组件。

建立一个至少未来不大可能会消失的主要组件，其他会采用其*Weblate internal URLs* 作为版本管理系统（VCS ）的配置，并且配置它来找到其中的所有组件。

匹配是使用正则表达式完成的，在正则表达式中，强大的功能是配置复杂性的一种折衷。一些常见用例的例子可以在附加组件帮助部分找到。

一旦点击了 *Save* ，将显示匹配组件的预览，可以检查配置是否匹配于自己的需要:


Weblate
Dashboard
Projects
Languages
Checks
Add
WeblateOrg / Language names / Addons / Component discovery
Configure addon
Please review and confirm the matched components.
Component
Matched files
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/cs/LC_MESSAGES/django.po (cs)
weblate/locale/he/LC_MESSAGES/django.po (he)
weblate/locale/hu/LC_MESSAGES/django.po (hu)
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 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 4.2.1 About Weblate Legal Contact Documentation Donate to Weblate

参见:

Template markup

批量编辑

3.11 新版功能.

批量编辑字符串标志、标签或状态。

新字符串自动贴标签会有用（通过搜索查询 `NOT has:label` 开始，并且添加所需要的标签，直到所有的字符串都适当地被贴上标签）。还以对 Weblate 元数据执行其他任何自动化操作。

参见:

批量编辑

将未更改的译文标记为“需要编辑”

3.1 新版功能.

每当新的翻译字符串从版本控制系统（VCS ）中导入时，它将在 Weblate 中被标记为需要编辑。这对于包含全部字符串的文件格式十分有用，即使它们没有被翻译。

将新的源字符串标记为“需要编辑”

每当一个新的源字符串从版本控制系统（VCS ）中导入时，它将在 Weblate 中被标记为需要编辑。这样您就可以简单地筛选并编辑开发者编写的源字符串。

将导入的新译文标记为“需要编辑”

每当一个新的可翻译字符串从版本控制系统（VCS ）中导入时，它将在 Weblate 中被标记为需要编辑。这样您就可以简单地筛选并编辑开发者创建的翻译。

统计数据生成器

生成一个包含翻译详细信息的文件。

可以在文件名和内容中使用 Django 模板，参见*Template markup* 的具体标记表述。

例如对每个翻译产生摘要文件：

所生成文件的名称 `locale/{{ language_code }}.json`

内容

```
{
    "language": "{{ language_code }}",
    "strings": "{{ stats.all }}",
    "translated": "{{ stats.translated }}",
    "last_changed": "{{ stats.last_changed }}",
    "last_author": "{{ stats.last_author }}",
}
```

参见:

Template markup

在注释中添加贡献信息

在 PO 文件的头部以注释的形式添加贡献者的名字与贡献年份。

PO 文件头包含贡献者与贡献年份的列表：

```
# 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.
```

更新“配置文件”中的 ALL_LINGUAS 变量

当新的翻译添加时，更新 `configure` 、`configure.in` 或任何 `configure.ac` 文件中的 ALL_LINGUAS 变量。

自定义 gettext 输出

允许自定义 gettext 的输出行为，例如是否启用自动换行。

提供了后面的选项：

- 在 77 个字符处和新一行时来换行
- 仅在换行符处折行
- 不换行

注解： 默认 gettext 在 77 个字符处和新一行时换行。使用 `--no-wrap` 参数时只在新一行时换行。

更新 LINGUAS 文件

添加新翻译时更新 LINGUAS 文件。

生成 MO 文件

为每个变更的 PO 文件自动生成 MO 文件。

更新 PO 文件以匹配 POT 文件 (msgmerge)

使用 msgmerge 更新所有的 PO 文件以匹配 POT 文件。会在每次拉取上游代码库的更新时触发。

压缩 Git 提交

推送变更之前压缩 Git 提交。

可以选择后面的模式之一：

3.4 新版功能.

- 所有提交成一个
- 每种语言
- 每个文件

3.5 新版功能.

- 每个作者

原始提交信息保留，但其作者信息丢失，除非选择了“Per author”，或者定制提交信息来包括它。

4.1 新版功能.

原始提交信息可选地由定制的提交信息覆盖。

预告（提交行像 `Co-authored-by: ...`）可选地从原始提交信息中去掉，并且添加在去掉的提交信息后面。这还可以为每一位翻译者产生适当的 `Co-authored-by:` 信誉。

自定义 JSON 输出

允许调整 JSON 的输出行为，例如缩进或排序。

格式化 Java 属性文件

排序 Java 属性文件。

陈旧注释删除

3.7 新版功能.

设置删除注释的时间。

这可以用于删除可能过时的陈旧评论。小心使用，因为陈旧评论不意味着失去了重要性。

陈旧建议删除

3.7 新版功能.

设置删除建议的时间。

可以用于连接建议投票（请见同行评审），将给定时间内没有得到足够积极票数的建议删除。

更新 RESX 文件

3.9 新版功能.

更新所有翻译文件以匹配上游单语种译文模版文件。未使用的字符串将被删除，新字符串将复制源字符串添加。

提示： 如果只想删除陈旧的翻译键，那么使用清理翻译文件。

自定义 YAML 输出

3.10.2 新版功能.

允许调整 YAML 的输出，例如自定义缩进或自定义换行。

2.14.2 定制附加组件列表

附加组件列表由*WEBLATE_ADDONS*配置。要添加其他附加组件，只需在这个设置中包含类绝对名称即可。

2.14.3 编写附加组件

你也可以编写自己的附加组件，所需要做的是子类 `BaseAddon`，定义附加组件的元数据，并实现一个会执行处理的回调。

这是一个附加组件示例：

```
#
# Copyright © 2012 - 2020 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 addon should receive
    events = (EVENT_PRE_COMMIT,)
    # Addon unique identifier
    name = "weblate.example.example"
    # Verbose name shown in the user interface
    verbose = _("Example addon")
    # Detailed addon description
    description = _("This addon does nothing it is just an example.")

    # Callback to implement custom behavior
    def pre_commit(self, translation, author):
        return
```

2.14.4 从附加组件执行脚本

附加逐渐还可以用于执行外部脚本。这曾经集成在 Weblate 中，但现在必须写一些代码，将脚本包裹在附加组件中。

```
#
# Copyright © 2012 - 2020 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/>.
#
"""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"
```

安装方法请见定制的质量检查、插件和自动修复 。

对于任何给定的组件，当前路径设置为版本控制系统（VCS ）仓库的根目录时，执行脚本。

此外，可以访问后面的环境参数：

WL_VCS
使用的版本控制系统。

WL_REPO
上游仓库 URL 。

WL_PATH
版本控制系统（VCS ）仓库的绝对路径。

WL_BRANCH
2.11 新版功能.

当前组件配置的仓库分支。

WL_FILEMASK
当前组件的 Filemask。

WL_TEMPLATE
单语言翻译模板的文件名（可以为空）。

WL_NEW_BASE
2.14 新版功能.

建立新的翻译所使用文件的文件名（可以为空）。

WL_FILE_FORMAT
当前组件使用的文件格式。

WL_LANGUAGE
当前处理的翻译语言（对于组件水平的 hook 不可获得）。

WL_PREVIOUS_HEAD
之前更新上的 HEAD （只有当运行过去的更新 hook 时可获得）。

WL_COMPONENT_SLUG
3.9 新版功能.

组件标识串用于构建 URL 。

WL_PROJECT_SLUG
3.9 新版功能.

项目标识串用于构建 URL 。

WL_COMPONENT_NAME
3.9 新版功能.

组件名称。

WL_PROJECT_NAME
3.9 新版功能.

项目名称。

WL_COMPONENT_URL
3.9 新版功能.

组件 URL 。

WL_ENGAGE_URL
3.9 新版功能.

项目参与 URL 。

参见:

Component configuration

以往的更新仓库处理

以往更新仓库处理可以用于在版本控制系统（VCS ）上游源改变时，更新翻译文件。为了实现这个功能，请记住 Weblate 只看到提交给版本控制系统（VCS ）的文件，所以需要同意更改作为脚本的一部分。

例如 Gulp ，可以使用后面的代码来执行：

```
#! /bin/sh
gulp --gulpfile gulp-i18n-extract.js
git commit -m 'Update source strings' src/languages/en.lang.json
```

翻译的预提交处理

使用提交脚本在提交给仓库前自动地对翻译做出更改。

它作为组成当前翻译文件名的单一参数而通过。

2.15 翻译记忆库

2.20 新版功能.

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, 机器翻译 view while translating.
- Automatically, by translating strings using 自动化翻译, or 自动化翻译 addon.

For installation tips, see *Weblate Translation Memory*, which is turned on by default.

2.15.1 Translation memory scopes

3.2 新版功能: 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.

共享翻译记忆库

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

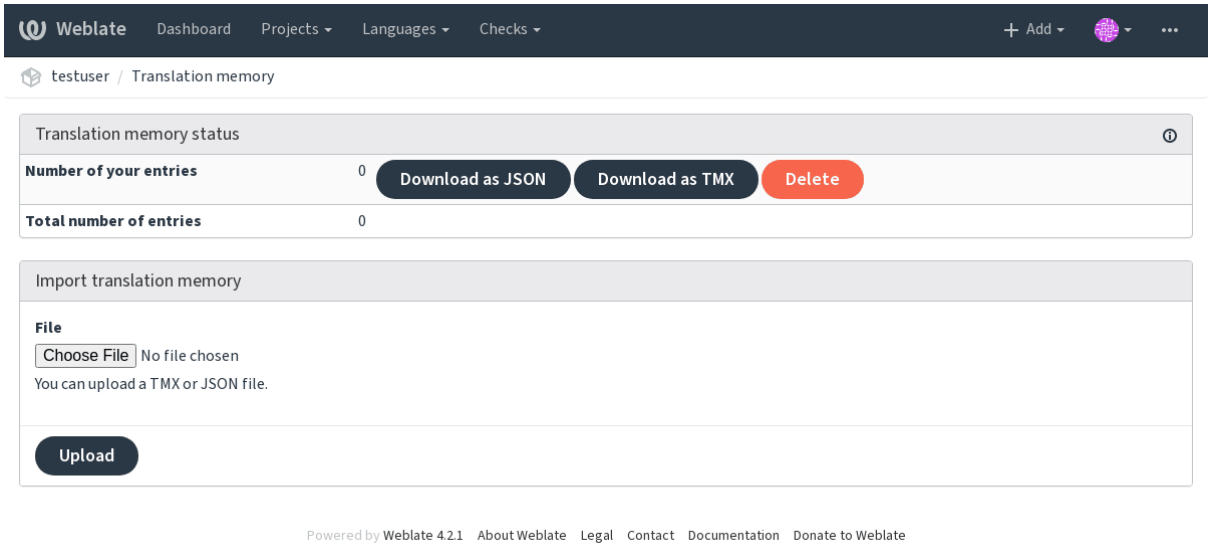
3.2 新版功能.

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.

提示： Translation memory in JSON can be imported into Weblate, TMX is provided for interoperability with other tools.

参见：

Weblate Translation Memory Schema



管理界面

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 配置

所有的设置存储在 `settings.py` 中，（如 Django 通常那样）。

注解： 在更改这些设置的任何一部分后，需要重新启动 Weblate ——WSGI 和 Celery 两个过程。

在它作为 mod_wsgi 运行的情况下，需要重新启动 Apache ，来重新载入配置。

参见：

还要请查看 Django’s documentation 中关于配置 Django 自身的参数。

2.16.1 AKISMET_API_KEY

Weblate 可以使用 Akismet 检查到来的对垃圾邮件的匿名建议。请访问 akismet.com 来购买 API 密钥，并将它与网站相联系。

2.16.2 ANONYMOUS_USER_NAME

未登录用户的用户名。

参见:

访问控制

2.16.3 AUDITLOG_EXPIRY

3.6 新版功能.

Weblate 应该将审计日志保存多少天，审计日志包括了账户活动的信息。

默认为 180 天。

2.16.4 AUTH_LOCK_ATTEMPTS

2.14 新版功能.

在 rate 限制应用前，授权尝试失败的最多次数。

当前，这应用在后面的位置:

- 登录。删除账户密码，防止用户不请求新的密码而登录。
- 密码重置。防止发出新的电子邮件，避免向用户发出太多密码重置尝试的垃圾邮件。

默认为 10 。

参见:

Rate limiting ,

2.16.5 AUTO_UPDATE

3.2 新版功能.

在 3.11 版更改: 改变原来的开关选项，来区别接受哪个字符串。

以每天的频率更新所有仓库。

提示: 在不使用通知钩子 来自动更新 Weblate 仓库的情况下有用。

注解: 除了字符串选项还存在开关选项，用于向后兼容。

选项有:

"none" 不进行每日更新。

"remote" 也是 False 只进行远程更新。

"full" 也是 True 更新远程，并合并工作副本。

注解： 这需要*Background tasks using Celery* 工作，并在重启后生效。

2.16.6 AVATAR_URL_PREFIX

构成头像 URL 的前缀为：`${AVATAR_URL_PREFIX}/avatar/${MAIL_HASH}?${PARAMS}` 。已知后面的服务工作：

Gravatar （默认），根据 https://gravatar.com/ `AVATAR_URL_PREFIX = 'https://www.gravatar.com/'`

Libravatar ，根据 https://www.libravatar.org/ `AVATAR_URL_PREFIX = 'https://www.libravatar.org/'`

参见：

头像缓存, *ENABLE_AVATARS*, *Avatars*

2.16.7 AUTH_TOKEN_VALID

2.14 新版功能.

身份验证令牌和密码重置电子邮件中临时密码的有效时间。以秒为单位，默认为 172800 （2 天）。

2.16.8 AUTH_PASSWORD_DAYS

2.15 新版功能.

应该允许几天来使用相同的密码。

注解： Weblate 2.15 版本之前的密码更改不遵从这个原则。

默认为 180 天。

2.16.9 AUTOFIX_LIST

当存储字符串时应用自动修复列表。

注解： 为应用自动修复见面的 Python 类提供完全合规的路径。

可用的修复：

weblate.trans.autofixes.whitespace.SameBookendingWhitespace 将字符串开始与结尾的空白字符与元字符串匹配。

weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis 如果字符串有省略号的话（…），用来连续的点符号（...）代替。

weblate.trans.autofixes.chars.RemoveZeroSpace 去掉零宽度字符，如果源字符串不包含的话。

weblate.trans.autofixes.chars.RemoveControlChars 去掉控制字符，如果源字符串不包含的话。

weblate.trans.autofixes.html.BleachHTML 从标记为 `safe-html` 的字符串中去掉不安全的 HTML 标记（请见*不安全的 HTML 网站* ）。

可以选择使用哪一个：

```
AUTOFIX_LIST = (
    'weblate.trans.autofixes.whitespace.SameBookendingWhitespace',
    'weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis',
)
```

参见：

自动修正, 定制的自动修正

2.16.10 BASE_DIR

Weblate 源所在的基本目录。用于默认得到几个其他路径：

- *DATA_DIR*

默认值：Weblate 源的顶层目录。

2.16.11 CSP_SCRIPT_SRC, CSP_IMG_SRC, CSP_CONNECT_SRC, CSP_STYLE_SRC, CSP_FONT_SRC

为 Weblate 定制 `Content-Security-Policy` 标头。根据允许集成的第三方服务（Matomo 、Google Analytics 、Sentry ……）来自动生成标头。

这些默认为空列表。

** 示例:: **

```
# Enable Cloudflare Javascript optimizations
CSP_SCRIPT_SRC = ["ajax.cloudflare.com"]
```

参见：

Content security policy, Content Security Policy (CSP)

2.16.12 CHECK_LIST

翻译时执行的质量检查列表。

注解： 为实施检查界面的 Python 类提供完全合规的路径。

调整检查列表，来包括与您相关的那些检查。

所有内建的质量检查 默认都打开，可以从那里更改设置。它们在*Sample configuration* 中被默认注释掉，从而使用默认值。然后每个新的 Weblate 版本执行新的检查。

可以关闭所有检查：

```
CHECK_LIST = ()
```

可以只打开一部分检查：

```
CHECK_LIST = (
    'weblate.checks.chars.BeginNewlineCheck',
    'weblate.checks.chars.EndNewlineCheck',
    'weblate.checks.chars.MaxLengthCheck',
)
```

注解： 更改这些设置只影响新更改的翻译，现存的检查仍然存储在数据库中。为了将更改同样应用到存储的翻译中，运行*updatechecks* 。

参见：

质量检查, 定制行为

2.16.13 COMMENT_CLEANUP_DAYS

3.6 新版功能.

在一定天数后删除评论。默认为 None ，意味着不删除。

2.16.14 COMMIT_PENDING_HOURS

2.10 新版功能.

通过后台任务方式执行的将更改挂起之间的小时数。

参见：

Component configuration, 对变更进行提交的延时时间, *Running maintenance tasks*, *commit_pending*

2.16.15 DATA_DIR

Weblate 文件夹将所有数据存储其中。它包含到 VCS 仓库的链接，全文本索引和外部工具的各种文件。

后面的子目录通常存在：

home Home 目录用于调用脚本。

ssh SSH 密钥和配置。

static 静态 Django 文件的默认位置，由 STATIC_ROOT 指定。

media Django 媒体文件的默认位置，由 MEDIA_ROOT 指定。

vcs 版本控制仓库。

backups 每日备份数据，细节请参考下载的数据用于备份 。

注解： 这个目录必须由 Weblate 写入。运行作为 uWSGI 意味着 www-data 用户应该对它具有写入权限。

实现这个的最简单方式是使用户作为目录的所有者：

```
sudo chown www-data:www-data -R $DATA_DIR
```

默认到 $BASE_DIR/data 。

参见：

BASE_DIR, 备份和移动 *Weblate*

2.16.16 DATABASE_BACKUP

3.1 新版功能.

数据库备份应该存储为纯文本，压缩还是跳过。授权值为：

- `"plain"`
- `"compressed"`
- `"none"`

参见：

备份和移动 Weblate

2.16.17 DEFAULT_ACCESS_CONTROL

3.3 新版功能.

新项目的默认访问控制设置：

`0` *Public*

`1` *Protected*

`100` *Private*

`200` *Custom*

如果手动管理 ACL 则使用 *Custom* ，这意味着不依赖于 Weblate 内部管理。

参见：

根据项目的访问控制, *访问控制*, *访问控制*

2.16.18 DEFAULT_RESTRICTED_COMPONENT

4.1 新版功能.

组件限制的默认值。

参见：

根据项目的访问控制, *Restricted access*, *访问控制*

2.16.19 DEFAULT_ADD_MESSAGE, DEFAULT_ADDON_MESSAGE, DEFAULT_COMMIT_MESSAGE, DEFAULT_DELETE_MESSAGE, DEFAULT_MERGE_MESSAGE

不同操作的默认执行信息，细节请查阅*Component configuration* 。

参见：

Template markup, *Component configuration*, *Commit, add, delete, merge and addon messages*

2.16.20 DEFAULT_ADDONS

安装在每个创建的组件上的默认附加组件。

注解： 此设置只影响新创建的组件。

例：

```
DEFAULT_ADDONS = {
     # Addon with no parameters
     "weblate.flags.target_edit": {},

     # Addon with parameters
     "weblate.autotranslate.autotranslate": {
         "mode": "suggest",
         "filter_type": "todo",
         "auto_source": "mt",
         "component": "",
         "engines": ["weblate-translation-memory"],
         "threshold": "80",
     }
}
```

参见：

install_addon

2.16.21 DEFAULT_COMMITER_EMAIL

2.4 新版功能.

已创建的翻译组件的提交者电子邮件地址，默认为 `noreply@weblate.org` 。

参见：

DEFAULT_COMMITER_NAME, *Component configuration*, 提交者邮箱

2.16.22 DEFAULT_COMMITER_NAME

2.4 新版功能.

建立翻译部件的执行者姓名默认为 `Weblate` 。

参见：

DEFAULT_COMMITER_EMAIL, *Component configuration*, 提交者姓名

2.16.23 DEFAULT_MERGE_STYLE

3.4 新版功能.

任何新组件的合并风格。

- *rebase* - default
- *merge*

参见：

Component configuration, 合并方式

2.16.24 DEFAULT_TRANSLATION_PROPAGATION

2.5 新版功能.

翻译传播的默认设置，默认为 `True` 。

参见:

Component configuration, *允许同步翻译*

2.16.25 DEFAULT_PULL_MESSAGE

Pull Request 的标题，默认为 `'Update from Weblate'` 。

2.16.26 ENABLE_AVATARS

是否为用户打开基于 Gravatar 的头像。默认打开。

头像取出并缓存在从服务器中，降低了泄漏个人信息的风险，加速了用户体验。

参见:

头像缓存, *AVATAR_URL_PREFIX*, *Avatars*

2.16.27 ENABLE_HOOKS

是否允许匿名远程钩子。

参见:

通知钩子

2.16.28 ENABLE_HTTPS

将链接作为 HTTPS 还是 HTTP 发送给 Weblate。这个设置影响发送电子邮件，并产生绝对 URL。

提示: 在默认配置中，这还用于与 HTTPS 相关的几个 Django 设置。

参见:

SESSION_COOKIE_SECURE, CSRF_COOKIE_SECURE, SECURE_SSL_REDIRECT, *Set correct site domain*

2.16.29 ENABLE_SHARING

打开/关闭:guilabel: *Share* 菜单，使用户能够将翻译过程分享到社交网络上。

2.16.30 GITLAB_USERNAME

GitLab 用户名，用于发送翻译更新的和并请求。

参见:

GitLab, *Setting up Lab*

2.16.31 GITHUB_USERNAME

GitHub 用户名，用于发送翻译更新的 Pull Request 。

参见:

GitHub, *Setting up hub*

2.16.32 GOOGLE_ANALYTICS_ID

谷歌分析 ID ，开启使用谷歌分析对 Weblate 的监控。

2.16.33 HIDE_REPO_CREDENTIALS

隐藏仓库证明避免出现在 web 网站界面中。在仓库 URL 带有用户名和密码的情况下，Weblate 会在相关信息显示给用户时将其隐藏起来。

例如除了 `https://user:password@git.example.com/repo.git` 会只显示 `https://git.example.com/repo.git` 。它也试图以相似的方式清除 VCS 错误信息。

注解: 默认这是打开的。

2.16.34 IP_BEHIND_REVERSE_PROXY

2.14 新版功能.

指示 Weblate 是否在反向代理后面运行。

如果设置为 `True` ， Weblate 会从:setting:‘IP_PROXY_HEADER‘定义的标头中得到 IP 地址。

警告: 确保真正使用反向代理，并且设置了这个标头，否则用户将能够假冒 IP 地址。

注解: 默认这不是打开的。

参见:

Running behind reverse proxy, *Rate limiting*, *IP_PROXY_HEADER*, *IP_PROXY_OFFSET*

2.16.35 IP_PROXY_HEADER

2.14 新版功能.

指示当*IP_BEHIND_REVERSE_PROXY* 打开时，Weblate 应该从那个标头得到 IP 地址。

默认为 `HTTP_X_FORWARDED_FOR` 。

参见:

Running behind reverse proxy, *Rate limiting*, `SECURE_PROXY_SSL_HEADER`, *IP_BEHIND_REVERSE_PROXY*, *IP_PROXY_OFFSET*

2.16.36 IP_PROXY_OFFSET

2.14 新版功能.

指示*IP_PROXY_HEADER* 的哪部分用作客户端 IP 地址。

依赖于您的设置，这个标头会包括几个 IP 地址，（例如 `X-Forwarded-For: a, b, client-ip` ），并且可以配置标头的哪个地址在这里用作客户端 IP 地址。

警告: 设置这个会影响您安装的安全性，您应该只配置它来使用信任的代理来确定 IP 地址。

默认为 0 。

参见:

Running behind reverse proxy, *Rate limiting*, `SECURE_PROXY_SSL_HEADER`, *IP_BEHIND_REVERSE_PROXY*, *IP_PROXY_HEADER*

2.16.37 LEGAL_URL

3.5 新版功能.

您的 Weblate 事例显示其法律文件的 URL 。

提示: 在您将您的法律文件保存在 Weblate 之外，而将其嵌入 Weblate 的情况下有用。细节请查看*法律声明* 。

例:

```
LEGAL_URL = "https://weblate.org/terms/"
```

2.16.38 LICENSE_EXTRA

包括在许可选择中的其他许可。

注解: 每个许可的定义应该是其短名称、长名称和 URL 的元组。

例如:

```
LICENSE_EXTRA = [
    (
        "AGPL-3.0",
        "GNU Affero General Public License v3.0",
        "https://www.gnu.org/licenses/agpl-3.0-standalone.html",
    ),
]
```

2.16.39 LICENSE_FILTER

用于显示的许可以外的选项。

注解： 这个过滤器使用了短许可名称。

例如：

```
LICENSE_FILTER = {"AGPL-3.0", "GPL-3.0-or-later"}
```

2.16.40 LICENSE_REQUIRED

定义了是否需要*Component configuration* 中的许可属性。

注解： 默认这是关闭的。

2.16.41 LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH

给定翻译的长度是否应被限制。限制为源字符串的长度 * 10 个字符。

提示： 将其设置为 `False` 来允许更长的翻译，而不管源字符串的长度。

注解： 默认为 `True` 。

2.16.42 LOCALIZE_CDN_URL and LOCALIZE_CDN_PATH

这些设置配置了*JavaScript 本地化 CDN* 插件。*LOCALIZE_CDN_URL* 定义了可获得本地化 CDN 的根 URL，而*LOCALIZE_CDN_PATH* 定义了 Weblate 应该存储生成文件的路径，生成的文件在*LOCALIZE_CDN_URL* 使用。

提示： 在 hosted Weblate 中，这使用 `https://weblate-cdn.com/` 。

参见：

JavaScript 本地化 CDN

2.16.43 LOGIN_REQUIRED_URLS

您希望需要登录的 URL 列表。(除了 Weblate 内建立的标准规则)。

提示: 这允许密码保护整个安装，通过使用:

```
LOGIN_REQUIRED_URLS = (
    r'/(.*)$',
)
REST_FRAMEWORK["DEFAULT_PERMISSION_CLASSES"] = [
    "rest_framework.permissions.IsAuthenticated"
]
```

提示: 同样想要锁住 API 访问，如上面的例子所示。

2.16.44 LOGIN_REQUIRED_URLS_EXCEPTIONS

用于*LOGIN_REQUIRED_URLS*的例外列表。如果不指定，可以允许用户访问登录页。

您可能想要包括的一些例外:

```
LOGIN_REQUIRED_URLS_EXCEPTIONS = (
    r'/accounts/(.*)$', # Required for login
    r'/static/(.*)$',   # 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
)
```

2.16.45 MATOMO_SITE_ID

您想要跟踪的 Matomo （以前的 Piwik ）中的网站 ID。

注解: 这个集成不支持 Matomo Tag Manager 。

参见:

MATOMO_URL

2.16.46 MATOMO_URL

您想要跟踪的 Matomo (以前的 Piwik)安装的全 URL (包括反斜杠)。更多细节请查阅 <https://matomo.org/> 。

提示: 这个集成不支持 Matomo Tag Manager 。

例如:

```
MATOMO_SITE_ID = 1
MATOMO_URL = "https://example.matomo.cloud/"
```

参见:

MATOMO_SITE_ID

2.16.47 MT_SERVICES

在 3.0 版更改: 设置从 MACHINE_TRANSLATION_SERVICES 重命名为 MT_SERVICES ，而与其它机器翻译设置一致。

允许使用的机器翻译服务的列表。

注解: 很多服务需要像 API 密钥一样的另外的配置，细节请查阅其文件 ref: *machine* 。

```
MT_SERVICES = (
    'weblate.machinery.apertium.ApertiumAPYTranslation',
    'weblate.machinery.deepl.DeepLTranslation',
    'weblate.machinery.glosbe.GlosbeTranslation',
    'weblate.machinery.google.GoogleTranslation',
    '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',
)
```

参见:

机器翻译, 机器翻译

2.16.48 MT_APERTIUM_APY

Apertium-APy 服务器的 URL，https://wiki.apertium.org/wiki/Apertium-apy

参见:

Apertium, 机器翻译, 机器翻译

2.16.49 MT_AWS_ACCESS_KEY_ID

Amazon Translate 的访问密钥 ID 。

参见:

AWS, 机器翻译, 机器翻译

2.16.50 MT_AWS_SECRET_ACCESS_KEY

Amazon Translate 的 API 密钥。

参见:

AWS, 机器翻译, 机器翻译

2.16.51 MT_AWS_REGION

Amazon Translate 使用的区域名称。

参见:

AWS, 机器翻译, 机器翻译

2.16.52 MT_BAIDU_ID

百度智云 API 的客户端 ID ， 可以在 https://api.fanyi.baidu.com/api/trans/product/index 注册

参见:

Baidu API machine translation, 机器翻译, 机器翻译

2.16.53 MT_BAIDU_SECRET

百度智云 API 的客户端密钥，可以在 https://api.fanyi.baidu.com/api/trans/product/index 注册

参见:

Baidu API machine translation, 机器翻译, 机器翻译

2.16.54 MT_DEEPL_API_VERSION

4.1.1 新版功能.

DeepL 服务使用的 API 版本。版本限制了使用范围:

v1 意味着计算机辅助翻译工具，并且在基于用户的订阅时使用。

v2 意味着 API 应用，并且订阅是基于使用的。

此前 Weblate 被 DeepL 分类为计算机辅助翻译工具，因此应该使用 v1 API ，但现在应该使用 v2 API 。这样默认为 v2 ，您可以在现有计算机辅助翻译工具订阅，并想要 Weblate 使用它的情况下，将其更改为 v1 。

参见:

DeepL, 机器翻译, 机器翻译

2.16.55 MT_DEEPL_KEY

DeepL API 的 API 密钥，您可以在 https://www.deepl.com/pro.html 注册使用

参见:

DeepL, 机器翻译, 机器翻译

2.16.56 MT_GOOGLE_KEY

谷歌翻译 API v2 的 API 密钥，您可以在 https://cloud.google.com/translate/docs 上注册使用

参见:

Google Translate, 机器翻译, 机器翻译

2.16.57 MT_GOOGLE_CREDENTIALS

Google 云控制台中得到 API v3 JSON 证明文件。请提供 OS 全路径。证明根据服务账户而隶属于特定项目。更多细节请查看 https://cloud.google.com/docs/authentication/getting-started 。

2.16.58 MT_GOOGLE_PROJECT

API v3 Google 云 *project id* ，启动了翻译服务并且激活了账户。更多细节请查看 https://cloud.google.com/appengine/docs/standard/nodejs/building-app/creating-project

2.16.59 MT_GOOGLE_LOCATION

API v3 Google 云应用引擎会是特定于位置的。如果默认 `global` 退回对您不会起作用，则进行更改。

细节请查看 https://cloud.google.com/appengine/docs/locations

参见:

Google Translate API V3 (Advanced)

2.16.60 MT_MICROSOFT_BASE_URL

在 “Base URLs” 部分 定义的基于 URL 域名的区域。

Azure Global 的默认值为 `api.cognitive.microsofttranslator.com` 。

对于 Azure China ， 请使用 `api.translator.azure.cn` 。

2.16.61 MT_MICROSOFT_COGNITIVE_KEY

Microsoft Cognitive Services Translator API 客户端密匙。

参见:

Microsoft Cognitive Services Translator, 机器翻译, 机器翻译, Cognitive Services - Text Translation API, Microsoft Azure Portal

2.16.62 MT_MICROSOFT_REGION

在 “Multi service subscription” 中定义的区域前缀。

2.16.63 MT_MICROSOFT_ENDPOINT_URL

在 “Authenticating with an access token” 部分 定义的用于访问令牌的区域端点 URL 域名。

Azure Global 的默认值为 `api.cognitive.microsoft.com`。

对于 Azure 中国，请使用您的 Azure Portal 的端点。

2.16.64 MT_MODERNMT_KEY

ModernMT 机器翻译引擎的 API 密钥。

参见:

ModernMT *MT_MODERNMT_URL*

2.16.65 MT_MODERNMT_URL

ModernMT URL 默认值为 `https://api.modernmt.com/` 。

参见:

ModernMT *MT_MODERNMT_KEY*

2.16.66 MT_MYMEMORY_EMAIL

MyMemory 身份电子邮件地址。每天允许 1000 个请求。

参见:

MyMemory, 机器翻译, 机器翻译, MyMemory: API technical specifications

2.16.67 MT_MYMEMORY_KEY

MyMemory 访问密钥，用于私有翻译记忆，与*MT_MYMEMORY_USER* 一起使用。

参见:

MyMemory, 机器翻译, 机器翻译, MyMemory: API key generator

2.16.68 MT_MYMEMORY_USER

MyMemory 用户 ID ，用于私有翻译记忆，与:setting:‘MT_MYMEMORY_KEY‘一起使用。

参见:

MyMemory, 机器翻译, 机器翻译, MyMemory: API key generator

2.16.69 MT_NETEASE_KEY

NetEase Sight API 密匙，您可以在 https://sight.netease.com/ 注册使用

参见:

NetEase Sight API machine translation, 机器翻译, 机器翻译

2.16.70 MT_NETEASE_SECRET

NetEase Sight API 的密码，您可以在 https://sight.netease.com/ 上注册

参见:

NetEase Sight API machine translation, 机器翻译, 机器翻译

2.16.71 MT_TMSERVER

正在运行 TMServer 的 URL 。

参见:

tmserver, 机器翻译, 机器翻译, tmserver

2.16.72 MT_YANDEX_KEY

Yandex 翻译 API 的 API 密钥，您可以在 https://tech.yandex.com/translate/ 注册

参见:

Yandex Translate, 机器翻译, 机器翻译

2.16.73 MT_YOUDAO_ID

有道志云 API ID，您可以注册在 https://ai.youdao.com/product-fanyi-text.s 。

参见:

Youdao Zhiyun API machine translation, 机器翻译, 机器翻译

2.16.74 MT_YOUDAO_SECRET

有道智云 API 密匙，您可以在 https://ai.youdao.com/product-fanyi-text.s 注册。

参见:

Youdao Zhiyun API machine translation, 机器翻译, 机器翻译

2.16.75 MT_SAP_BASE_URL

SAP Translation Hub 服务的 API URL 。

参见:

SAP Translation Hub, 机器翻译, 机器翻译

2.16.76 MT_SAP_SANDBOX_APIKEY

sandbox API 使用的 API 密钥

参见:

SAP Translation Hub, 机器翻译, 机器翻译

2.16.77 MT_SAP_USERNAME

SAP 的用户名

参见:

SAP Translation Hub, 机器翻译, 机器翻译

2.16.78 MT_SAP_PASSWORD

您的 SAP 密码

参见:

SAP Translation Hub, 机器翻译, 机器翻译

2.16.79 MT_SAP_USE_MT

除了术语数据库，是否还使用机器翻译服务。可能值：`True` 或 `False`

参见:

SAP Translation Hub, 机器翻译, 机器翻译

2.16.80 NEARBY_MESSAGES

在查看当前翻译字符串时显示多少字符串。这只是默认值，用户可以在:ref:‘user-profile‘中调整。

2.16.81 RATELIMIT_ATTEMPTS

3.2 新版功能.

在应用次数限制前，身份认证尝试的最多次数。

默认为 5 。

参见:

Rate limiting, *RATELIMIT_WINDOW*, *RATELIMIT_LOCKOUT*

2.16.82 RATELIMIT_WINDOW

3.2 新版功能.

应用次数限制后，可接受多少次身份认证。

秒数默认为 300 （5 分钟）。

参见：

Rate limiting, *RATELIMIT_ATTEMPTS*, *RATELIMIT_LOCKOUT*

2.16.83 RATELIMIT_LOCKOUT

3.2 新版功能.

在应用次数限制后，身份认证锁定多久。

秒数默认为 600 （10 分钟）。

参见：

Rate limiting, *RATELIMIT_ATTEMPTS*, *RATELIMIT_WINDOW*

2.16.84 REGISTRATION_ALLOW_BACKENDS

4.1 新版功能.

在*REGISTRATION_OPEN* 另外禁止的情况下，允许注册的身份验证后端列表。

例：

```
REGISTRATION_ALLOW_BACKENDS = ["azuread-oauth2", "azuread-tenant-oauth2"]
```

提示： 与身份验证 URL 中使用的名称匹配的后端名称。

参见：

REGISTRATION_OPEN

2.16.85 REGISTRATION_CAPTCHA

`True` 或 `False` 的值指示新账户注册是否被 CAPTCHA 保护。这个设置是可选的，默认的 `True` 是假定不提供。

如果打开，CAPTCHA 会添加到用户输入其电子邮箱地址的所有页面中：

- 新账户注册。
- 找回密码。
- 将电子邮箱地址添加到账户中。
- 供未登录用户使用的联系表格。

2.16.86 REGISTRATION_EMAIL_MATCH

2.17 新版功能.

允许您筛选哪个电子邮箱地址可以注册。

默认为 `.*`，允许使用任何电子邮箱地址注册。

您可以用它限制注册到单一的电子邮箱域名：

```
REGISTRATION_EMAIL_MATCH = r'^.*@weblate\.org$'
```

2.16.87 REGISTRATION_OPEN

是否注册新账户在当前是允许的。这个可选设置可以保持默认为 `True`，或更改为 `False`。

这个设置影响了内建的通过电子邮箱地址或通过 Python Social Auth 的身份验证（您可以使用*REGISTRATION_ALLOW_BACKENDS* 为适当的后端建立白名单）。

注解： 如果使用第三方身份验证方法，如*LDAP 身份验证*，只是隐藏注册表格，而新用户仍然能够登录并建立账户。

参见：

REGISTRATION_ALLOW_BACKENDS, *REGISTRATION_EMAIL_MATCH*

2.16.88 REPOSITORY_ALERT_THRESHOLD

4.0.2 新版功能.

触发仓库过期警告的阈值，或者包含了太多更改的阈值。默认为 25。

参见：

Translation component alerts

2.16.89 SENTRY_DSN

3.9 新版功能.

Sentry DSN 用于收集错误报告。

参见：

Django integration for Sentry

2.16.90 SIMPLIFY_LANGUAGES

对于默认语言/国家组合使用简单语言编码。例如，`fr_FR` 翻译将使用 `fr` 语言编码。这通常是受欢迎的特性，因为他简化了这些默认组合列出的语言。

如果对每种不同的变化想要不同的翻译，那么请将其关闭。

2.16.91 SITE_DOMAIN

配置网站域名。这在很多领域产生正确的绝对链接是必要的（例如激活电子邮箱、通知或 RSS 推送）。

在 Weblate 运行在非标准端口时，这里同样要包括它。

示例：

```
# Production site with domain name
SITE_DOMAIN = "weblate.example.com"

# Local development with IP address and port
SITE_DOMAIN = "127.0.0.1:8000"
```

注解： 这个设置应该只包含域名。对于配置协议，（允许或强制 HTTPS ）使用*ENABLE_HTTPS* and for changing URL, use *URL_PREFIX* 。

提示： 关于 Docker 容器，网站域名通过 *WEBLATE_ALLOWED_HOSTS* 来配置。

参见：

Set correct site domain, *Allowed hosts setup*, *Correctly configure HTTPS* *WEBLATE_SITE_DOMAIN*, *ENABLE_HTTPS*

2.16.92 SITE_TITLE

用于网站和发送电子邮件的网站名称。

2.16.93 SPECIAL_CHARS

屏幕键盘中包括的另外的字符，*Visual keyboard* 。

默认值为：

```
SPECIAL_CHARS = ('\t', '\n', '…')
```

2.16.94 SINGLE_PROJECT

3.8 新版功能.

将用户直接重定向到项目或组件，而不是显示控制面板。您可以将其设置为 `True` ，在这种情况下，只在 Weblate 实际只有单一的项目时有用。另外可以设置项目标识串，将无条件地重定向到这个项目。

在 3.11 版更改: 设置只接受项目标识串，来强制显示那个单一项目。

例：

```
SINGLE_PROJECT = "test"
```

2.16.95 STATUS_URL

Weblate 事例报告其状态的 URL 。

2.16.96 SUGGESTION_CLEANUP_DAYS

3.2.1 新版功能.

给定天数后自动删除建议。默认为 `None` ，意味着不删除。

2.16.97 URL_PREFIX

这个设置允许在一些路径下运行 Weblate （否则它依赖于从 web 服务器根目录运行）。

注解: 为了使用这个设置，还需要配置服务器来去掉这个前缀。例如 WSGI ，可以通过设置 `WSGIScriptAlias` 来实现。

提示: 前缀应该以 / 开始。

例:

```
URL_PREFIX = '/translations'
```

注解: 这个设置在 Django’s 内建服务器中不起作用，必须调整 `urls.py` 来包含这个前缀。

2.16.98 VCS_BACKENDS

可用的版本控制系统（VCS ）后端的配置。

注解: Weblate 尝试使用您所有工具支持的后端。

提示: 可以使用这个来限制选择或添加定制版本控制系统（VCS ）后端。

```
VCS_BACKENDS = (
    'weblate.vcs.git.GitRepository',
)
```

参见:

版本控制集成

2.16.99 VCS_CLONE_DEPTH

3.10.2 新版功能.

配置 Weblate 应该对仓库进行深度为多少的克隆。

注解： 当前这只在*Git* 中支持。默认 Weblate 进行仓库的浅克隆，使克隆更快并节省磁盘空间。根据应用（例如当使用定制的附加组件 时），您可能想要增加深度，或通过设置为 0 来完全关闭浅克隆。

提示： 在从 Weblate 推送而得到 `fatal: protocol error: expected old/new/ref, got 'shallow <commit hash>'` 错误的情况下，通过设置来完全关闭浅克隆：

```
VCS_CLONE_DEPTH = 0
```

2.16.100 WEBLATE_ADDONS

可供使用的插件列表。为了使用，必须对给定的翻译部件允许它们。默认包括了所有内建组件，当扩展列表时可能想要允许现有的那些，例如：

```
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",

    # Addon you want to include
    "weblate.addons.example.ExampleAddon",
)
```

参见：

附加组件

2.16.101 WEBLATE_EXPORTERS

4.2 新版功能.

可用的导出程序列表，提供下载各种文件格式的翻译或词汇表。

参见:

支持的文件格式

2.16.102 WEBLATE_FORMATS

3.0 新版功能.

可供使用的文件格式列表。

注解: 默认列表已经具有了常见格式。

参见:

支持的文件格式

2.16.103 WEBLATE_GPG_IDENTITY

3.1 新版功能.

Weblate 使用的身份，用于登录 Git Commits ，例如：

```
WEBLATE_GPG_IDENTITY = 'Weblate <weblate@example.com>'
```

搜索 Weblate GPG 钥匙链（`home/.gnupg`，在*DATA_DIR* 之下）。如果没有找到，则产生密钥，更多细节请查询*Signing Git commits with GnuPG* 。

参见:

Signing Git commits with GnuPG

2.17 Sample configuration

The following example is shipped as `weblate/settings_example.py` with Weblate:

```
#
# Copyright © 2012 - 2020 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/>.
```

(下页继续)

(续上页)

```
#


import os
import platform
from logging.handlers import SysLogHandler

#
# 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",
        # Database password.
        "PASSWORD": "",
        # 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,
        },
    }
}

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
```

(下页继续)

(续上页)

```
LANGUAGE_CODE = "en-us"

LANGUAGES = (
    ("ar", "ةيبرعلا"),
    ("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", "日本語"),
    ("kab", "Taqbaylit"),
    ("kk", "Қазақ тілі"),
    ("ko", "🅵🅵🅵"),
    ("nb", "Norsk bokmål"),
    ("nl", "Nederlands"),
    ("pl", "Polski"),
    ("pt", "Português"),
    ("pt-br", "Português brasileiro"),
    ("ru", "Русский"),
    ("sk", "Slovenčina"),
    ("sl", "Slovenščina"),
    ("sq", "Shqip"),
    ("sr", "Српски"),
    ("sv", "Svenska"),
    ("tr", "Türkçe"),
    ("uk", "Українська"),
    ("zh-hans", "简体字"),
    ("zh-hant", "正體字"),
)

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

# URL prefix to use, please see documentation for more details
```

(下页继续)

(续上页)

```
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.
    # 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,
        },
    }
]
```

(下页继续)

(续上页)

```
# GitHub username for sending pull requests.
# Please see the documentation for more details.
GITHUB_USERNAME = None

# GitLab username for sending merge requests.
# Please see the documentation for more details.
GITLAB_USERNAME = 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",
    "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_FACEBOOK_API_VERSION = "3.1"

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",
```

(下页继续)

(续上页)

```
    "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
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 = "{0}/accounts/email-sent/".format(URL_PREFIX)
SOCIAL_AUTH_LOGIN_ERROR_URL = "{0}/accounts/login/".format(URL_PREFIX)
SOCIAL_AUTH_EMAIL_FORM_URL = "{0}/accounts/email/".format(URL_PREFIX)
SOCIAL_AUTH_NEW_ASSOCIATION_REDIRECT_URL = "{0}/accounts/profile/#account".format(
    URL_PREFIX
)
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")
    #     }
    # },
]

# Allow new user registrations
```

(下页继续)

(续上页)

```
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.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "weblate.accounts.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
    "social_django.middleware.SocialAuthExceptionMiddleware",
    "weblate.accounts.middleware.RequireLoginMiddleware",
    "weblate.api.middleware.ThrottlingMiddleware",
    "weblate.middleware.SecurityMiddleware",
]

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.langdata",
    "weblate.memory",
    "weblate.screenshots",
    "weblate.fonts",
    "weblate.accounts",
    "weblate.utils",
    "weblate.vcs",
    "weblate.wladmin",
    "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",
```

(下页继续)

(续上页)

```
    "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
# - 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 IOError:
        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": "%(levelname)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",
```

(下页继续)

(续上页)

```
        },
        "django.server": {
            "level": "INFO",
            "class": "logging.StreamHandler",
            "formatter": "django.server",
        },
        "syslog": {
            "level": "DEBUG",
            "class": "logging.handlers.SysLogHandler",
            "formatter": "syslog",
            "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},
    },
}

# 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",
```

(下页继续)

(续上页)

```
    #     "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",
    #     "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

# 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
```

(下页继续)

(续上页)

```
MT_TMSERVER = None

# SAP Translation Hub
MT_SAP_BASE_URL = None
MT_SAP_SANDBOX_APIKEY = None
MT_SAP_USERNAME = None
MT_SAP_PASSWORD = None
MT_SAP_USE_MT = True

# Title of site to use
SITE_TITLE = "Weblate"

# Site domain
SITE_DOMAIN = ""

# Whether site uses https
ENABLE_HTTPS = False

# 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 = 1209600
# 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 * 10

# 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
```

(下页继续)

(续上页)

```
SECURE_PROXY_SSL_HEADER = None

# URL of login
LOGIN_URL = "{0}/accounts/login/".format(URL_PREFIX)

# URL of logout
LOGOUT_URL = "{0}/accounts/logout/".format(URL_PREFIX)

# Default location for login
LOGIN_REDIRECT_URL = "{0}/".format(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 = "[{0}] ".format(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.PuctuationSpacingCheck",
#     "weblate.checks.format.PythonFormatCheck",
#     "weblate.checks.format.PythonBraceFormatCheck",
#     "weblate.checks.format.PHPFormatCheck",
#     "weblate.checks.format.CFormatCheck",
#     "weblate.checks.format.PerlFormatCheck",
```

(下页继续)

(续上页)

```
#     "weblate.checks.format.JavaScriptFormatCheck",
#     "weblate.checks.format.CSharpFormatCheck",
#     "weblate.checks.format.JavaFormatCheck",
#     "weblate.checks.format.JavaMessageFormatCheck",
#     "weblate.checks.format.PercentPlaceholdersCheck",
#     "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",
# )

# 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.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",
```

(下页继续)

(续上页)

```
#     "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.yaml.YAMLCustomizeAddon",
#     "weblate.addons.cdn.CDNJSAddon",
#     "weblate.addons.autotranslate.AutoTranslateAddon",
# )

# 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 = [SITE_DOMAIN]

# 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",
            "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": (
```

(下页继续)

(续上页)

```
        "rest_framework.authentication.TokenAuthentication",
        "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
```

(下页继续)

(续上页)

```
# 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"},
}

# 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
AKISMET_API_KEY = None
```

2.18 Management commands

注解: 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
```

参见:

使用 *Docker 安装*, *Installing on Debian and Ubuntu*, *Installing on SUSE and openSUSE*, *Installing on RedHat, Fedora and CentOS*

- *Installing from sources*，建议用于开发。

2.18.2 add_suggestions

weblate add_suggestions <project> <component> <language> <file>

2.5 新版功能.

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).

例:

```
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>

2.5 新版功能.

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 不一致的).

--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.

例:

```
weblate auto_translate --user nijel --inconsistent --source weblate/application␣
↪weblate website cs
```

参见:

自动化翻译

2.18.4 celery_queues

weblate celery_queues

3.7 新版功能.

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.

注解： 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*.

参见：

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.

参见：

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).

在 2.9 版更改: Added parameters `--username`, `--email`, `--name` and `--update`.

2.18.10 dump_memory

`weblate dump_memory`

2.20 新版功能.

Export a JSON file containing Weblate Translation Memory content.

参见：

翻译记忆库, *Weblate Translation Memory Schema*

2.18.11 dumpuserdata

weblate dumpuserdata <file.json>

Dumps userdata to a file for later use by *importuserdata*

提示： This comes in handy when migrating or merging Weblate instances.

2.18.12 import_demo

weblate import_demo

4.1 新版功能.

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>

2.7 新版功能.

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.

在 2.9 版更改: 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"
  }
]
```

参见:

import_memory

2.18.14 import_memory

weblate import_memory <file>

2.20 新版功能.

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.

参见:

翻译记忆库, *Weblate Translation Memory Schema*

2.18.15 import_project

weblate import_project <project> <gitrepo> <branch> <filemask>

在 3.0 版更改: The import_project command is now based on the 组件发现 addon, 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 支持的文件格式), 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'
```

参见:

More detailed examples can be found in the *Starting with internationalization* 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

3.2 新版功能.

weblate install_addon --addon ADDON <project|project/component>

Installs an addon to a set of components.

--addon ADDON
Name of the addon to install. For example `weblate.gettext.customize`.

--configuration CONFIG
JSON encoded configuration of an addon.

--update
Update the existing addon configuration.

You can either define which project or component to install the addon in (for example `weblate/application`), or use `--all` to include all existing components.

To install *自定义 gettext 输出* for all components:

```
weblate install_addon --addon weblate.gettext.customize --config '{"width": -1}' --
↪update --all
```

参见:

附加组件

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/Jazyky>.

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.

注解: 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.

提示： 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.

参见：

unlock_translation

2.18.24 move_language

weblate move_language source target

3.0 新版功能.

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.

例：

```
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.

注解： Weblate pushes changes automatically if *Push on commit* 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.

提示： 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.

参见:

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 根据项目的访问控制.

参见:

访问控制

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.

提示: 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.

注解: Usually it is better to configure hooks in the repository to trigger 通知钩子, instead of regular polling by *updategit*.

2.19 公告

在 4.0 版更改: 在此前的发布版本中，这个特性被称为白板消息。

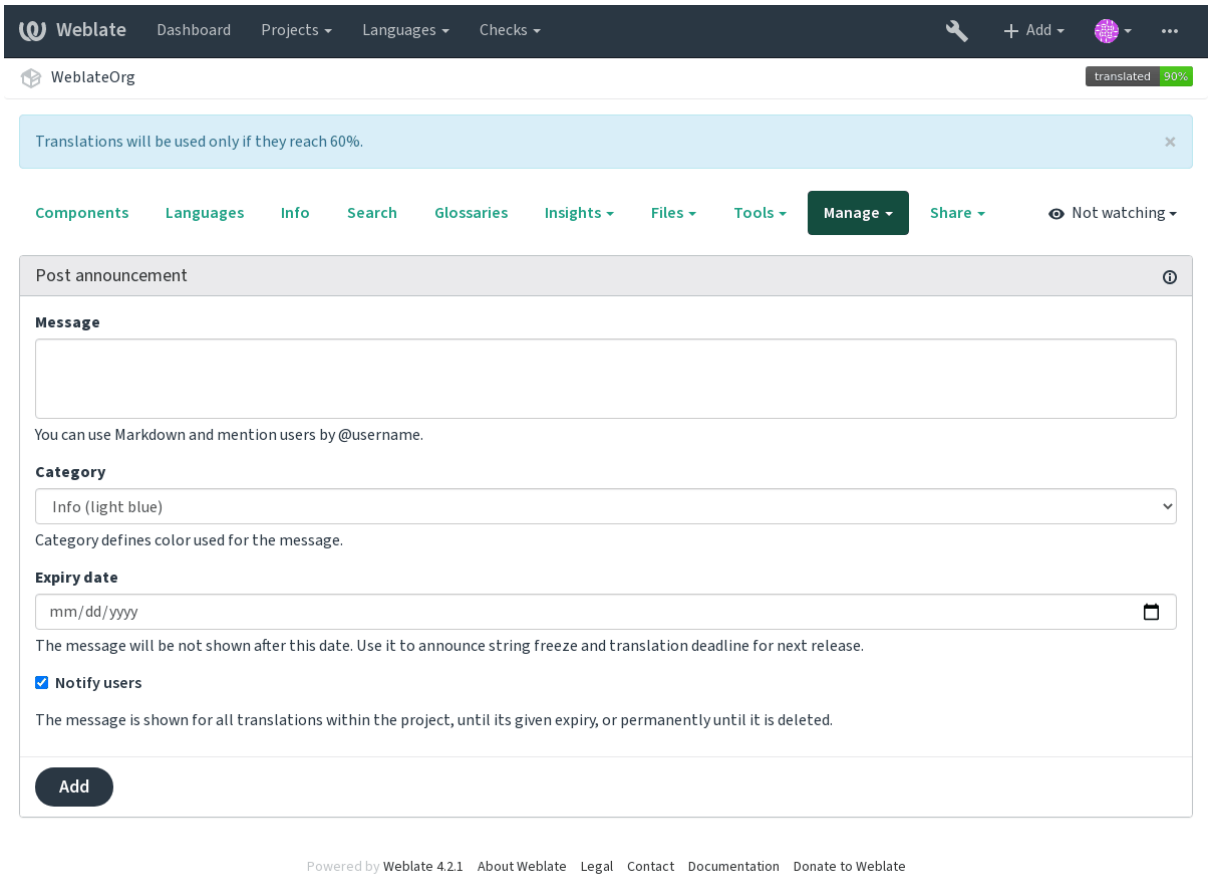
通过张贴网站范围的，根据项目、组件或语言的公告，向翻译员提供信息。

宣布翻译的目的、截止期限、状态或特定目标。

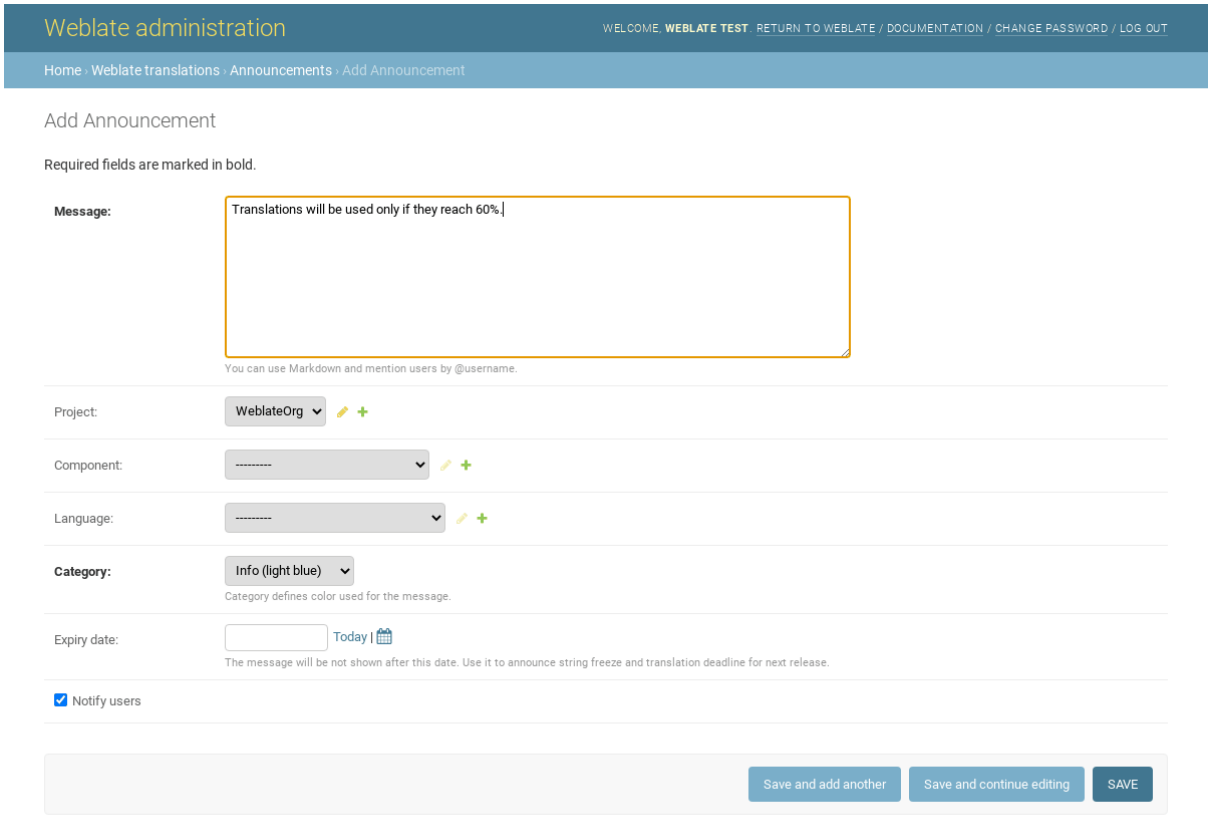
用户可以接收到关注项目公告的通知（除非用户选择关闭）。

这会用于从发布网站的目的到指定翻译的目标的各种事情。

可以使用 *Post announcement* ，在 *Manage* 菜单的每一届张贴公告:



还可以使用管理界面添加:



然后根据特定的上下文显示公告：

没有特定的上下文

显示在控制面版上（着陆页面）。

特定项目

项目内显示，包括其所有的组件和翻译。

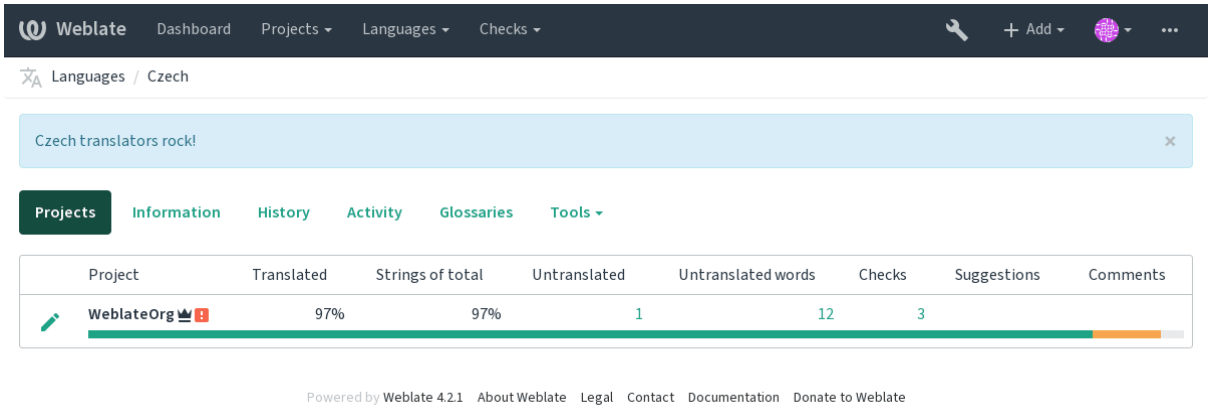
特定组件

对于给定的组件机器翻译来显示。

特定语言

显示语言的全景和该语言的全部翻译。

这就是语言全景页面上的样子：



2.20 组件列表

指定多个列表的组件，出现在用户控制面板上作为选项，从用用户可以选择一个作为默认视图。请见控制面板 来了解更多信息。

在 2.20 版更改: 控制面板上出现的每个组件列表都会显示状态。

可以在管理界面的 *Component lists* 部分指定组件列表的名称和内容。每种组件列表必须名称来显示给用户，并具有标识串将其显示在 URL 中。

在 2.13 版更改: 从管理界面检查匿名用户的控制面板设置，修改掉控制面本显示给未授权用户的内容。

2.20.1 自动组件列表

2.13 新版功能.

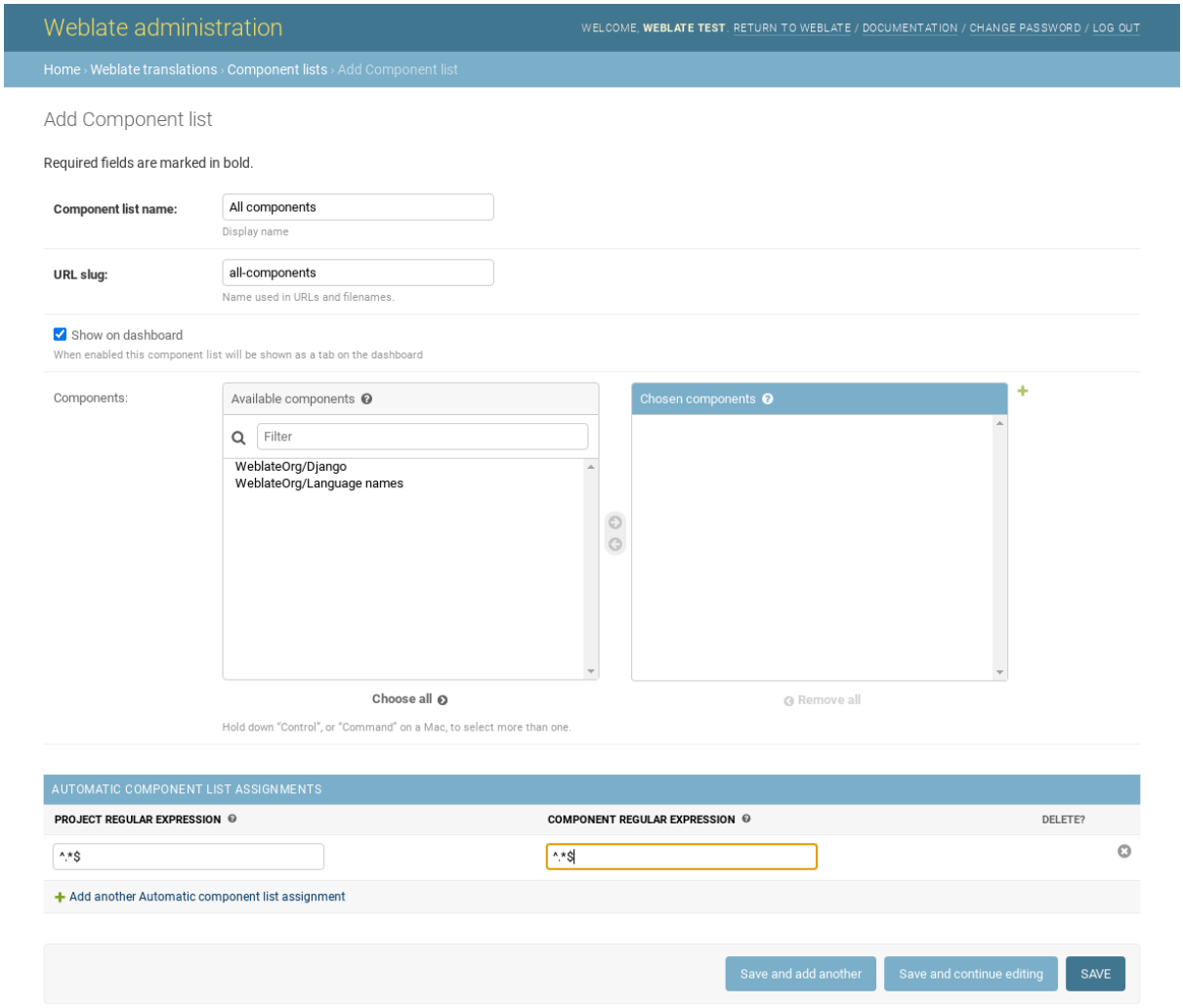
通过建立 *Automatic component list assignment* 规则，根据其标识串自动将组件添加到列表中。

- 对于维护大型安装的逐渐列表有用，或者如果你希望在 Weblate 安装中有一个包含所有组件的组件列表。

提示: 制作组件列表，包含自己的 Weblate 安装时的所有组件。

---

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

2.10 新版功能.

Provides you read-only access to the underlying Git repository using HTTP(S).

安装

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/master/`:

```
git clone 'https://example.org/git/weblate/master/'
```

Repositories are available anonymously unless 根据项目的访问控制 is turned on. This requires authenticate using your API token (it can be obtained in your 用户资料):

```
git clone 'https://user:KEY@example.org/git/weblate/master/'
```

2.21.2 账单

2.4 新版功能.

This is used on Hosted Weblate to define billing plans, track invoices and usage limits.

安装

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 用户资料.

The billing module additionally allows project admins to create new projects and components without being superusers (see 添加翻译项目和组件). 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 法律声明

2.15 新版功能.

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

---

注解: Legal documents for the Hosted Weblate service is available in this Git repository <https://github.com/WeblateOrg/hosted/tree/master/wlhosted/legal/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.

安装

1. Add `weblate.legal` to installed apps in `settings.py`:

```
INSTALLED_APPS += (
    'weblate.legal',
)

# Optional:

# Social auth pipeline to confirm TOS upon registration/subsequent login
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

参见:

头像缓存, *AVATAR_URL_PREFIX*, *ENABLE_AVATARS*

2.21.5 Spam protection

You can protect against suggestion spamming by unauthenticated users by using the akismet.com service.

1. Install the *akismet* Python module
2. Configure the Akismet API key.

注解: This (among other things) relies on IP address of the client, please see *Running behind reverse proxy* for properly configuring that.

参见:

Running behind reverse proxy, *AKISMET_API_KEY*

2.21.6 Signing Git commits with GnuPG

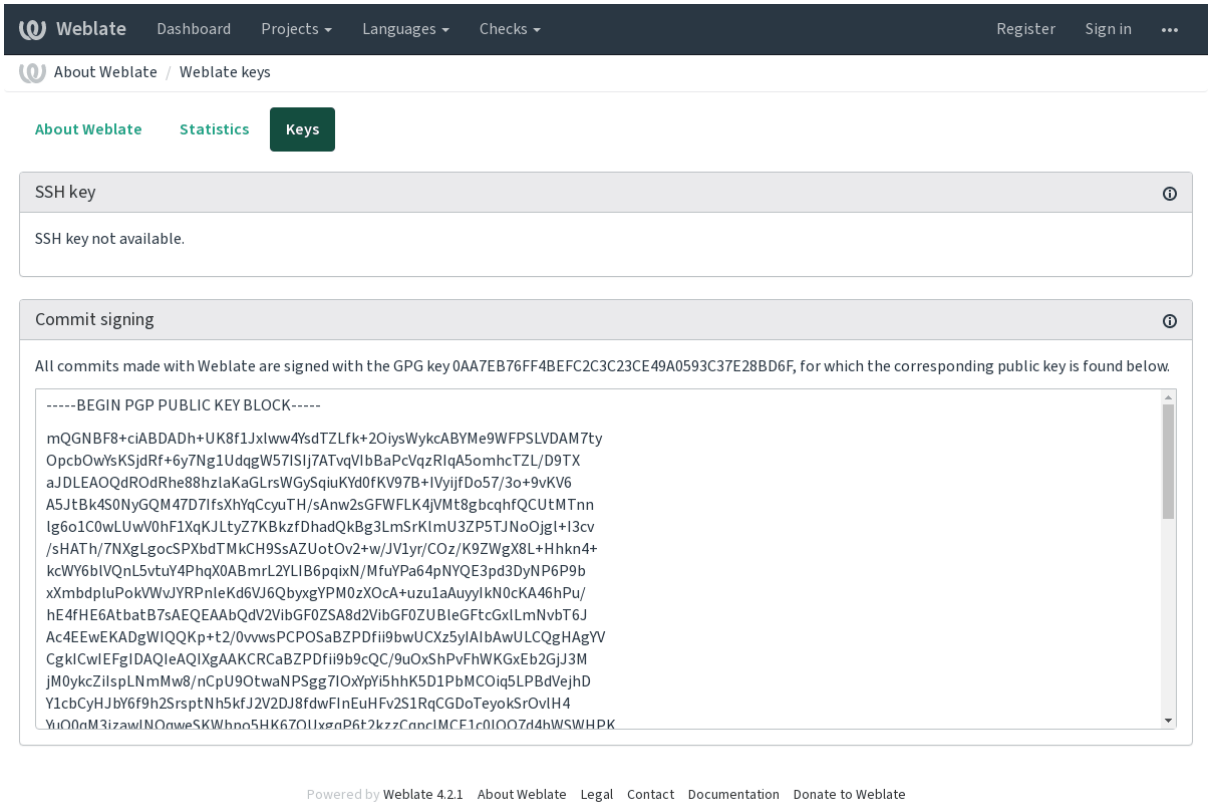
3.1 新版功能.

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.

参见:

WEBLATE_GPG_IDENTITY

2.21.7 Rate limiting

在 3.2 版更改: The rate limiting now accepts more fine-grained configuration.

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:

名称	范围	Allowed attempts	Ratelimit window	Lockout period
注册	`REGISTRATION`	5	300	600
Sending message to admins	`MESSAGE`	5	300	600
Password authentication on login	`LOGIN`	5	300	600
Sitewide search	`SEARCH`	6	60	60
Translating	`TRANSLATE`	30	60	600
Adding to glossary	`GLOSSARY`	30	60	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.

参见:

Rate limiting, *Running behind reverse proxy*

2.22 定制 Weblate

使用 Django 和 Python 扩展与定制。将您的更改贡献给上游使每人都能够受益。这降低了您的维护成本；Weblate 中的代码对更改内部界面或重构编码时的情况。

警告： 内部界面与模板都不被认为是稳定的 API 。请您对每次升级都复查自己的定制，接口或其语义可能未经通知就进行更改。

参见：

Contributing to Weblate

2.22.1 建立 Python 模块

如果您不熟悉 Python ，您可以查看 Python For Beginners ，它解释了其基本内容并指向了高级教程。

为了写出一些定制 Python 代码（被称为模块），需要一个地方进行存储，或者在系统路径（通常像 `/usr/lib/python3.7/site-packages/` 的地方），或者在 Weblate 目录下，同样也添加到翻译搜索路径下。

更好地是，将您的定制化转变为适当的 Python 包：

1. 为您的包建立文件夹（我们会使用 *weblate_customization* ）。
2. 在里面建立 `setup.py` 文件来描述包：

```
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",
    packages=['weblate_customization'],
)
```

3. 建立定制代码的 Python 模块（也被成为 `weblate_customization` ）的文件夹。
4. 在里面建立 `__init__.py` 文件来确认 Python 可以导入模块。
5. 现在可以使用 **`pip install -e`** 安装这个包。更多信息可以在 “Editable” Installs 中找到。
6. 模块一旦安装，就可以用在 Webalte 配置中（例如 `weblate_customization.checks.FooCheck` ）。

您的模块结构应该看起来像这样：

```
weblate_customization
├── setup.py
└── weblate_customization
    ├── __init__.py
    ├── addons.py
    └── checks.py
```

可以在 <https://github.com/WeblateOrg/customize-example> 找到定制 Weblate 的例子，它涵盖了下面描述的所有题目。

2.22.2 更改 Logo

1. Create a simple Django app containing the static files you want to overwrite (see 建立 *Python* 模块).

2. 把它添加到:setting:*django:INSTALLED_APPS*:

```
INSTALLED_APPS = (
    # Add your customization as first
    'weblate_customization',

    # Weblate apps are here…
)
```

品牌出现在后面的文件中：

icons/weblate.svg 导航栏中显示的 Logo 。

logo-*.png 根据屏幕分辨率和 web 浏览器的 Web 图标。

favicon.ico 传统浏览器使用的 Web 图标。

weblate-*.png 机器人或匿名用户使用的头像。一些 Web 浏览器使用这些作为快捷图标。

email-logo.png 在通知电子邮件中使用。

3. 运行 `weblate collectstatic --noinput` ，来收集提供给客户端的静态文件。

参见:

Managing static files (e.g. images, JavaScript, CSS), *Serving static files*

2.22.3 定制的质量检查、插件和自动修复

为了定制的自动修正 、编写自己的检查 或编写附加组件 并在 Weblate 中安装您的代码：

1. 将文件放在您的包含 Weblate 定制的 Python 模块中（请见建立 *Python* 模块 ）。

2. 在专用设置（*WEBLATE_ADDONS* 、*CHECK_LIST* 或*AUTOFIX_LIST* ）中将其完全合法的路径添加到 Python 类中：

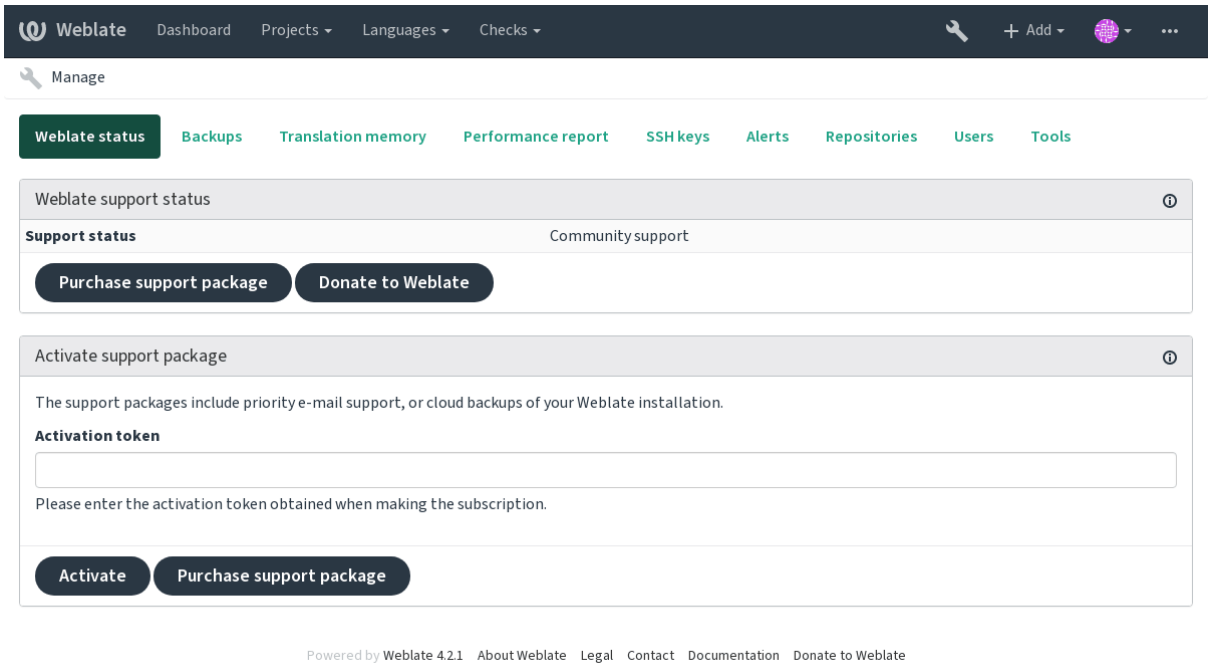
```
# Checks
CHECK_LIST += (
    'weblate_customization.checks.FooCheck',
)

# Autofixes
AUTOFIX_LIST += (
  'weblate_customization.autofix.FooFixer',
)

# Addons
WEBLATE_ADDONS += (
  'weblate_customization.addons.ExamplePreAddon',
)
```

参见:

定制的自动修正:, 编写自己的检查, 编写附加组件, 从附加组件执行脚本

2.23 管理界面

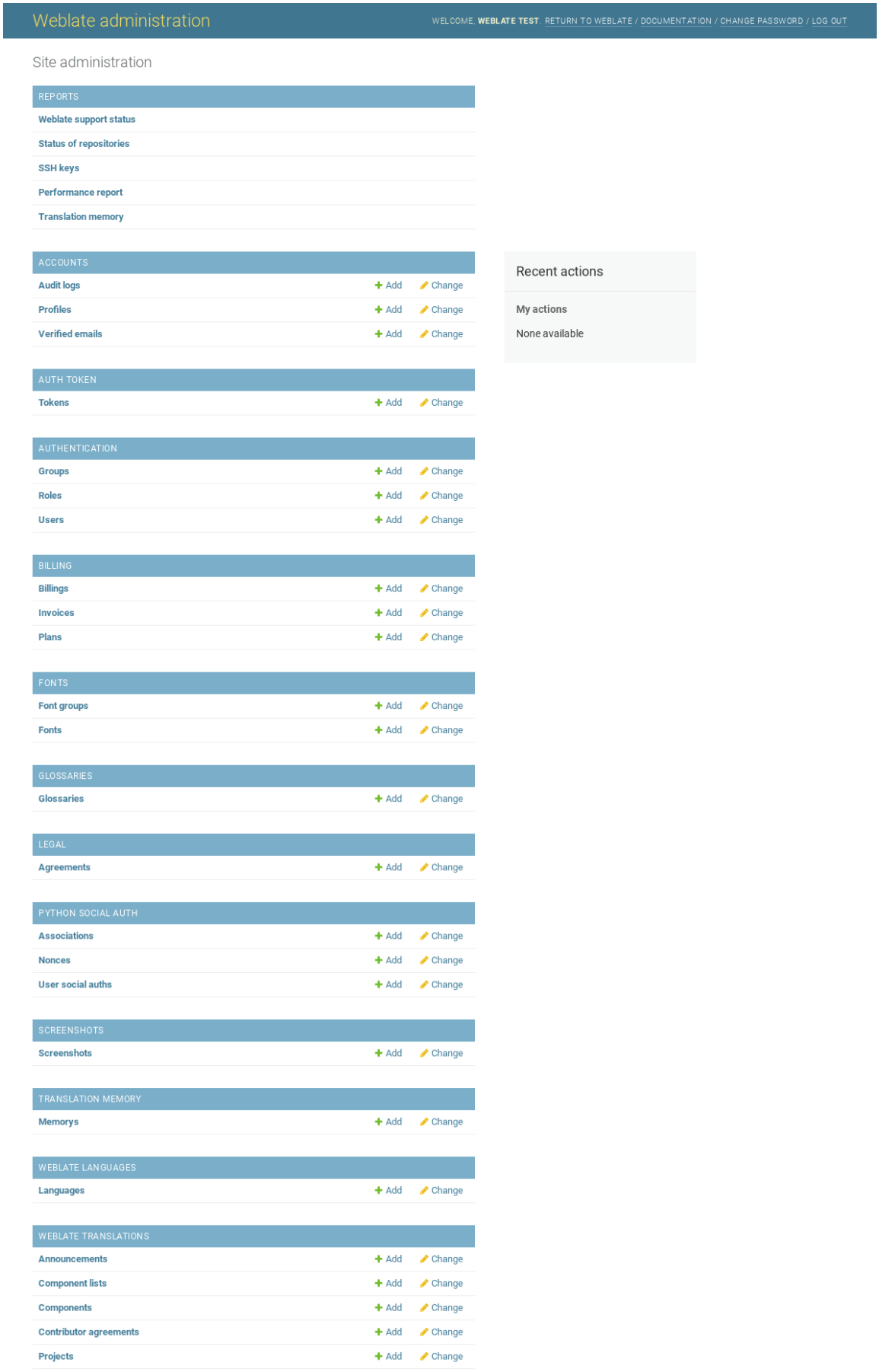
管理界面在 `/management/` URL 下面提供管理设置。它对于具有管理特权的登录用户是可用的，通过使用右上角的扳手图标来访问：



2.23.1 Django 管理界面

> **警告：** 将来会删除，因为其应用困难——多数特性可以直接在 Weblate 中管理。

可以在这里管理数据库中存储的对象，如用户、翻译和其他设置：


Weblate administration
WELCOME, WEBLATE TEST. RETURN TO WEBLATE / DOCUMENTATION / CHANGE PASSWORD / LOG OUT
Site administration
REPORTS
Weblate support status
Status of repositories
SSH keys
Performance report
Translation memory
ACCOUNTS
Audit logs + Add Change
Profiles + Add Change
Verified emails + Add Change
AUTH TOKEN
Tokens + Add Change
AUTHENTICATION
Groups + Add Change
Roles + Add Change
Users + Add Change
BILLING
Billings + Add Change
Invoices + Add Change
Plans + Add Change
FONTS
Font groups + Add Change
Fonts + Add Change
GLOSSARIES
Glossaries + Add Change
LEGAL
Agreements + Add Change
PYTHON SOCIAL AUTH
Associations + Add Change
Nonces + Add Change
User social auths + Add Change
SCREENSHOTS
Screenshots + Add Change
TRANSLATION MEMORY
Memorys + 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

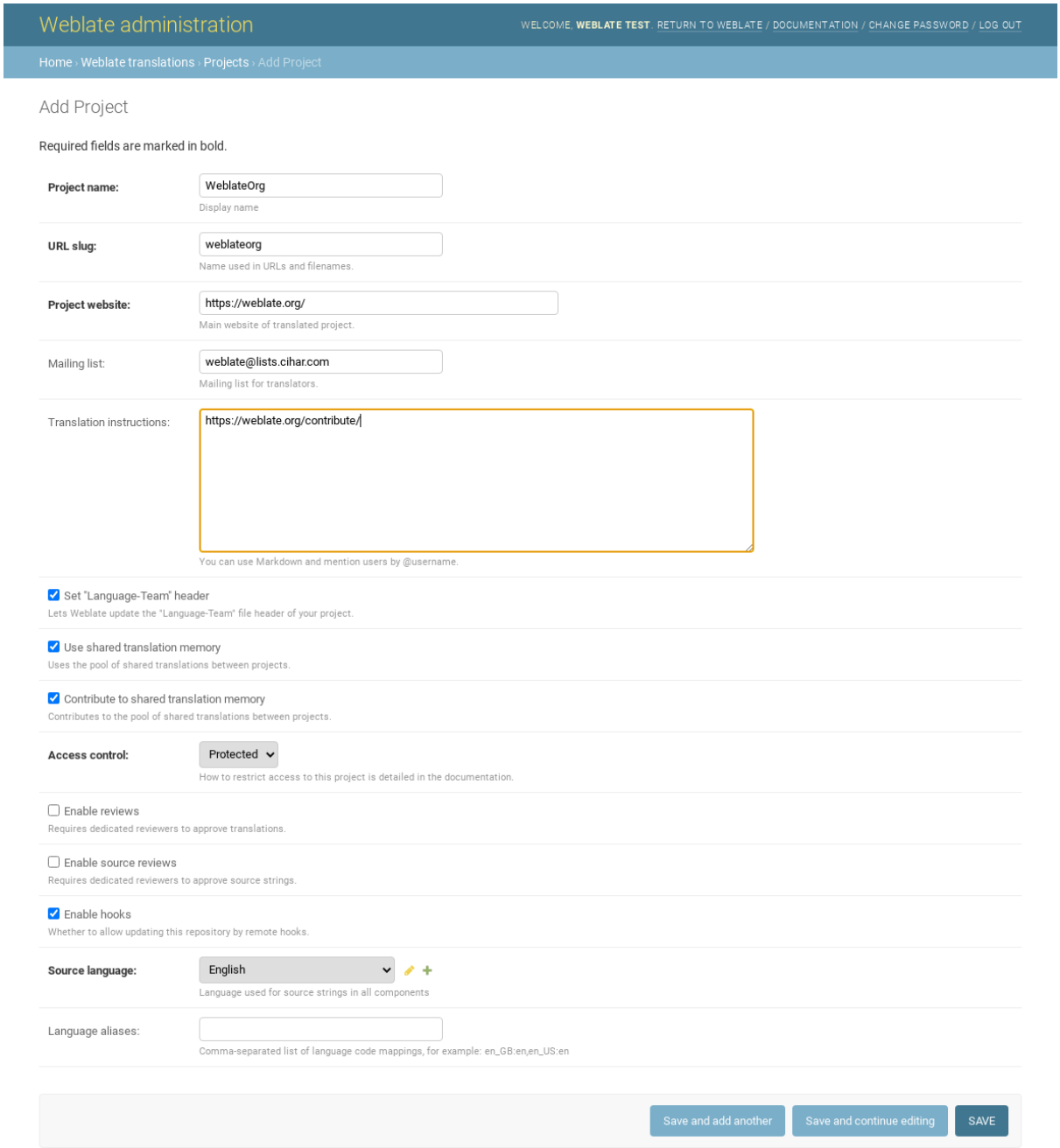
在 *Reports* 部分，可以检查网站的状态，为*Production setup* 进行调整，或者管理用于访问的 SSH 密钥*Accessing repositories* 。

管理任意部分下的数据库对象。最有趣的也许是 *Weblate translations* ， 你可以在这里管理可翻译的项目，请见项目配置 和*Component configuration* 。

Weblate languages 保持语言定义， 在*Language definitions* 中进一步解释。

添加项目

添加项目作为所有组件的容器。通常可以为一部分软件或图书（各自参数的信息请见项目配置 ）来建立一个项目：

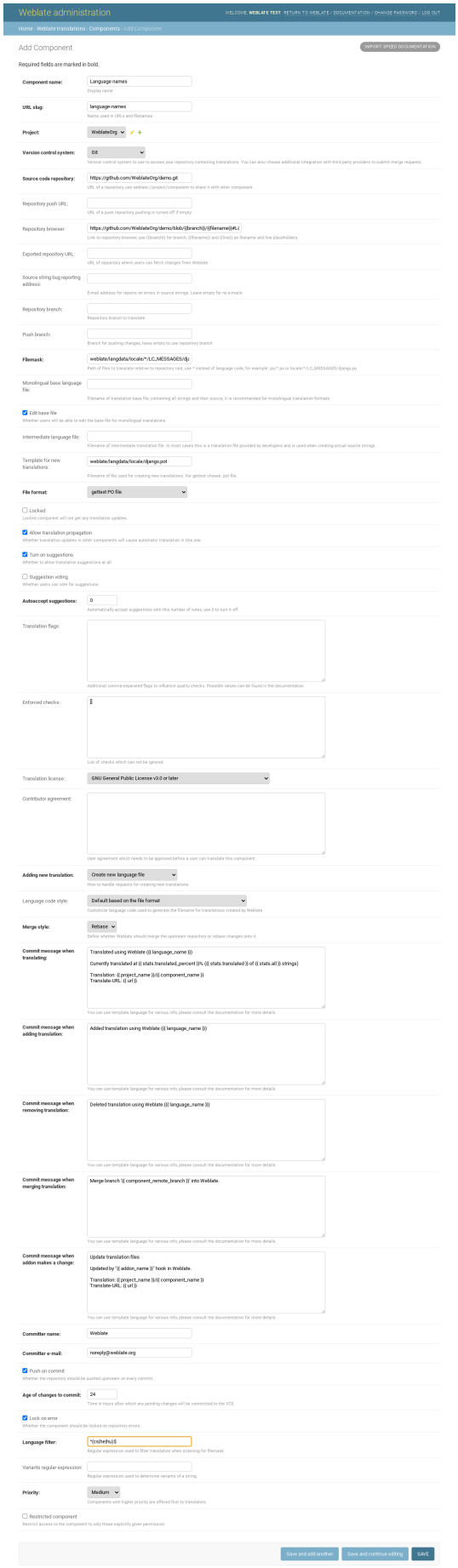


参见：

项目配置

双语言组件

一旦添加了一个项目，就可以添加翻译组件了。(关于各自参数的信息，请见*Component configuration*):

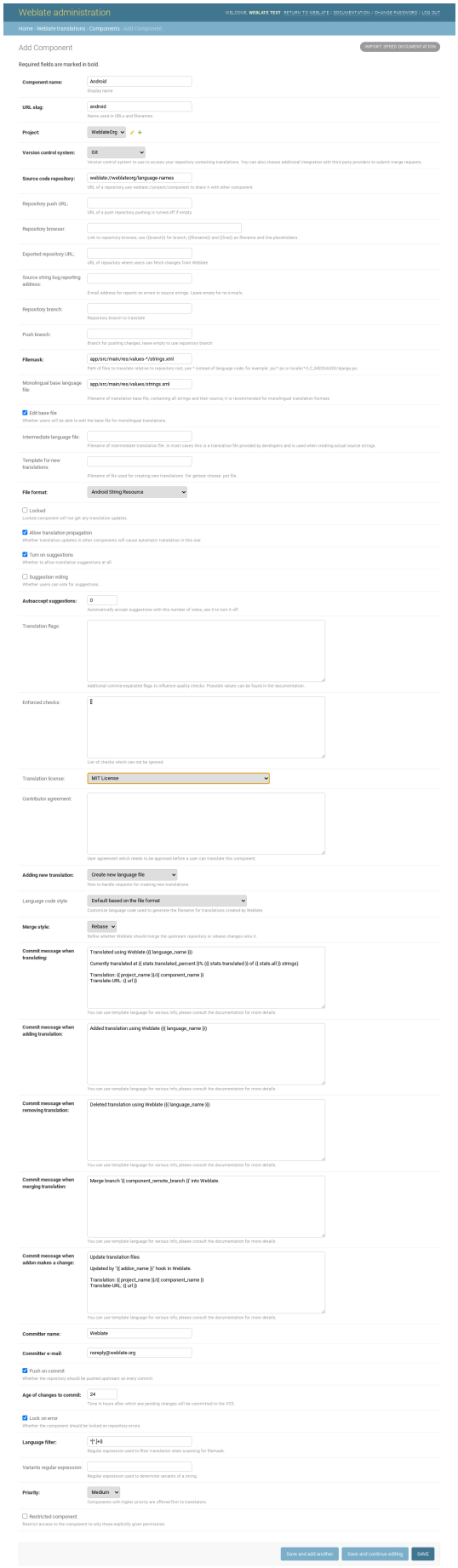

Weblate administration
WELCOME, WEBLATE TEST. RETURN TO WEBLATE / DOCUMENTATION / CHANGE PASSWORD / LOG OUT
Home › Weblate translations › Components › Add Component
Add Component
IMPORT SPEED DOCUMENTATION
Required fields are marked in bold.
Component name:
Language names
Display name
URL slug:
language-names
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.
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/langdata/locale/*/LC_MESSAGES/djs
Path of files to translate relative to repository root, use * instead of language code, for example: po/*.po or locale/*/LC_MESSAGES/djangojs.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.
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
Whether users can vote for suggestions.
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 quality checks. Possible values can be found in the documentation.
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.
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.
Committer name:
Weblate
Committer e-mail:
noreply@weblate.org
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.
Language filter:
^(cs|he|hu)$
Regular expression used to filter translation 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.
Save and add another
Save and continue editing
SAVE

参见:

Component configuration, 双语和单语格式

单语言组件

为了使这些翻译更容易，提供了模板文件，包含了各自源语言的对应信息 ID （通常为英语）。(对于各自参数的信息，请见*Component configuration*):


Weblate administration
WELCOME, WEBLATE TEST. RETURN TO WEBLATE / DOCUMENTATION / CHANGE PASSWORD / LOG OUT
Home › Weblate translations › Components › Add Component
Add Component
IMPORT SPEED 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.
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/django.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
Whether users can vote for suggestions.
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 quality checks. Possible values can be found in the documentation.
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.
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.
Committer name:
Weblate
Committer e-mail:
noreply@weblate.org
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.
Language filter:
^[^i]+$
Regular expression used to filter translation 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.
Save and add another
Save and continue editing
SAVE

参见:

Component configuration, 双语和单语格式

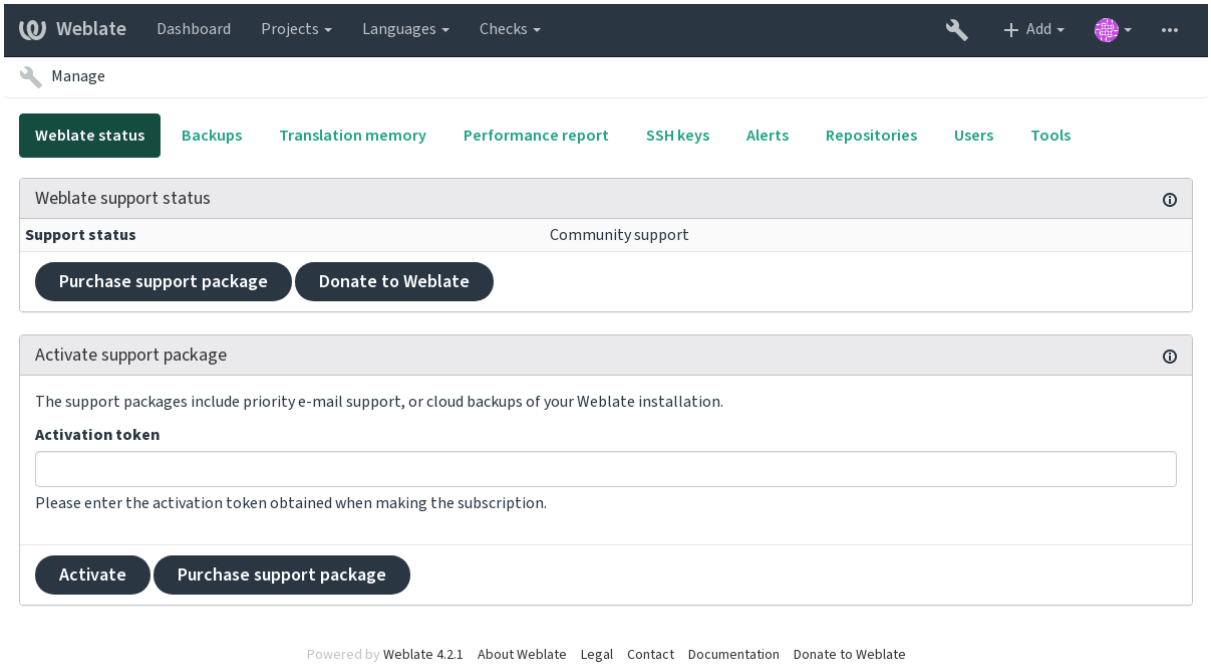
2.24 Getting support for Weblate

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

3.8 新版功能.

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

No other data is submitted.

2.24.3 Integration services

- See if your support package is still valid
- 使用 *Weblate* 配置的备份存储

提示： Purchased support packages are already activated upon purchase, and can be used without integrating them.

2.25 Legal documents

注解： 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 depend 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)

参见:

Export Controls (EAR) on Open Source Software

CHAPTER 3

Contributor docs

3.1 Contributing to Weblate

There are dozens of ways to contribute in Weblate. Any help is welcomed, be it coding, graphics design, documentation or sponsorship:

- *Reporting issues in Weblate*
- *Starting contributing code to Weblate*
- *Translating Weblate*
- *Funding Weblate development*

3.1.1 Translating Weblate

Weblate is being translated using Weblate itself, feel free to take part in the effort of making Weblate available in as many human languages as possible.

3.1.2 Funding Weblate development

You can fund further Weblate development on the donate page. Funds collected there are used to fund gratis hosting for libre software projects, and further development of Weblate. Please check the *donate page* for details, such as funding goals and rewards you can get for being a 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)

Do you want to be in the list? Please see options on the Donate to Weblate.

3.2 Starting contributing code to Weblate

To understand Weblate source code, please first look into *Weblate source code*, *Weblate frontend* and *Weblate internals*.

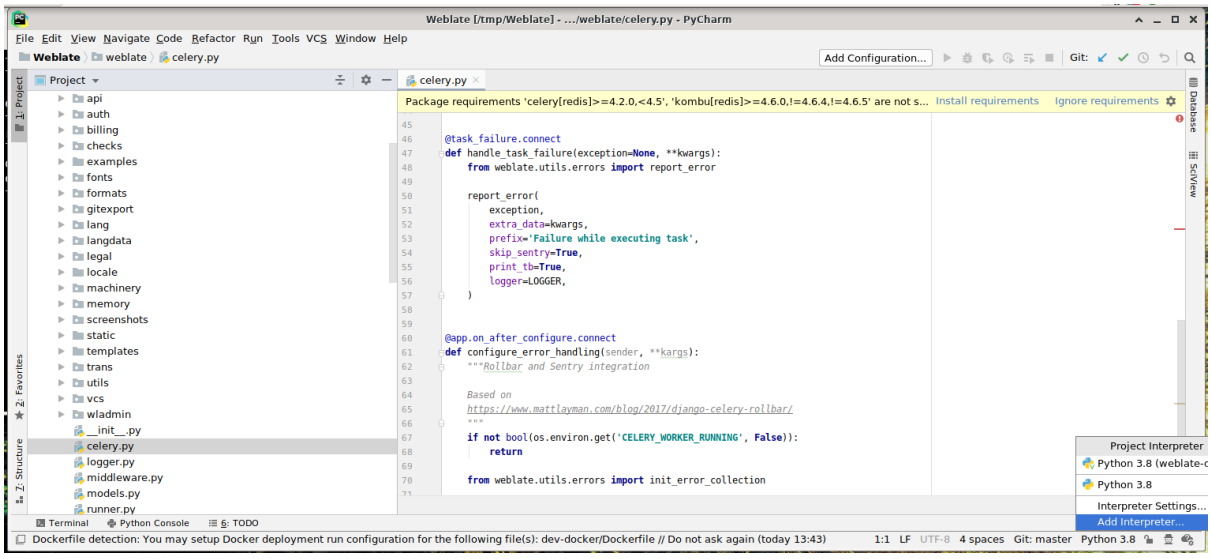
3.2.1 Starting with our codebase

If looking for some bugs to familiarize yourself with the Weblate codebase, look for ones labelled good first issue.

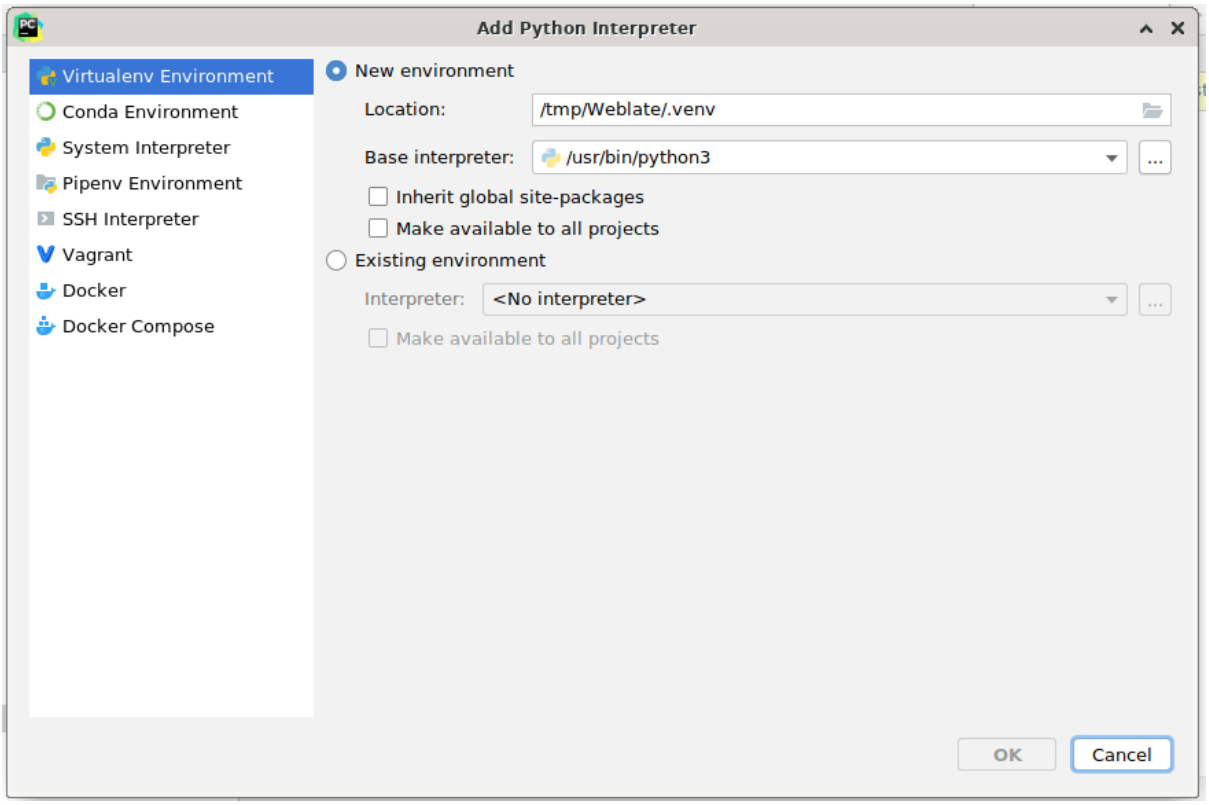
3.2.2 Coding Weblate with PyCharm

PyCharm is a known IDE for Python, here's some guidelines to help you setup Weblate project in it.

Considering you have just cloned the Github repository, just open the folder in which you cloned it in PyCharm. Once the IDE is open, the first step is to specify the interpreter you want:

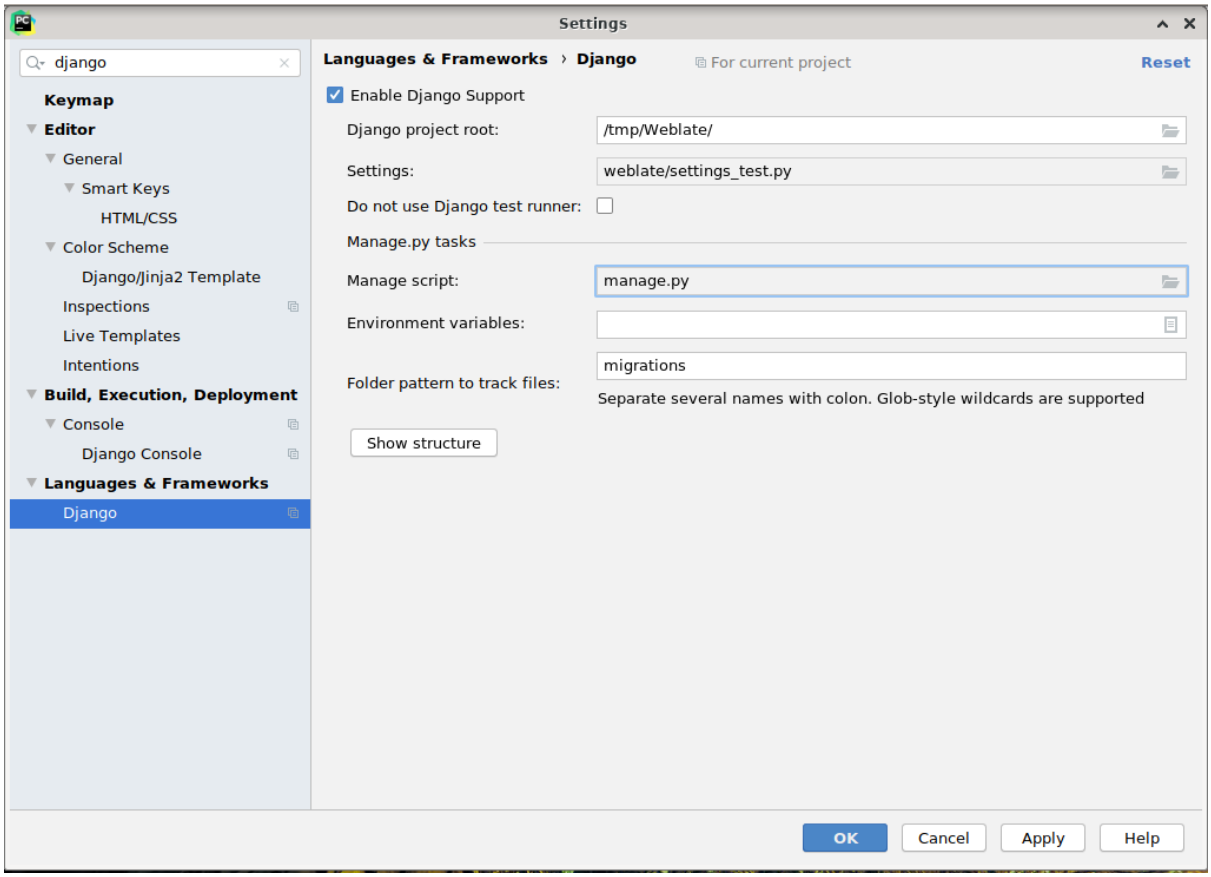


You can either chose to let PyCharm create the virtualenv for you, or select an already existing one:



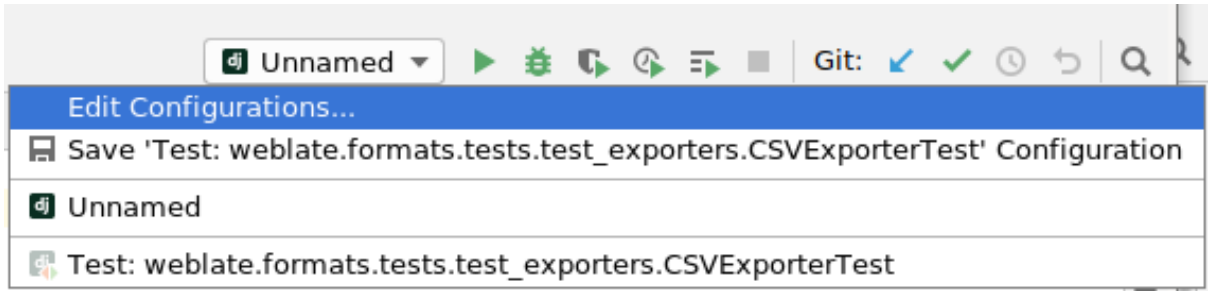
Don’t forget to install the dependencies once the interpreter is set: you can do it, either through the console (the console from the IDE will directly use your virtualenv by default), or through the interface when you get a warning about missing dependencies.

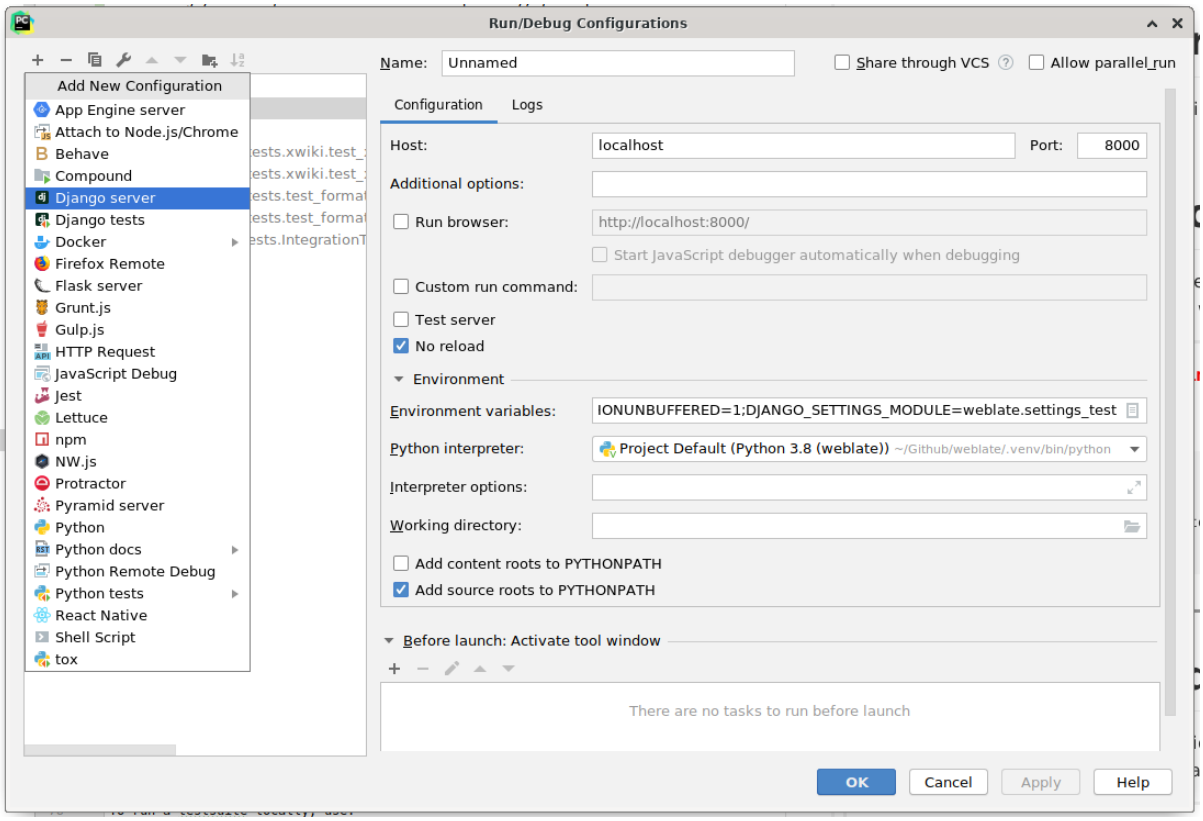
The second step is to set the right information to use natively Django inside PyCharm: the idea is to be able to immediately trigger the unit tests in the IDE. For that you need to specify the root path of Django and the path of one setting:



Be careful, the *Django project root* is the root of the repository, not the weblate sub-directory. About the settings, I personally use the *settings_test* from the repository, but you could create your own setting and set it there.

Last step is to be able to run the server and to put breakpoints on the code to be able to debug it. This is done by creating a new *Django Server* configuration:





Be careful to properly checked “No reload”: you won’t get anymore the server live reload if you modify some files, but the debugger will be stopped on the breakpoint you set.

3.2.3 Running Weblate locally

The most comfortable approach to get started with Weblate development is to follow *Installing from sources*. It will get you a virtual env with editable Weblate sources.

To install all dependencies useful for development, do:

```
pip install -r requirements-dev.txt
```

To start a development server run:

```
weblate runserver
```

Depending on your configuration you might also want to start Celery workers:

```
./weblate/examples/celery start
```

Running Weblate locally in Docker

If you have Docker and docker-compose installed, you can spin up the development environment simply by running:

```
./rundev.sh
```

It will create development Docker image and start it. Weblate is running on <http://127.0.0.1:8080/> and you can sign in with `admin` user and `admin` password. The new installation is empty, so you might want to continue with 添加翻译项目和组件.

The `Dockerfile` and `docker-compose.yml` for this are located in `dev-docker` directory.

The script also accepts some parameters, to execute tests run it with `test` parameter and then specify any `test` parameters, for example:

```
./rundev.sh test --failfast weblate.trans
```

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 stop the background containers run:

```
./rundev.sh stop
```

Running the script without args will recreate Docker container and restart it.

注解： This is not suitable setup for production, it includes several hacks which are insecure, but make development easier.

3.2.4 Bootstraping your devel instance

You might want to use *import_demo* to create demo translations and *createadmin* to create 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.

参见：

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 it's 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 调试模式

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.

参见:

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).

Docker containers log to their output (as usual in the Docker world), so you can look at the logs using `docker-compose logs`.

参见:

Sample configuration contains `LOGGING` configuration.

3.4.3 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 收集错误报告.

3.4.4 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 收集错误报告 helps you to notice such failures easier.

3.4.5 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 <https://pypi.org/project/dogslow/>* along with 收集错误报告 and get pinpointed and detailed tracebacks in the error collection tool.

3.5 Weblate internals

注解： 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, 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`
: Addons to tweak Weblate behavior, see 附加组件.

`api`
: API based on Django REST framework.

`auth`
: Authentication and permissions.

`billing`
: The optional 账单 module.

`checks`
: Translation string 质量检查 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.

`langdata`
: Language data definitions.

`legal`

The optional 法律声明 module.

`machinery`

Integration of machine translation services.

`memory`

Built in translation memory, see 翻译记忆库.

`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 Weblate frontend

The frontend is currently built using Bootstrap, jQuery and few third party libraries.

3.6.1 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.

3.6.2 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.

3.6.3 本地化

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'
```

参见：

Translation topic in the Django documentation

3.6.4 Icons

Weblate currently uses material design icons, in case you are looking for new one, check <https://materialdesignicons.com/>.

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.

3.7 Reporting issues in Weblate

Our issue tracker is hosted at GitHub:

Feel welcome to report any issues with, or suggest improvement of Weblate there. If what you have found is a security issue in Weblate, please consult the "Security issues" section below.

3.7.1 安全问题

In order to give the community time to respond and upgrade your are strongly urged to report all security issues privately. HackerOne is used to handle security issues, and can be reported directly at HackerOne.

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*.

注解： 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

3.8 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.

3.8.1 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=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
./ci/run-setup
```

3.8.2 Local testing

To run a testsuite locally, use:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py test
```

提示： 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=postgresql

# Database server configuration
```

(下页继续)

(续上页)

```
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
```

提示： The tests can also be executed inside developer docker container, see *Running Weblate locally in Docker*.

参见：

See Testing in Django for more info on running and writing tests for Django.

3.9 Data schemas

Weblate uses JSON Schema to define layout of external JSON files.

3.9.1 Weblate Translation Memory Schema

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
		pattern	^(.*)$
		default	
	• **source**	*The Source String*	
		type	*string*
		examples	Hello
		pattern	^(.+)$
		default	
	• **source_language**	*The Source Language*	
		ISO 639-1 / ISO 639-2 / IETF BCP 47	
		type	*string*

下页继续

表 1 – 续上页

		examples	en
		pattern	^([^]+)$
		default	
	• **target**	*The Target String*	
		type	*string*
		examples	Ahoj
		pattern	^(.+)$
		default	
	• **target_language**	*The Target Language*	
		ISO 639-1 / ISO 639-2 / IETF BCP 47	
		type	*string*
		examples	cs
		pattern	^([^]+)$
		default	
	additionalProperties	False	
definitions			

参见:

翻译记忆库, *dump_memory*, *import_memory*

3.9.2 Weblate user data export

type	*object*		
properties			
• **basic**	*Basic*		
	type	*object*	
	properties		
	• **username**	*Username*	
		type	*string*
		examples	admin
		pattern	^.*$
		default	
	• **full_name**	*Full name*	
		type	*string*
		examples	Weblate Admin
		pattern	^.*$
		default	
	• **email**	*E-mail*	
		type	*string*
		examples	noreply@example.com
		pattern	^.*$
		default	
	• **date_joined**	*Date joined*	
		type	*string*
		examples	2019-11-18T18:53:54.862Z
		pattern	^.*$
		default	
• **profile**	*Profile*		
	type	*object*	
	properties		
	• **language**	*Language*	
		type	*string*
		examples	cs
		pattern	^.*$

下页继续

表 2 – 续上页

		default			
	• **suggested**	*Number of sugested 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		
	• **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	None		
		anyOf	•	type	*null*
			•	type	*integer*
	• **languages**	*Translated languages*			

下页继续

表 2 – 续上页

		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*		
	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				

参见:

用户资料, *dumpuserdata*

3.10 Releasing Weblate

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**

Perform the release:

4. Create a release **./scripts/create-release --tag** (see below for requirements)

Post release manual steps:

5. Update Docker image.
6. Close GitHub milestone.
7. Once the Docker image is tested, add a tag and push it.
8. Update Helm chart to new version.
9. Include new version in `.github/workflows/migrations.yml` to cover it in migration testing.
10. 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)

3.11 关于 Weblate

3.11.1 Project goals

Web-based continuous localization tool with tight 版本控制集成 supporting a wide range of 支持的文件格式, making it easy for translators to contribute.

3.11.2 项目名称

“Weblate” is a portmanteau of the words “web” and “translate”.

3.11.3 项目网站

The landing page is <https://weblate.org/> and a cloud hosted service at <https://hosted.weblate.org/>. This documentation can be found on <https://docs.weblate.org/>.

3.11.4 Project logos

The project logos and other graphics is available in <https://github.com/WeblateOrg/graphics/> repository.

3.11.5 Leadership

This project is maintained by Michal Čihař <michal@cihar.com>.

3.11.6 Authors

Weblate was started by Michal Čihař <michal@cihar.com>. Since its inception in 2012, thousands of people have contributed.

3.12 许可协议

CHAPTER 4

Change History

4.1 Weblate 密钥

Released on August 21st 2020.

- Fixed saving plurals for some locales in Android resources.
- Fixed crash in the cleanup addon for some XLIFF files.
- Allow to configure localization CDN in Docker image.

4.2 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.
- 添加了几种单语言格式的导出
- Improved activity charts.
- Number of displayed nearby strings can be configured.
- 增加了对锁定遇到存储库错误的组件的支持。
- Simplified main navigation (replaced buttons with icons).
- Improved language code handling in Google Translate integration.
- The Git squash addon 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.
- 添加了对 ECMAScript 模板文本的检查。
- 添加了监视组件的选项。
- Removed leading dot from JSON unit keys.
- 删除了翻译记忆库单独的 Celery 队列。
- 允许使用同一种语言同时翻译所有组件
- Allow to configure `Content-Security-Policy` HTTP headers.
- Added support for aliasing languages at project level.
- New addon to help with HTML or JavaScript localization, see *JavaScript 本地化 CDN*.
- The Weblate domain is now configured in the settings, see *SITE_DOMAIN*.
- 增加对按组件和项目进行搜索的支持

4.3 Weblate 4.1.1

Released on June 19th 2020.

- Fixed changing autofix or addons 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 *SAML 身份验证*.

4.4 Weblate 4.1

Released on June 15th 2020.

- Added support for creating new translations with included country code.
- 增加了对用截图搜索源字符串的支持。
- 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 字符串*, *HTML files*, *OpenDocument Format*, *IDML Format*, *Windows RC files*, *INI translations*, *Inno Setup INI 翻译*, *GWT properties*, *go-i18n JSON files*, *ARB File*.
- Consistently use dismissed as state of dismissed checks.
- Add support for configuring default addons 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: 多个未命名的变量, 长期未翻译, 连续重复的单词.
- 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 定制行为.
- Always show rendered text check if enabled.
- API now supports filtering of changes.
- Added support for sharing glossaries between projects.

4.5 Weblate 4.0.4

Released on May 07th 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.

4.6 Weblate 4.0.3

Released on May 02nd 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.

4.7 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.

4.8 Weblate 4.0.1

Released on April 16th 2020.

- Fixed package installation from PyPI.

4.9 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.
- 显示的建议未能通过检查。
- 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 未更改的翻译 avoid internal blacklist.
- Improved comments extraction from monolingual po files.
- Renamed whiteboard messages to announcements.
- Fixed occasional problems with registration mails.
- Improved LINGUAS update addon 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.

4.10 Weblate 3.x series

4.10.1 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 addons.
- 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.

4.10.2 Weblate 3.11.2

Released on February 22nd 2020.

- Fixed rendering of suggestions.
- Fixed some strings wrongly reported as having no words.

4.10.3 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.

4.10.4 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.
- 添加了对分组字符串整形的支持。
- 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 *String labels*.
- Added bulk edit addon.
- Added option for 强制检查.
- Increased default validity of confirmation links.
- Improved Matomo integration.
- Fixed 已被翻译 to correctly handle source string change.
- Extended automatic updates configuration by `AUTO_UPDATE`.
- LINGUAS addons now do full sync of translations in Weblate.

4.10.5 Weblate 3.10.3

Released on January 18th 2020.

- Support for translate-toolkit 2.5.0.

4.10.6 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 addon to configure YAML formatting.
- Fixed same plurals check to not fire on single plural form languages.
- Fixed regex search on some fields.

4.10.7 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 cleanup 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.

4.10.8 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.

4.10.9 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.

4.10.10 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 addon to bootstrap translations.
- Extend automatic translation to add suggestions.
- Display addon 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 addon 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.
- 增加了对 GitLab Merge Request 的支持。
- Improved display of repository status.
- Perform automated translation in the background.

4.10.11 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 addons.
- 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 addons 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.

4.10.12 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.

4.10.13 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 addon.
- Delay opening SMTP connection when sending notifications.
- Improved error logging.
- Allow custom location in MO generating addon.
- Added addons to cleanup old suggestions or comments.
- Added option to enable horizontal mode in the Zen editor.
- 提高了许多被链接组件的导入性能。
- 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.

4.10.14 Weblate 3.6.1

Released on April 26th 2019.

- Improved handling of monolingual XLIFF files.
- Fixed digest notifications in some corner cases.
- Fixed addon 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.

4.10.15 Weblate 3.6

Released on April 20th 2019.

- Add support for downloading user data.
- Addons are now automatically triggered upon installation.
- Improved instructions for resolving merge conflicts.
- Cleanup addon 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 addons 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.

4.10.16 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 addon 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.

4.10.17 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.
- Addon 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 addon now removes projects for unpaid billings after 45 days.

4.10.18 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 addon 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.

4.10.19 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.

4.10.20 Weblate 3.2.2

Released on October 20th 2018.

- Remove no longer needed Babel dependency.
- Updated language definitions.
- Improve documentation for addons, 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.

4.10.21 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.

4.10.22 Weblate 3.2

Released on October 6th 2018.

- Add install_addon management command for automated addon 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 addons.
- 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.

4.10.23 Weblate 3.1.1

Released on July 27th 2018.

- Fix testsuite failure on some setups.

4.10.24 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 addon 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.

4.10.25 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.

4.10.26 Weblate 3.0

Released on June 1st 2018.

- Rewritten access control.
- Several code cleanups that lead to moved and renamed modules.
- New addon for automatic component discovery.
- The import_project management command has now slightly different parameters.
- Added basic support for Windows RC files.
- New addon to store contributor names in PO file headers.
- The per component hook scripts are removed, use addons instead.
- Add support for collecting contributor agreements.
- Access control changes are now tracked in history.
- New addon 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.

4.11 Weblate 2.x series

4.11.1 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 addon to configure gettext output wrapping.
- New addon 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.
- 增加了对已提交更改重新排序的支持。

4.11.2 Weblate 2.19.1

Released on February 20th 2018.

- Fixed migration issue on upgrade from 2.18.
- Improved file upload API validation.

4.11.3 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 addons to customize translation workflows.

4.11.4 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.

4.11.5 Weblate 2.17.1

Released on October 13th 2017.

- Fixed running testsuite in some specific situations.
- Locales updates.

4.11.6 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.

4.11.7 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.

4.11.8 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.

4.11.9 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.

4.11.10 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.

4.11.11 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.

4.11.12 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.

4.11.13 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.

4.11.14 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 addon.
- 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.

4.11.15 Weblate 2.10.1

Released on Jan 20th 2017.

- Do not leak account existence on password reset form (CVE-2017-5537).

4.11.16 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.

4.11.17 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.

4.11.18 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.

4.11.19 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.

4.11.20 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.

4.11.21 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.
- 修正了全站检查失败的问题。

- 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 review (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.

4.11.22 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.

4.11.23 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.

4.11.24 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.

4.11.25 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.

4.11.26 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.

4.12 Weblate 1.x series

4.12.1 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.

4.12.2 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.

4.12.3 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.

4.12.4 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.

4.12.5 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.

4.12.6 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 string tests coverage.

4.12.7 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.

4.12.8 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 of counted strings with failing checks.
- Checking of requirements during setup.
- Documentation improvements.

4.12.9 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.

4.12.10 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.

4.13 Weblate 0.x series

4.13.1 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.

4.13.2 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.

4.13.3 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.

4.13.4 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.

4.13.5 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).

4.13.6 Weblate 0.4

Released on February 8th 2012.

- Added usage guide to documentation.
- Fixed API hooks not to require CSRF protection.

4.13.7 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.

4.13.8 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.

4.13.9 Weblate 0.1

Released on February 6th 2012.

- Initial release.

Python 模块索引

W

HTTP Routing Table

/

/api

/exports

/hooks

索引

符号

A

B

C

D

E

F

G

H

I

J

L

M

N

P

T

U

V