# User docs

1. **User docs**
   - 1.1 Weblate basics ........................................... 1
   - 1.2 Registration and user profile ............................... 1
   - 1.3 Translating using Weblate ................................. 10
   - 1.4 Downloading and uploading translations .................. 19
   - 1.5 Sözlük .................................................. 23
   - 1.6 Checks and fixups ........................................ 25
   - 1.7 Searching ............................................... 47
   - 1.8 Translation workflows .................................... 52
   - 1.9 Frequently Asked Questions ............................... 55
   - 1.10 Supported file formats ................................... 62
   - 1.11 Sürüm denetimi tümleşimi ............................... 82
   - 1.12 Weblate’s REST API ..................................... 89
   - 1.13 Weblate Client .......................................... 133
   - 1.14 Weblate’s Python API .................................. 137

2. **Administrator docs**
   - 2.1 Configuration instructions ............................... 140
   - 2.2 Weblate deployments ..................................... 199
   - 2.3 Upgrading Weblate ........................................ 200
   - 2.4 Backing up and moving Weblate ........................... 207
   - 2.5 Kimlik doğrulama ......................................... 213
   - 2.6 Erişim denetimi ......................................... 223
   - 2.7 Çeviriprojeleri ........................................... 233
   - 2.8 Dil tanımları ............................................. 249
   - 2.9 Sürekli yerelleştirme ..................................... 252
   - 2.10 Licensing translations .................................... 261
   - 2.11 Çeviri işleyişi ........................................... 263
   - 2.12 Checks and fixups ....................................... 269
   - 2.13 Makine çevirisı ......................................... 277
   - 2.14 Eklentiler ............................................... 284
   - 2.15 Çeviri Belleği ............................................ 300
   - 2.16 Configuration ............................................ 301
   - 2.17 Sample configuration ..................................... 330
   - 2.18 Management commands .................................... 346
   - 2.19 Duyurular ............................................... 357
   - 2.20 Component Lists ......................................... 359
   - 2.21 Optional Weblate modules ............................... 360
   - 2.22 Customizing Weblate .................................... 365
   - 2.23 Management interface ................................... 367
   - 2.24 Getting support for Weblate ............................. 375
<table>
<thead>
<tr>
<th>3</th>
<th>Contributor docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Weblate’e Katkıda Bulunma</td>
</tr>
<tr>
<td>3.2</td>
<td>Weblate kodlarına katkıda bulunmaya başlama</td>
</tr>
<tr>
<td>3.3</td>
<td>Weblate kaynak kodları</td>
</tr>
<tr>
<td>3.4</td>
<td>Debugging Weblate</td>
</tr>
<tr>
<td>3.5</td>
<td>Weblate internals</td>
</tr>
<tr>
<td>3.6</td>
<td>Eklenti geliştirme</td>
</tr>
<tr>
<td>3.7</td>
<td>Weblate frontend</td>
</tr>
<tr>
<td>3.8</td>
<td>Reporting issues in Weblate</td>
</tr>
<tr>
<td>3.9</td>
<td>Weblate testsuite and continuous integration</td>
</tr>
<tr>
<td>3.10</td>
<td>Data schemas</td>
</tr>
<tr>
<td>3.11</td>
<td>Releasing Weblate</td>
</tr>
<tr>
<td>3.12</td>
<td>Güvenlik ve gizlilik</td>
</tr>
<tr>
<td>3.13</td>
<td>Weblate hakkında</td>
</tr>
<tr>
<td>3.14</td>
<td>Lisans</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4</th>
<th>Change History</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Weblate 4.9</td>
</tr>
<tr>
<td>4.2</td>
<td>Weblate 4.8.1</td>
</tr>
<tr>
<td>4.3</td>
<td>Weblate 4.8</td>
</tr>
<tr>
<td>4.4</td>
<td>Weblate 4.7.2</td>
</tr>
<tr>
<td>4.5</td>
<td>Weblate 4.7.1</td>
</tr>
<tr>
<td>4.6</td>
<td>Weblate 4.7</td>
</tr>
<tr>
<td>4.7</td>
<td>Weblate 4.6.2</td>
</tr>
<tr>
<td>4.8</td>
<td>Weblate 4.6.1</td>
</tr>
<tr>
<td>4.9</td>
<td>Weblate 4.6</td>
</tr>
<tr>
<td>4.10</td>
<td>Weblate 4.5.3</td>
</tr>
<tr>
<td>4.11</td>
<td>Weblate 4.5.2</td>
</tr>
<tr>
<td>4.12</td>
<td>Weblate 4.5.1</td>
</tr>
<tr>
<td>4.13</td>
<td>Weblate 4.5</td>
</tr>
<tr>
<td>4.14</td>
<td>Weblate 4.4.2</td>
</tr>
<tr>
<td>4.15</td>
<td>Weblate 4.4.1</td>
</tr>
<tr>
<td>4.16</td>
<td>Weblate 4.4</td>
</tr>
<tr>
<td>4.17</td>
<td>Weblate 4.3.2</td>
</tr>
<tr>
<td>4.18</td>
<td>Weblate 4.3.1</td>
</tr>
<tr>
<td>4.19</td>
<td>Weblate 4.3</td>
</tr>
<tr>
<td>4.20</td>
<td>Weblate 4.2.2</td>
</tr>
<tr>
<td>4.21</td>
<td>Weblate 4.2.1</td>
</tr>
<tr>
<td>4.22</td>
<td>Weblate 4.2</td>
</tr>
<tr>
<td>4.23</td>
<td>Weblate 4.1.1</td>
</tr>
<tr>
<td>4.24</td>
<td>Weblate 4.1</td>
</tr>
<tr>
<td>4.25</td>
<td>Weblate 4.0.4</td>
</tr>
<tr>
<td>4.26</td>
<td>Weblate 4.0.3</td>
</tr>
<tr>
<td>4.27</td>
<td>Weblate 4.0.2</td>
</tr>
<tr>
<td>4.28</td>
<td>Weblate 4.0.1</td>
</tr>
<tr>
<td>4.29</td>
<td>Weblate 4.0</td>
</tr>
<tr>
<td>4.30</td>
<td>Weblate 3.x series</td>
</tr>
<tr>
<td>4.31</td>
<td>Weblate 2.x series</td>
</tr>
<tr>
<td>4.32</td>
<td>Weblate 1.x series</td>
</tr>
<tr>
<td>4.33</td>
<td>Weblate 0.x series</td>
</tr>
</tbody>
</table>

**Python Modül Dizini** | 446 |

**HTTP Routing Table** | 447 |

**Dizin** | 450 |
1.1 Weblate basics

1.1.1 Proje ve bileşen yapısı

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

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

Ayrıca bakınız:
../devel/integration

1.2 Registration and user profile

1.2.1 Kayıt

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 Panel

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

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

İpucu: 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 *Component Lists*.

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

---

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

---

### 1.2.3 Kullanıcı profili

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.

**Not:** All language selections only offer currently translated languages.

---

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

---

### Diller

#### 1.2.4 Arayüz dili

Kullanıcı arayüzünü göstermek istediğiniz dili seçin.
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.

<table>
<thead>
<tr>
<th>Component</th>
<th>Translated</th>
<th>Untranslated</th>
<th>Untranslated words</th>
<th>Checks</th>
<th>Suggestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>WeblateOrg/Android — Czech 🇨🇿</td>
<td>76%</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WeblateOrg/Django — Hungarian 🇭🇺 GMPL-3.0</td>
<td>60%</td>
<td>8</td>
<td>109</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>WeblateOrg/Django — Czech 🇨🇿 GMPL-3.0</td>
<td>96%</td>
<td>1</td>
<td>12</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>WeblateOrg/Django — Hebrew 🇮🇱 GMPL-3.0</td>
<td>92%</td>
<td>2</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WeblateOrg/Django — Hungarian 🇭🇺 GMPL-3.0</td>
<td>96%</td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WeblateOrg/Django — Hebrew 🇮🇱 GMPL-3.0</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WeblateOrg/Django — Czech 🇨🇿 GMPL-3.0</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WeblateOrg/Language names — Czech 🇨🇿</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WeblateOrg/Language names — Hungarian 🇭🇺</td>
<td>81%</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WeblateOrg/Language names — Hebrew 🇮🇱</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WeblateOrg/WeblateOrg — Hungarian 🇭🇺 GMPL-3.0</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WeblateOrg/WeblateOrg — Czech 🇨🇿 GMPL-3.0</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WeblateOrg/WeblateOrg — Hebrew 🇮🇱 GMPL-3.0</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
İkincil diller

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:

![Translation interface with Hebrew and Czech languages and a screenshot of a Hebrew string with context information.]
1.2.5 Tercihler

Varsayılan panel görünümü

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.

Ayrıca bakınız:

Component Lists

Düzenleyici bağlantısı

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

İpucu: 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.

Ayrıca bakınız:

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

1.2.6 Bildirimler

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

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

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

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

Not: You will not receive notifications for your own actions.
1.2. Registration and user profile
1.2.7 Hesap

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 kullanıcılara verilerini dışa aktarma).

Not: 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.
1.2. Registration and user profile
1.2.8 Profil

Bu sayfada tüm alanlar isteğe bağlıdır ve istediğiniz zaman silinebilir ve bunları doldurarak, bu verileri kullanıcı profilinizin görüldüğü her yerde paylaşmamızı izin vermiş olursunuz.

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

1.2.9 Lisanslar

1.2.10 API erişimi

You can get or reset your API access token here.

1.2.11 Denetim günlüğü

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.

Ayrıca bakınız:

Running behind reverse proxy

1.3 Translating using Weblate

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

Overall, there are two modes of translation:

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

Please see Translation workflows for more info on translation workflow.

Options for translation project visibility:

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

Ayrıca bakınız:

Erişim denetimi, Translation workflows

1.3.1 Çeviri projeleri

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

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

Not: Actual permissions might vary depending on your Weblate configuration.

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

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

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

Markdownt söz dizimini tüm yorumlarda kullanabilir ve kullanıcılardan bahsedebilirsiniz.

Ayrıca bakınız: report-source, Source strings reviews, Kaynak gözden geçirmerelerini etkinleştir

1.3.5 Çeşitler

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

Ayrıca bakınız: variants, Çeşitler

1.3.6 Etiketler

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

Ayrıca bakınız: labels

1.3.7 Translating

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

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

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

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

Çoğullar

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

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

Çoğul formülü

**Translation**

**Singular**

% (count) s word

**Plural**

% (count) s words

**Czech, One**

% (count) s slovo

**Czech, Few**

% (count) s slova

**Czech, Other**

% (count) s slova

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

**Glossary**

No related strings found in the glossary.

**Add term to glossary**

**String information**

Screenshot context

No screenshot currently associated.

**Add screenshot**

Explanation

No explanation currently provided.

Labels

No labels currently set.

Flags

python-format

Source string location

webtext/templates/translation.html

String age

13 seconds ago

Source string age

14 seconds ago

Translation file

webtext/locale/cs/C_MESSAGES

5/django-po, string 5

---

**Bölüm 1. User docs**
Keyboard shortcuts

2.18 sürümünde değişti: 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:

<table>
<thead>
<tr>
<th>Klavye kısayolu</th>
<th>Açıklama</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt+Home</td>
<td>Geçerli aramadaki ilk çeviriye git.</td>
</tr>
<tr>
<td>Alt+End</td>
<td>Geçerli aramadaki son çeviriye git.</td>
</tr>
<tr>
<td>Alt+PageUp or Ctrl ↑</td>
<td>Geçerli aramadaki önceki çeviriye git.</td>
</tr>
<tr>
<td>Alt PageDown or</td>
<td>Geçerli aramadaki sonraki çeviriye git.</td>
</tr>
<tr>
<td>Alt Enter or Ctrl+Enter or Cmd+Enter</td>
<td>Geçerli çeviriyi kaydet.</td>
</tr>
<tr>
<td>Ctrl+Shift+Enter or</td>
<td>Düzenleme gerekli işaretini kaldır ve çeviriyi gönder.</td>
</tr>
<tr>
<td>Cmd+Shift+Enter</td>
<td></td>
</tr>
<tr>
<td>Ctrl+E or Ctrl+M</td>
<td>Focus translation editor.</td>
</tr>
<tr>
<td>Ctrl+U or Cmd+U</td>
<td>Focus comment editor.</td>
</tr>
<tr>
<td>Ctrl+I to Ctrl+9 or</td>
<td>Copies placeable of given number from source string.</td>
</tr>
<tr>
<td>Cmd+1 to Cmd+9</td>
<td>Copy the machine translation of given number to current translation.</td>
</tr>
<tr>
<td>Ctrl+I+1 to 9 or</td>
<td>Kusur denetimleri listesindeki bir öğeyi yok say.</td>
</tr>
<tr>
<td>Cmd+I+1 to 9</td>
<td>Yakındaki dizgiler sekmesini gösterir.</td>
</tr>
<tr>
<td>Ctrl+J or Cmd+J</td>
<td>Focus search field.</td>
</tr>
<tr>
<td>Ctrl+S or Cmd+S</td>
<td>Kaynak dizgiyi kopyala.</td>
</tr>
<tr>
<td>Ctrl+O or Cmd+O</td>
<td>Düzenleme gerekli bayrağını değiştir.</td>
</tr>
</tbody>
</table>

Visual keyboard

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

The shown symbols factor into three categories:

- User configured characters defined in the Kullanıcı profili
- Per-language characters provided by Weblate (e.g. quotes or RTL specific characters)
- Characters configured using SPECIAL_CHARS
Translation context

This contextual description provides related info about the current string.

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

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

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

Diğer kullanımlar 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 Tutarsız). You can choose which one to use.
The Weblate Manual, Yayım 4.9

Çeviri Belleği Look at similar strings translated in past, see Çeviri Belleği.

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

En son değişiklikler List of people whom have changed this message recently using Weblate.

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

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

Translation history

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

Translated string length

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

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

1.3.8 Otomatik öneriler

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

Ayrıca bakınız:
You can find the list of supported tools in Makine çevirişi.

1.3.9 Kendiliğinden çeviri

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.

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

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

**Ayrıca bakınız:**

*Keeping translations same across components*
1.3.10 Rate limiting

To avoid abuse of the interface, rate limiting is applied to several operations like searching, sending contact forms or translating. If affected by it, you are blocked for a certain period until you can perform the operation again. Default limits and fine-tuning is described in the administrative manual, see Rate limiting.

1.3.11 Arama ve değiştirme

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

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

1.3.12 Toplu düzenleme

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

- Changing string state (for example to approve all unreviewed strings).
- Adjust translation flags (see Customizing behavior using flags)
- Adjust string labels (see labels)

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

Ayrıca bakınız:

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.

Not: Available options might be limited by access control settings.

1.4.1 Downloading translations

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

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

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

- gettext (PO)
- Gettext uzantılı XLIFF
- XLIFF 1.1
- TermBase eXchange
- Translation Memory eXchange
- gettext MO (only available when translation is using gettext PO)
- CSV
- Excel Open XML
- JSON (sadece tek dilli çeviriler için erişilebilir)
- Android Dizi Kaynağı (sadece tek dilli çeviriler için erişilebilir)
- iOS Dizileri (sadece tek dilli çeviriler için erişilebilir)

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

**Ayrıca bakınız:**

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

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

Supported file formats

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

Ayrıca bakınız:

Supported file formats

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

These are the choices presented when uploading translation files:

Çeviri olarak ekle (translate) Imported translations are added as translations. This is the most common use-case, and the default behavior.

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

Öneri olarak ekle (suggest) Imported translations are added as suggestions, do this when you want to have your uploaded strings reviewed.

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

Düzenlenmesi gereken çeviri olarak ekle (fuzzy) Imported translations are added as translations needing edit.

- This can be useful when you want translations to be used, but also reviewed.

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

Mevcut çeviri dosyasını değiştir (replace) Existing file is replaced with new content. This can lead to loss of existing translations, use with caution.

Kaynak dizgileri güncelle (source) Updates source strings in bilingual translation file. This is similar to what POT ile eşleşmesi için PO dosyalarını güncelle (msgmerge) does.

- Bu seçeneğin yaniçınız bazı dosya biçimleri için desteklenir.

Yeni dizgiler ekle (add) Adds new strings to the translation. It skips the one which already exist.

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

- This option is available only with Dizgileri yönet turned on.

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

Ayrıca bkz:

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 Sözlük

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

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

1.5.1 Managing glossaries

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

Use any component as a glossary by turning on **Sözlük olarak kullan.** You can create multiple glossaries for one project.

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

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

Tüm sözlük terimlerine göz atabilirsiniz:
 veya bunları herhangi bir çeviri olarak düzenleyin.

1.5.2 Sözlük terimleri

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

4.5 sürümünde geldi.

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

Ayrıca bakınız:

Customizing behavior using flags

Yasaklı çeviriler

4.5 sürümünde geldi.

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

Ayrıca bakınız:

Customizing behavior using flags

Terminoloji

4.5 sürümünde geldi.

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

Ayrıca bakınız:

Customizing behavior using flags

Çeşitler

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

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

Ayrıca bakınız:

variants

1.6 Checks and fixups

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

Once submitting a translation with a failing check, this is immediately shown to the user:
The translation has been saved, however there are some newly failing checks: Missing plurals, Python format

**Things to check**

- **Python format**
  - Following format strings are missing: `%(count)s`
  - Dismiss

- **Missing plurals**
  - Some plural forms are not translated
  - Dismiss

**Glossary**

- **English**
- **Czech**
  - No related strings found in the glossary
  - Add term to glossary

**String information**

- Screenshot context
  - No screenshot currently associated
  - Add screenshot

- Explanation
  - No explanation currently provided

- Labels
  - No labels currently set

- Flags
  - `python-format`

- Source string location
  - `weblate/templates/translation.html`

- Strings age
  - 20 seconds ago

- Translation file
  - `weblate/locale/cs/LC_MESSAGES/django.po, string 5`
1.6.1 Automatic fixups

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

Ayrıca bakınız:
AUTOFIX_LIST

1.6.2 Quality checks

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

Ayrıca bakınız:
CHECK_LIST, Customizing behavior using flags

1.6.3 Translation checks

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

BBcode işaretlemesi

- **Özet** Çevirideki BBcode kaynakla eşleşmiyor
- **Kapsam** çevrilmiş dizgiler
- **Check class** weblate.checks.markup.BBCodeCheck
- **Flag to ignore** ignore-bbcode

BBCode represents simple markup, like for example highlighting important parts of a message in bold font, or italics. This check ensures they are also found in translation.

**Not:** The method for detecting BBcode is currently quite simple so this check might produce false positives.

Art arda yinelenen sözcükler

4.1 sürümünde geldi.

- **Özet** Metin aynı sözcüğü üst üste iki kez içeriyor:
- **Kapsam** çevrilmiş dizgiler
- **Check class** weblate.checks.duplicate.DuplicateCheck
- **Flag to ignore** ignore-duplicate

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

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

4.5 sürümünde geldi.

Özet Çeviri, bir sözlükte tanımlanan terimleri takip etmiyor.
Kapsam çevrilmış dizgiler
Check class weblate.checks.glossary.GlossaryCheck
Flag to enable check-glossary
Flag to ignore ignore-check-glossary

This check has to be turned on using check-glossary flag (see Customizing behavior using flags). Please consider following prior to enabling it:

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

Ayrıca bakınız:
Sözlük, Customizing behavior using flags, Çeviri işaretleri

Çift boşluk

Özet Çeviri çift boşluk içeriyor
Kapsam çevrilmış dizgiler
Check class weblate.checks.chars.DoubleSpaceCheck
Flag to ignore ignore-double-space

Checks that double space is present in translation to avoid false positives on other space-related checks.
Check is false when double space is found in source meaning double space is intentional.

Formatted strings

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

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

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

İpucu: In case specific format check is not available in Weblate, you can use generic Yer tutucular.

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

Özet AngularJS ilişkilendirme dizgileri kaynakla eşleşmiyor
Kapsam çevrilmiş dizgiler
Check class weblate.checks.angularjs.AngularJSInterpolationCheck
Flag to enable angularjs-format
Flag to ignore ignore-angularjs-format
Named format string example Your balance is {{amount}} {{ currency }}

Ayrıca bakınız:
Formatted strings, AngularJS text interpolation

1.6. Checks and fixups
**C biçim**

**Özet**  C biçimindeki dizgi kaynakla eşleşmiyor

**Kapsam**  çevriliş dizgiler

**Check class**  weblate.checks.format.CFormatCheck

**Flag to enable**  c-format

**Flag to ignore**  ignore-c-format

**Simple format string example**  There are %d apples

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

Ayrıca bakınız:

*Formatted strings, C format strings, C printf format*

**C# biçim**

**Özet**  C# biçimindeki dizgi kaynakla eşleşmiyor

**Kapsam**  çevriliş dizgiler

**Check class**  weblate.checks.format.CSharpFormatCheck

**Flag to enable**  c-sharp-format

**Flag to ignore**  ignore-c-sharp-format

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

Ayrıca bakınız:

*Formatted strings, C# String Format*

**ECMAScript şablon değişmezleri**

**Özet**  ECMAScript şablon değişmezleri kaynakla eşleşmiyor

**Kapsam**  çevriliş dizgiler

**Check class**  weblate.checks.format.ESTemplateLiteralsCheck

**Flag to enable**  es-format

**Flag to ignore**  ignore-es-format

**Interpolation example**  There are ${number} apples

Ayrıca bakınız:

*Formatted strings, Template literals*
i18next ilişkilendirmesi

4.0 sürümüne geldi.

Özet  i18next ilişkilendirmesi kaynakla eşleşmiyor

Kapsam  çevrilmiş dizgiler

Check class  weblate.checks.format.I18NextInterpolationCheck

Flag to enable  i18next-interpolation

Flag to ignore  ignore-i18next-interpolation

Interpolation example  There are {{number}} apples

Nesting example  There are $t(number) apples

Ayrıca bakınız:

Formatted strings, i18next interpolation

Java biçimi

Özet  Java biçimindeki dizgi kaynakla eşleşmiyor

Kapsam  çevrilmiş dizgiler

Check class  weblate.checks.format.JavaFormatCheck

Flag to enable  java-format

Flag to ignore  ignore-java-format

Simple format string example  There are %d apples

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

Ayrıca bakınız:

Formatted strings, Java Format Strings

Java MessageFormat

Özet  Java MessageFormat dizgisi kaynakla eşleşmiyor

Kapsam  çevrilmiş dizgiler

Check class  weblate.checks.format.JavaMessageFormatCheck

Flag to enable uncodintionally  java-messageformat

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

Flag to ignore  ignore-java-messageformat

Position format string example  There are {0} apples

Ayrıca bakınız:

Formatted strings, Java MessageFormat

1.6. Checks and fixups
JavaScript biçimi

Özet  JavaScript biçimindeki dizgi kaynakla eşleşmiyor
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.format.JavaScriptFormatCheck
Flag to enable  javascript-format
Flag to ignore  ignore-javascript-format
Simple format string example  There are %d apples

Ayrıca bakınız:
Formatted strings, JavaScript formatting strings

Lua biçimi

Özet  Lua biçimindeki dizgi kaynakla eşleşmiyor
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.format.LuaFormatCheck
Flag to enable  lua-format
Flag to ignore  ignore-lua-format
Simple format string example  There are %d apples

Ayrıca bakınız:
Formatted strings, Lua formatting strings

Object Pascal biçimı

Özet  Object Pascal biçimindeki dizgi kaynakla eşleşmiyor
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.format.ObjectPascalFormatCheck
Flag to enable  object-pascal-format
Flag to ignore  ignore-object-pascal-format
Simple format string example  There are %d apples

Ayrıca bakınız:
Formatted strings, Object Pascal formatting strings Free Pascal formatting strings Delphi formatting strings

Yüzde yer tutucuları

4.0 sürümünde geldi.

Özet  Yüzde yer tutucuları kaynakla eşleşmiyor
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.format.PercentPlaceholdersCheck
Flag to enable  percent-placeholders
Flag to ignore  ignore-percent-placeholders
Simple format string example  There are %number% apples

Ayrıca bakınız:
Formatted strings,

Perl biçimi

<table>
<thead>
<tr>
<th>Özet</th>
<th>Perl biçimindeki dizgi kaynakla eşleşmiyor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kapsam</td>
<td>çevrilmiş dizgiler</td>
</tr>
<tr>
<td>Check class</td>
<td>weblate.checks.format.PerlFormatCheck</td>
</tr>
<tr>
<td>Flag to enable</td>
<td>perl-format</td>
</tr>
<tr>
<td>Flag to ignore</td>
<td>ignore-perl-format</td>
</tr>
</tbody>
</table>

Simple format string example  There are %d apples

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

Ayrıca bakınız:
Formatted strings, Perl sprintf, Perl Format Strings

PHP biçimi

<table>
<thead>
<tr>
<th>Özet</th>
<th>PHP biçimindeki dizgi kaynakla eşleşmiyor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kapsam</td>
<td>çevrilmiş dizgiler</td>
</tr>
<tr>
<td>Check class</td>
<td>weblate.checks.format.PHPFormatCheck</td>
</tr>
<tr>
<td>Flag to enable</td>
<td>php-format</td>
</tr>
<tr>
<td>Flag to ignore</td>
<td>ignore-php-format</td>
</tr>
</tbody>
</table>

Simple format string example  There are %d apples

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

Ayrıca bakınız:
Formatted strings, PHP sprintf documentation, PHP Format Strings

Python ayracı biçimi

<table>
<thead>
<tr>
<th>Özet</th>
<th>Python ayracı biçimindeki dizgi kaynakla eşleşmiyor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kapsam</td>
<td>çevrilmiş dizgiler</td>
</tr>
<tr>
<td>Check class</td>
<td>weblate.checks.format.PythonBraceFormatCheck</td>
</tr>
<tr>
<td>Flag to enable</td>
<td>python-brace-format</td>
</tr>
<tr>
<td>Flag to ignore</td>
<td>ignore-python-brace-format</td>
</tr>
</tbody>
</table>

Simple format string  There are {} apples

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

Ayrıca bakınız:
Formatted strings, Python brace format, Python Format Strings
Python biçimi

Özet  Python biçim dizgisi kaynakla eşleşmiyor
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.format.PythonFormatCheck
Flag to enable  python-format
Flag to ignore  ignore-python-format
Simple format string  There are %d apples
Named format string example  Your balance is %(amount)d %(currency)s

Ayrıca bakınız:
Formatted strings, Python string formatting, Python Format Strings

Qt biçimi

Özet  Qt biçimindeki dizgi kaynakla eşleşmiyor
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.qt.QtFormatCheck
Flag to enable  qt-format
Flag to ignore  ignore-qt-format
Position format string example  There are %1 apples

Ayrıca bakınız:
Formatted strings, Qt QString::arg()

Qt çokul biçimi

Özet  Qt çokul biçimindeki dizgi kaynakla eşleşmiyor
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.qt.QtPluralCheck
Flag to enable  qt-plural-format
Flag to ignore  ignore-qt-plural-format
Plural format string example  There are %Ln apple(s)

Ayrıca bakınız:
Formatted strings, Qt i18n guide
Ruby biçimi

**Özet**  Ruby biçim dizgisi kaynakla eşleşmiyor

**Kapsam**  çevrilen dizgiler

**Check class**  weblate.checks.ruby.RubyFormatCheck

**Flag to enable**  ruby-format

**Flag to ignore**  ignore-ruby-format

**Simple format string example**  There are %d apples

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

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

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

Ayrıca bakınız:

*Formatted strings, Ruby Kernel#sprintf*

Şema biçimi

**Özet**  Şema biçim dizgisi kaynakla eşleşmiyor

**Kapsam**  çevrilen dizgiler

**Check class**  weblate.checks.format.SchemeFormatCheck

**Flag to enable**  scheme-format

**Flag to ignore**  ignore-scheme-format

**Simple format string example**  There are ~d apples

Ayrıca bakınız:

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

Vue I18n biçimlendirmesi

**Özet**  Vue I18n biçimlendirmesi kaynakla eşleşmiyor

**Kapsam**  çevrilen dizgiler

**Check class**  weblate.checks.format.VueFormattingCheck

**Flag to enable**  vue-format

**Flag to ignore**  ignore-vue-format

**Adlandırılmış biçimlendirme**  There are {count} apples

**Rails I18n biçimlendirmesi**  There are %{count} apples

**Bağlantılı yerel iletileri**  @:message.dio @:message.the_world!

Ayrıca bakınız:

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

Özet  Bu dizgi geçmişte çevrilmüş
Kapsam  tüm dizgiler
Check class  weblate.checks.consistency.TranslatedCheck
Flag to ignore  ignore-translated

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

Tutarsız

Özet  Bu dizginin bu projede birden fazla çevirisi var ya da bazı bileşenlerde çevrilmemiş.
Kapsam  tüm dizgiler
Check class  weblate.checks.consistency.ConsistencyCheck
Flag to ignore  ignore-inconsistent

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

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

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

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

You can use Kendiliğinden çeviri addon to automate translating of newly added strings which are already translated in another component.

Ayrıca bakınız:  Keeping translations same across components

Kashida harfi kullanıldı

3.5 sürümünde geldi.

Özet  Dekoratif kashida harfleri kullanılmamalıdır
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.chars.KashidaCheck
Flag to ignore  ignore-kashida

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

Ayrıca bakınız:  Kashida on Wikipedia
Markdown bağlantıları

3.5 sürümünde geldi.

Özet  Markdown bağlantıları kaynakla eşleşmiyor
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.markup.MarkdownLinkCheck
Flag to enable  md-text
Flag to ignore  ignore-md-link

Markdown links do not match source.

Ayrıca bakınız:
Markdown links

Markdown başvuruları

3.5 sürümünde geldi.

Özet  Markdown başvuru başvuruları kaynakla eşleşmiyor
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.markup.MarkdownRefLinkCheck
Flag to enable  md-text
Flag to ignore  ignore-md-reflink

Markdown link references do not match source.

Ayrıca bakınız:
Markdown links

Markdown sözdizimi

3.5 sürümünde geldi.

Özet  Markdown sözdizimi kaynakla eşleşmiyor
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.markup.MarkdownSyntaxCheck
Flag to enable  md-text
Flag to ignore  ignore-md-syntax

Markdown sözdizimi kaynakla eşleşmiyor

Ayrıca bakınız:
Markdown span elements
Çevirinin en çok uzunluğu

Özet  Çeviri verilen uzunluğu aşmamalı
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.chars.MaxLengthCheck
Flag to enable  max-length
Flag to ignore  ignore-max-length

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

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

İpucu:  This check looks at number of chars, what might not be the best metric when using proportional fonts to render the text. The Çeviri maksimum boyutu check does check actual rendering of the text.

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

Çeviri maksimum boyutu

Özet  Çevrilmiş metnin belirtilen boyutu aşmaması gerekir
Kapsam  çevrilmiş dizgiler
Check class  weblate.checks.render.MaxSizeCheck
Flag to enable  max-size
Flag to ignore  ignore-max-size

3.7 sürümünde geldi.
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 Customizing behavior using flags), 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

İpucu:  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.

Ayrıca bakınız:
Managing fonts, Customizing behavior using flags, Çevirinin en çok uzunluğu
Özet Çevirideki \n sayısı kaynakla eşleşmiyor

Kapsam çevrilmiş dizgiler

Check class weblate.checks.chars.EscapedNewlineCountingCheck

Flag to ignore ignore-escaped-newline

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

İkinci nokta üst üste uyuşmadı

Özet Kaynağın ve çeviri ikinci nokta üst üste ile bitmiyor

Kapsam çevrilmiş dizgiler

Check class weblate.checks.chars.EndColonCheck

Flag to ignore ignore-end-colon

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

Ayrıca bakınız:

Colon on Wikipedia

Üç nokta uyuşmadı

Özet Kaynağın ve çevirinin ikisi üç noktası ile bitmiyor

Kapsam çevrilmiş dizgiler

Check class weblate.checks.chars.EndEllipsisCheck

Flag to ignore ignore-end-ellipsis

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

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

Ayrıca bakınız:

Ellipsis on Wikipedia

Ünlem işareti uyuşmadı

Özet Kaynak ve çeviri bir ünlem işaret ile bitmiyor

Kapsam çevrilmiş dizgiler

Check class weblate.checks.chars.EndExclamationCheck

Flag to ignore ignore-end-exclamation

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

Ayrıca bakınız:

Exclamation mark on Wikipedia
Nokta işareti uyuşmadı

Özet Kaynağın ve çevirinin ikiside nokta ile bitmiyor
Kapsam çevrilmiş dızgiler
Check class weblate.checks.chars.EndStopCheck
Flag to ignore ignore-end-stop

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

Ayrıca bakınız:
Full stop on Wikipedia

Soru işareti uyuşmadı

Özet Kaynağın ve çeviri bir soru işareti ile bitmiyor
Kapsam çevrilmiş dızgiler
Check class weblate.checks.chars.EndQuestionCheck
Flag to ignore ignore-end-question

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

Ayrıca bakınız:
Question mark on Wikipedia

Noktalı virgül uyuşmadı

Özet Kaynağın ve çevirinin ikiside noktalı virgül ile bitmiyor
Kapsam çevrilmiş dızgiler
Check class weblate.checks.chars.EndSemicolonCheck
Flag to ignore ignore-end-semicolon

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

Ayrıca bakınız:
Semicolon on Wikipedia

Uyuşmayan satır sayıları

Özet Çevirideki yeni satır sayısı kaynakla eşleşmiyor
Kapsam çevrilmiş dızgiler
Check class weblate.checks.chars.NewLineCountCheck
Flag to ignore ignore-newline-count

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

- **Özet**: Bazı çoğul biçimler çevrilmemiş
- **Kapsam**: çevrilmiş dizgiler
- **Check class**: `weblate.checks.consistency.PluralsCheck`
- **Flag to ignore**: `ignore-plurals`

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

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

Yer tutucular

3.9 sürümünde geldi.

- **Özet**: Bazı yer tutucuları eksik çeviri
- **Kapsam**: çevrilmiş dizgiler
- **Check class**: `weblate.checks.placeholders.PlaceholderCheck`
- **Flag to enable**: `placeholders`
- **Flag to ignore**: `ignore-placeholders`

4.3 sürümünde değişti: Düzenli ifadeyi yer tutucu olarak kullanabilirsiniz.

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

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

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

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

**Ayrıca bakınız:**

*Customizing behavior using flags*

Noktalama aralığı

3.9 sürümünde geldi.

- **Özet**: Çift noktalama işaretinden önce kırılmaz boşluk eksik
- **Kapsam**: çevrilmiş dizgiler
- **Check class**: `weblate.checks.chars.PunctuationSpacingCheck`
- **Flag to ignore**: `ignore-punctuation-spacing`

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

**Ayrıca bakınız:**

*French and English spacing on Wikipedia*
Düzenli ifade

3.9 sürümünde geldi.

Özet Çeviri düzenli ifadeyle eşleşmiyor:

Kapsam çevrilmiş dizgiler

Check class weblate.checks.placeholders.RegexCheck

Flag to enable regex

Flag to ignore ignore-regex

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

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

Aynı çoğullar

Özet Bazı çoğul biçimler aynı şekilde çevrilmiş

Kapsam çevrilmiş dizgiler

Check class weblate.checks.consistency.SamePluralsCheck

Flag to ignore ignore-same-plurals

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

Baştaki yeni satır

Özet Kaynağın ve çevirinin ikisi yeni satır ile başlamıyor

Kapsam çevrilmiş dizgiler

Check class weblate.checks.chars.BeginNewlineCheck

Flag to ignore ignore-begin-newline

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

Ayrıca bakınız:

Sondaki yeni satır

Baştaki boşluk

Özet Kaynağın ve çevirinin ikisi aynı boşluk sayısıyla başlamıyor

Kapsam çevrilmüş dizgiler

Check class weblate.checks.chars.BeginSpaceCheck

Flag to ignore ignore-begin-space

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

**Özet** Kaynağın ve çevirinin ikiside yeni satır ile bitmiyor

**Kapsam** çevrilmiş dizgiler

**Check class** weblate.checks.chars.EndNewlineCheck

**Flag to ignore** ignore-end-newline

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

**Ayrıca bakınız:**

**Baştaki yeni satır**

**Sondaki boşluk**

**Özet** Kaynağın ve çevirinin ikiside boşluk ile bitmiyor

**Kapsam** çevrilmiş dizgiler

**Check class** weblate.checks.chars.EndSpaceCheck

**Flag to ignore** ignore-end-space

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

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

**Değiştirilmemiş çeviri**

**Özet** Kaynak ve çeviri aynı

**Kapsam** çevrilmiş dizgiler

**Check class** weblate.checks.same.SameCheck

**Flag to ignore** ignore-same

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

This check can help find strings mistakenly untranslated.

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

**Ayrıca bakınız:**

*Component configuration, Customizing behavior using flags*
Güvenli olmayan HTML

3.9 sürümünde geldi.

Özet Çeviri güvenli olmayan HTML işaretlemesi kullanıyor

Kapsam çevrilmiş dizgiler

Check class weblate.checks.markup.SafeHTMLCheck

Flag to enable safe-html

Flag to ignore ignore-safe-html

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

Ayrıca bakınız:

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

URL

3.5 sürümünde geldi.

Özet Çeviri bir URL içermiyor

Kapsam çevrilmiş dizgiler

Check class weblate.checks.markup.URLCheck

Flag to enable url

Flag to ignore ignore-url

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

XML işaretlemesi

Özet Çevirideki XML etiketleri kaynakla eşleşmiyor

Kapsam çevrilmiş dizgiler

Check class weblate.checks.markup.XMLTagsCheck

Flag to ignore ignore-xml-tags

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

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

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

2.8 sürümünde geldi.

- **Özet**: Çeviri geçerli XML değil
- **Kapsam**: çevrilmiş dizgiler
- **Check class**: weblate.checks.markup.XMLValidityCheck
- **Flag to ignore**: ignore-xml-invalid

The XML markup is not valid.

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

---

**Sıfır genişlikli boşluk**

- **Özet**: Çeviri ekstra sıfır genişlikli boşluk karakteri içeriyor
- **Kapsam**: çevrilmiş dizgiler
- **Check class**: weblate.checks.chars.ZeroWidthSpaceCheck
- **Flag to ignore**: ignore-zero-width-space

Zero-width space (\u200B) 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.

**Ayrıca bakınız:**

Zero width space on Wikipedia

---

**1.6.4 Source checks**

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

**Üç nokta**

- **Özet**: Dizgi, üç nokta karakteri (…) yerine üç tane nokta (…) kullanır
- **Kapsam**: kaynak satırlar
- **Check class**: weblate.checks.source.EllipsisCheck
- **Flag to ignore**: ignore-ellipsis

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

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

**Ayrıca bakınız:**

Ellipsis on Wikipedia
**Uzun süre çevrilmeyen**

4.1 sürümünde geldi.

**Özet** Dizgi uzun zamanız çevrilmedi

**Kapsam** kaynak satırlar

**Check class** weblate.checks.source.LongUntranslatedCheck

**Flag to ignore** ignore-long-untranslated

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

**Birden çok kusur denetimi**

**Özet** Birkaç dildeki çevirilerin kusur denetimleri var

**Kapsam** kaynak satırlar

**Check class** weblate.checks.source.MultipleFailingCheck

**Flag to ignore** ignore-multiple-failures

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

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

**Birden çok adsız değişken**

4.1 sürümünde geldi.

**Özet** Dizgi bir den adlandırılmamış değişken var, bu da çevirmenlerin bunları yeniden sıralamasını imkansız Hale getirmektedir

**Kapsam** kaynak satırlar

**Check class** weblate.checks.format.MultipleUnnamedFormatsCheck

**Flag to ignore** ignore-unnamed-format

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

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

**Çoğulsuz**

**Özet** Dizgi çoğul olarak kullanılır, ancak çoğul biçimleri kullanmaz

**Kapsam** kaynak satırlar

**Check class** weblate.checks.source.OptionalPluralCheck

**Flag to ignore** ignore-optional-plural

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

For example with Gettext in Python it could be:

```python
from gettext import ngettext

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

3.9 sürümünde geldi.

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

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

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

1.7.2 Fields

- **source:** TEXT  Source string case insensitive search.
- **target:** TEXT  Target string case insensitive search.
- **context:** TEXT  Context string case insensitive search.
- **key:** TEXT  Key string case insensitive search.
- **note:** TEXT  Comment string case insensitive search.
- **location:** TEXT  Location string case insensitive search.
- **priority:** NUMBER  String priority.
- **added:** DATETIME  Timestamp for when the string was added to Weblate.
- **state:** TEXT  State search (approved, translated, needs-editing, empty, read-only), supports Field operators.
- **pending:** BOOLEAN  String pending for flushing to VCS.
- **has:** TEXT  Search for string having attributes - plural, context, suggestion, comment, check, dismissed-check, translation, variant, screenshot, flags, explanation, glossary, note.
- **is:** TEXT  Search for string states (pending, translated, untranslated).
- **language:** TEXT  String target language.
- **component:** TEXT  Component slug or name case insensitive search, see Component slug and Bileşen adı.
- **project:** TEXT  Project slug, see URL kısaltması.
- **changed_by:** TEXT  String was changed by author with given username.
- **changed:** DATETIME  String content was changed on date, supports Field operators.
- **change_time:** DATETIME  String was changed on date, supports Field operators, unlike changed this includes event which don’t change content and you can apply custom action filtering using change_action.
- **change_action:** TEXT  Filters on change action, useful together with change_time. Accepts English name of the change action, either quoted and with spaces or lowercase and spaces replaced by a hyphen. See Değişiklik arama for examples.
- **check:** TEXT  String has failing check.
- **dismissed_check:** TEXT  String has dismissed check.
- **comment:** TEXT  Search in user comments.
- **comment_author:** TEXT  Filter by comment author.
- **suggestion:** TEXT  Search in suggestions.
- **suggestion_author:** TEXT  Filter by suggestion author.
- **explanation:** TEXT  Açıklamalarda ara.
1.7.3 Boolean operators

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

1.7.4 Field operators

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

```
state:> =translated  State is translated or better (approved).
changed:[2019-03-01 to 2019-04-01]  Changed between two given dates.
```

1.7.5 Exact operators

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

1.7.6 Değişiklik arama

4.4 sürümünde geldi.

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

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

1.7.7 Regular expressions

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

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

1.7.8 Predefined queries

You can select out of predefined queries on the search page, this allows you to quickly access the most frequent searches:
The Weblate Manual, Yayım 4.9

Translation

English

Singular

%(count)s word

Plural

%(count)s words

Czech, One

%(count)s slovo

Czech, Few

%(count)s slova

Czech, Other

%(count)s slov

Plural formula: \( n=1 \) ? 0; \( n=2 \) &\( n=4 \) ? 1 : 2

Needs editing

Save

Suggest

Skip

Nearby strings: 20

Comments

Automatic suggestions

Other languages 3

History

New comment

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

Scope

Translation comment, discussions with other translators

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

New comment

You can use Markdown and mention users by @username.

Save

Explanation

No explanation currently provided.

Labels

No labels currently set.

Flags

python-format

Source string location

weblate/templates/translation.html

m18n

String age

13 seconds ago

Source string age

14 seconds ago

Translation file

weblate/locale/xx/LC_MESSAGES

$c/4$
1.7.9 Ordering the results

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

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

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

1.8.1 Translation access

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

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

1.8.2 Translation states

Each translated string can be in one of following states:

- **Çevrilmemiş**: Translation is empty, it might or not be stored in the file, depending on the file format.
- **Düzenleme gerekli**: Translation needs editing, this is usually the result of a source string change, fuzzy matching or translator action. The translation is stored in the file, depending on the file format it might be marked as needing edit (for example as it gets a fuzzy flag in the Gettext file).
- **Gözden geçirme için bekliyor**: Translation is made, but not reviewed. It is stored in the file as a valid translation.
- **Onaylı**: Translation has been approved in the review. It can no longer be changed by translators, but only by reviewers. Translators can only add suggestions to it.
- **Öneriler**: Suggestions are stored in Weblate only and not in the translation file.

The states are represented in the translation files when possible.

**İpucu:** In case file format you use does not support storing states, you might want to use Değişirmemiş çevirileri “Düzenleme gerekli” olarak işaretle addon to flag unchanged strings as needing editing.

**Ayrıca bakınız:**
Translation types capabilities, Translation workflows

1.8.3 Direct translation

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

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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
<th>Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gözden geçirmeleri etkinleştir</td>
<td>Kapalı</td>
<td>Configured at project level.</td>
</tr>
<tr>
<td>Önerileri etkinleştir</td>
<td>Açık</td>
<td>It is useful for users to be able to suggest when they are not sure.</td>
</tr>
<tr>
<td>Oneri oylaması</td>
<td>Kapalı</td>
<td></td>
</tr>
<tr>
<td>Önerileri kendiliğinden kabul et</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Translators group</td>
<td>Kullanıcılar</td>
<td>Ya da bkz. per-project access control ile Çevir.</td>
</tr>
<tr>
<td>Reviewers group</td>
<td>YOK</td>
<td>Not used.</td>
</tr>
</tbody>
</table>
1.8.4 Peer review

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

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

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
<th>Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gözden geçirmeleri etkinleştir</td>
<td>kapalı</td>
<td>Configured at project level.</td>
</tr>
<tr>
<td>Onerileri etkinleştiri</td>
<td>açık</td>
<td></td>
</tr>
<tr>
<td>Oneri oylaması</td>
<td>kapalı</td>
<td></td>
</tr>
<tr>
<td>Onerileri kendiliğinden kabul et</td>
<td>1</td>
<td>You can set higher value to require more peer reviews.</td>
</tr>
<tr>
<td>Translators group</td>
<td>Kullanıcılar</td>
<td>Ya da bkz <em>per-project access control</em> ile Çevir.</td>
</tr>
<tr>
<td>Reviewers group</td>
<td>YOK</td>
<td>Not used, all translators review.</td>
</tr>
</tbody>
</table>

1.8.5 Dedicated reviewers

2.18 sürümünde geldi: The proper review workflow is supported since Weblate 2.18.

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

- *Any user* can edit unapproved translations.
- *Gözden geçiren* dizgileri onaylayabilir/onaylamayabilir.
- *Reviewer* can edit all translations (including approved ones).
- Suggestions can also be used to suggest changes for approved strings.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
<th>Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gözden geçirmeleri etkinleştir</td>
<td>açık</td>
<td>Configured at project level.</td>
</tr>
<tr>
<td>Onerileri etkinleştiri</td>
<td>kapalı</td>
<td></td>
</tr>
<tr>
<td>Oneri oylaması</td>
<td>kapalı</td>
<td></td>
</tr>
<tr>
<td>Onerileri kendiliğinden kabul et</td>
<td>0</td>
<td>It is useful for users to be able to suggest when they are not sure.</td>
</tr>
<tr>
<td>Translators group</td>
<td>Kullanıcılar</td>
<td>Ya da bkz <em>per-project access control</em> ile Çevir.</td>
</tr>
<tr>
<td>Reviewers group</td>
<td>Gözden geçirenler</td>
<td>Or <em>Review</em> with <em>per-project access control</em>.</td>
</tr>
</tbody>
</table>

1.8.6 Turning on reviews

Reviews can be turned on in the project configuration, from the *Workflow* subpage of project settings (to be found in the *Manage → Settings* menu):
Depending on Weblate configuration, the setting might not be available to you. For example on Hosted Weblate this is not available for projects hosted for free.

### 1.8.7 Quality gateway for the source strings

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

By setting *Ara dil dosyası*, 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.
1.8.8 Source strings reviews

With Kaynak gözden geçirmelerini etkinleştir enabled, the review process can be applied on the source strings. Once enabled, users can report issues in the source strings. The actual process depends on whether you use bilingual or monolingual formats.

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

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

1.9 Frequently Asked Questions

1.9.1 Configuration

How to create an automated workflow?

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

1. Set up your Git repository to tell Weblate when there is any change, see Bildirim kancaları for info on how to do it.
2. Set a push URL at your Component configuration in Weblate, this allows Weblate to push changes to your repository.
3. Turn on İşlemeye yolla on your Component configuration in Weblate, this will make Weblate push changes to your repository whenever they happen at Weblate.

How to access repositories over SSH?

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

How to fix merge conflicts in translations?

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

If you’ve already encountered a merge conflict, the easiest way to solve all conflicts locally on your machine, is to add Weblate as a remote repository, merge it into upstream and fix any conflicts. Once you push changes back, Weblate will be able to use the merged version without any other special actions.
Not: 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.

```bash
# 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/.
# Update weblate remote:
git remote update weblate
# Merge Weblate changes:
git merge weblate/main
# Resolve conflicts:
edit ...
git add ...
... 
git commit
# Push changes to upstream repository, Weblate will fetch merge from there:
git push
# Open Weblate for translation:
wlc unlock
```

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

```bash
# Add and update Weblate remotes
git remote add weblate-one https://hosted.weblate.org/git/project/one/
git remote add weblate-second https://hosted.weblate.org/git/project/second/
git remote update weblate-one weblate-second
# Merge QA_4_7 branch:
git checkout QA_4_7
git merge weblate-one/QA_4_7
... # Resolve conflicts
git commit
# Merge main branch:
git checkout main
git merge weblates-second/main
... # Resolve conflicts
git commit
# Push changes to the upstream repository, Weblate will fetch the merge from there:
git push
```

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

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

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

(sonraki sayfaya devam)
```bash
# Update Weblate remote:
git remote update weblate

# Merge Weblate changes:
git merge weblate/main

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

git commit

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

**Ayrıca bakınız:**

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

**How do I translate several branches at once?**

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

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

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

**Ayrıca bakınız:**

*Keeping translations same across components*

**How to translate multi-platform projects?**

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

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

**Ayrıca bakınız:**

*Keeping translations same across components*
How to export the Git repository that Weblate uses?

There is nothing special about the repository, it lives under the \texttt{DATA\_DIR} directory and is named \texttt{vcs/\texttt{<project>}/\texttt{<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 \textit{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 \textit{How to create an automated workflow?}).
- You manually tell Weblate to push (it needs push access to the upstream repository).
- Somebody manually merges changes from the Weblate git repository into the upstream repository.
- Somebody rewrites history produced by Weblate (e.g. by eliminating merge commits), merges changes, and tells Weblate to reset the content in the upstream repository.

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

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

You can use \texttt{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:
   \begin{lstlisting}[language=bash]
   git submodule add git@example.com:project-translations.git path/to/translations
   \end{lstlisting}
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:
   \begin{lstlisting}[language=bash]
   git submodule update --remote path/to/translations
   \end{lstlisting}

Please consult the \texttt{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 \textit{Performance report} link in the admin interface, or open the \texttt{/manage/performance/} URL directly.

Why are all commits committed by Weblate \texttt{<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.

Ayrıca bakınız:

\textit{Component configuration}
1.9.2 Usage

How do I review the translations of others?

• There are several review based workflows available in Weblate, see Translation workflows.
• You can subscribe to any changes made in Bildirimler 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.

Ayrıca bakınız:
Translation workflows

How do I provide feedback on a source string?

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

Ayrıca bakınız:
report-source, Yorumlar

How can I use existing translations while translating?

• Paylaşılan çeviri belleği sayesinde, Weblate içindeki tüm çeviriler kullanılabilir.
• Mevcut çeviri belleği dosyalarını Weblate’e aktarabilirsiniz.
• 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.

Ayrıca bakınız:
Makine çevirisi, Otomatik öneriler, Çeviri Belleği

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.

Ayrıca bakınız:
updating-target-files
Where do language definitions come from and how can I add my own?

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

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

Ayrıca bakınız:

Dil tanımları

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.

**Not:** 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 *POT ile eşleşmesi için PO dosyalarını güncelle* (msgmerge).

Ayrıca bakınız:

updating-target-files

1.9.3 Troubleshooting

Requests sometimes fail with “too many open files” error

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

The easiest way to do this is to run:

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

Bölüm 1. User docs
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"]
```

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.

- Kaynak dile çevrinin istemesi durumunda, lütfen proje ayarlarındaki Kaynak dil seçeneğini değiştirin.
- Kaynak dilin çeviri dosyasına gerek olmaması durumunda, lütfen bunu depodan kaldırın.
- Kaynak dilin çeviri dosyasına ihtiyaç duyduğu, ancak Weblate tarafından yok sayılmının gerektiğini durumda, lütfen haric tutmak için Dil süzgeci seçeneğini ayarlayın.

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

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

1.9.4 Özellikler

Does Weblate support other VCSes than Git and Mercurial?

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

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

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

Not: 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.
How does Weblate credit translators?

Every change made in Weblate is committed into VCS under the translators name. This way every single change has proper authorship, and you can track it down using the standard VCS tools you use for code. Additionally, when the translation file format supports it, the file headers are updated to include the translator's name.

Ayrıca bakınız:

list_translators, ../devel/reporting

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

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

4.2 sürümünde değişti: Çevirmenler, bir projenin tüm bileşenlerini bir bütün olarak belirli bir dille çevirebilir.

Why does Weblate use language codes such sr_Latn or zh_Hant?

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

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

Not: Weblate defaults to POSIX style language codes with underscore, see Dil tanımları for more details.

Ayrıca bakınız:

Dil tanımları, Dil kodu stili, Adding new translations

1.10 Supported file formats

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

Ayrıca bakınız:

Translation Related File Formats

Not: When choosing a file format for your application, it’s better to stick some well established format in the tool-kit/platform you use. This way your translators can additionally use whatever tools they are used to, and will more likely contribute to your project.

1.10.1 Bilingual and monolingual formats

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

For correct use of monolingual files, Weblate requires access to a file containing complete list of strings to translate with their source—this file is called Tek dilli taban dil dosyası within Weblate, though the naming might vary in your paradigm.
Additionally this workflow can be extended by utilizing *Ara dil dosyası* to include strings provided by developers, but not to be used as is in the final strings.

1.10.2 Kendiliğinden algılama

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

1.10.3 Translation types capabilities

Capabilities of all supported formats:

<table>
<thead>
<tr>
<th>Format</th>
<th>Lingualty</th>
<th>Plurals</th>
<th>Comments</th>
<th>Context</th>
<th>Location</th>
<th>Flags</th>
<th>Additional states</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNU gettext</td>
<td>bilingual</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes²</td>
<td>needs editing</td>
</tr>
<tr>
<td>Mono-lingual gettext</td>
<td>mono</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes⁷</td>
<td>needs editing</td>
</tr>
<tr>
<td>XLIFF</td>
<td>both</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes⁷</td>
<td>needs editing</td>
</tr>
<tr>
<td>Java properties</td>
<td>both</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>m18n lang dosyalar</td>
<td>mono</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>GWT özellikleri</td>
<td>mono</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Joomla translations</td>
<td>mono</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Qt Linguist.ts</td>
<td>both</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes⁷</td>
<td>needs editing</td>
</tr>
<tr>
<td>Android string resources</td>
<td>mono</td>
<td>yes</td>
<td>yes⁷</td>
<td>no</td>
<td>no</td>
<td>yes⁷</td>
<td></td>
</tr>
<tr>
<td>Apple iOS strings</td>
<td>bilingual</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>PHP dizgileri</td>
<td>mono</td>
<td>no¹¹</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>JSON files</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>JSON i18next files</td>
<td>mono</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>go-i18n JSON files</td>
<td>mono</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>ARB File</td>
<td>mono</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>WebExtension .JSON</td>
<td>mono</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>.XML resource files</td>
<td>mono</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes⁷</td>
</tr>
</tbody>
</table>

sonraki sayfaya devam
### Table 1 – from the previous page

<table>
<thead>
<tr>
<th>Format</th>
<th>Linguality</th>
<th>Plurals</th>
<th>Comments</th>
<th>Context</th>
<th>Location</th>
<th>Flags</th>
<th>Additional states</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV files</td>
<td>both</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>needs editing</td>
</tr>
<tr>
<td>YAML files</td>
<td>mono</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Ruby YAML files</td>
<td>mono</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>DTD files</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Flat XML files</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows RC files</td>
<td>mono</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Excel Open XML</td>
<td>mono</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>needs editing</td>
</tr>
<tr>
<td>Uygulama mağaza sist. veri dosyaları</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Subtitle files</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>HTML files</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>OpenDocument Format</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>IDML Format</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>INI translations</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Inno Setup INI çevirileri</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>TermBase eXchange biçimi</td>
<td>bilingual</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Metin dosyaları</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Stringsdict biçimi</td>
<td>mono</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Fluent biçimi</td>
<td>mono</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

1 See **Bilingual and monolingual formats**
2 Plurals are necessary to properly localize strings with variable count.
3 Comments can be used to pass additional info about the string to translate.
4 Context is used to differentiate identical strings used in different scopes (for example, “Sun” can be used as an abbreviated name of the day “Sunday” or as the name of our closest star).
5 Location of a string in source code might help proficient translators figure out how the string is used.
6 Additional states supported by the file format in addition to “Not translated” and “Translated”.
7 The gettext type comments are used as flags.
8 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**.
9 XML comment placed before the `<string>` element, parsed as a developer comment.
10 The plurals are supported only for Laravel which uses in string syntax to define them, see **Localization in Laravel**.
11 Plurals are handled in the syntax of the strings and not exposed as plurals in Weblate.
Salt okunur dizgiler

3.10 sürümünde geldi.

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 Customizing behavior using flags.

1.10.4 GNU gettext

Özgür yazılım çevirmek için en yaygın olarak kullanılan biçim.

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

The bilingual gettext PO file typically looks like this:

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

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

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

Typical Weblate Component configuration

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>po/*.po</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dili taban dili dosyası</td>
<td>Empty</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>po/messages.pot</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>Gettext PO file</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:

devel/gettext, devel/sphinx, Gettext on Wikipedia, PO Files, “configure” dosyasındaki ALL_LINGUAS değişkenini güncelleyin, Gettext çıktıını özelleştirin, LINGUAS dosyasını güncelleyin, MO dosyalarını üretin, POT ile eşleşmesi için PO dosyalarını güncelleyin (msgmerge)

Monolingual gettext

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

The monolingual gettext PO file typically looks like this:

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

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

#: weblate/accounts/avatar.py:163
msgid "none-user"
msgstr "Žádný"
```
While the base language file will be:

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

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

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

| Typical Weblate Component configuration |
|-------------------------------|------------------|
| Dosya maskesi                  | po/*.po          |
| Tek dilli taban dil dosyası     | po/en.po         |
| Yeni çeviriler için şablon     | po/messages.pot  |
| Dosya biçimleri                | Gettext PO file (monolingual) |

### 1.10.5 XLIFF

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

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

**Ayrıca bakınız:**

XML Localization Interchange File Format (XLIFF) specification

### Translation states

3.3 sürümünde değişti: Weblate ignored the state attribute prior to the 3.3 release.

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

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

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

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

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

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

**Ayrıca bakınız:**

Dedicated reviewers
Whitespace and newlines in XLIFF

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

For example:

```xml
<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 Customizing behavior using flags) by using the weblate-flags attribute. Weblate also understands maxwidth and font attributes from the XLIFF specification:

```xml
<trans-unit id = "10" maxwidth = "100" size-unit = "pixel" font = "ubuntu;22:bold">
  <source>Source $s</source>
</trans-unit>
<trans-unit id = "20" maxwidth = "100" size-unit = "char" weblate-flags = "c-format">
  <source>Source $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 Managing fonts).

Dizgi anahtarlari

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

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

### Typical Weblate Component configuration for bilingual XLIFF

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>localizations/*.xliff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>Empty</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>localizations/en-US.xliff</td>
</tr>
</tbody>
</table>

| Dosya biçimi | XLIFF Translation File |

### Typical Weblate Component configuration for monolingual XLIFF

<table>
<thead>
<tr>
<th>File mask</th>
<th>localizations/*.xliff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>localizations/en-US.xliff</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>localizations/en-US.xliff</td>
</tr>
</tbody>
</table>

| Dosya biçimi | XLIFF Translation File |

Ayrıca bakınız:

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.

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

![Typical Weblate Component configuration](table)

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>src/app/Bundle_*.properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>src/app/Bundle.properties</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>Java Properties (ISO-8859-1)</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:

Java properties on Wikipedia, Mozilla and Java properties files, mi18n lang dosyaları, GWT özellikleri, updating-target-files, Java özellikleri dosyasını biçimlendirir, Çeviri dosyası temizle

1.10.7 mi18n lang dosyaları

4.7 sürümünde geldi.

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

![Typical Weblate Component configuration](table)

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>*.lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>en-US.lang</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>mi18n lang dosyası</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:

mi18n Mozilla and Java properties files, Java properties, updating-target-files, Java özellikleri dosyasını biçimlendirir, Çeviri dosyası temizle

1.10.8 GWT özellikleri

Native GWT format for translations.

GWT properties are usually used as monolingual translations.

![Typical Weblate Component configuration](table)

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>src/app/Bundle_*.properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>src/app/Bundle.properties</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>GWT Properties</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:

GWT localization guide, GWT Internationalization Tutorial, Mozilla and Java properties files, updating-target-files, Java özellikleri dosyasını biçimlendirir, Çeviri dosyası temizle
1.10.9 INI translations

4.1 sürümünde geldi.

INI file format for translations.

INI translations are usually used as monolingual translations.

Typical Weblate **Component configuration**

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>language/* .ini</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>language/en .ini</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>Ini File</td>
</tr>
</tbody>
</table>

**Not:** Weblate only extracts keys from sections within an INI file. In case your INI file lacks sections, you might want to use Joomla translations or Java properties instead.

Ayrıca bakınız:
INI Files, Java properties, Joomla translations, Inno Setup INI çevirileri

1.10.10 Inno Setup INI çevirileri

4.1 sürümünde geldi.

Çeviriler için Inno Setup INI dosya biçimi.

Inno Setup INI çevirileri genellikle tek dilli çeviriler olarak kullanılır.

**Not:** The only notable difference to INI translations is in supporting \%n and \%t placeholders for line break and tab.

Typical Weblate **Component configuration**

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>language/*.islu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>language/en.islu</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>Inno Setup INI Dosyası</td>
</tr>
</tbody>
</table>

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

Ayrıca bakınız:
INI Files, Joomla translations, INI translations

1.10.11 Joomla translations

2.12 sürümünde geldi.

Native Joomla format for translations.

Joomla translations are usually used as monolingual translations.
Typical Weblate Component configuration

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>language/*/com_foobar.ini</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>language/en-GB/com_foobar.ini</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>Joomla Language File</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:
Specification of Joomla language files, Mozilla and Java properties files, INI translations, Inno Setup INI çevirileri

1.10.12 Qt Linguist .ts

Translation format used in Qt based applications.
Qt Linguist files are used as both bilingual and monolingual translations.

Typical Weblate Component configuration when using as bilingual

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>i18n/app.*.ts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>Empty</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>i18n/app.de.ts</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>Qt Linguist Translation File</td>
</tr>
</tbody>
</table>

Typical Weblate Component configuration when using as monolingual

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>i18n/app.*.ts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>i18n/app.en.ts</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>i18n/app.en.ts</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>Qt Linguist Translation File</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:
Qt Linguist manual, Qt .ts, Bilingual and monolingual formats

1.10.13 Android string resources

Android specific file format for translating applications.
Android string resources are monolingual, the Tek dilli taban dil dosyası is stored in a different location from the others res/values/strings.xml.

Typical Weblate Component configuration

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>res/values-*/strings.xml</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>res/values/strings.xml</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>Android String Resource</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:
Android string resources documentation, Android string resources

Not: Android string-array structures are not currently supported. To work around this, you can break your string arrays apart:
The string-array that points to the string elements should be stored in a different file, and not be made available for translation.

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

### 1.10.14 Apple iOS strings

Apple specific file format for translating applications, used for both iOS and iPhone/iPad application translations. Apple iOS strings are usually used as bilingual translations.

**Typical Weblate Component configuration**

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>Resources/*.lproj/Localizable.strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>Resources/en.lproj/Localizable.strings or Resources/Base.lproj/Localizable.strings</td>
</tr>
<tr>
<td>Yeni çeviri için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>iOS Strings (UTF-8)</td>
</tr>
</tbody>
</table>

**Ayrıca bakınız:**

Stringsdict biçimi, Apple “strings files” documentation, Mac OSX strings

### 1.10.15 PHP dizgileri

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

**Typical Weblate Component configuration**

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>lang/*texts.php</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>lang/en/texts.php</td>
</tr>
<tr>
<td>Yeni çeviri için şablon</td>
<td>lang/en/texts.php</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>PHP strings</td>
</tr>
</tbody>
</table>

1.10. Supported file formats
Laravel PHP dizgileri

4.1 sürümünde değişti.
The Laravel PHP localization files are supported as well with plurals:

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

Ayrıca bkz:
PHP, Localization in Laravel

1.10.16 JSON files

2.0 sürümünde geldi.
2.16 sürümünde değişti: Since Weblate 2.16 and with translate-toolkit at-least 2.2.4, nested structure JSON files are supported as well.
4.3 sürümünde değişti: The structure of JSON file is properly preserved even for complex situations which were broken in prior releases.

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

Weblate currently supports several variants of JSON translations:

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

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

Example file:

```json
{
    "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:

```json
{
    "weblate": {
        "hello": "Ahoj světe!\n",
        "orangutan": "",
        "try": "",
        "thanks": ""
    }
}
```
İpucu: The JSON file and JSON nested structure file can both handle same type of files. Both preserve existing JSON structure when translating. The only difference between them is when adding new strings using Weblate. The nested structure format parses the newly added key and inserts the new string into the matching structure. For example app.name key is inserted as:

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

Typical Weblate Component configuration

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>langs/translation-*.json</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>langs/translation-en.json</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>JSON nested structure file</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:

JSON, updating-target-files, JSON çıktısi özelleştir, Çeviri dosyasını temizle,

1.1.0.17 JSON i18next files

2.17 sürümünde değişti: 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.

**Not:** Weblate supports the i18next JSON v3 format. The v2 and v1 variants are mostly compatible, with exception of how plurals are handled.

Example file:

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

Typical Weblate Component configuration

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>langs/*/json</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>langs/en.json</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>i18next JSON file</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:

JSON, i18next JSON Format, updating-target-files, JSON çıktısi özelleştir, Çeviri dosyasını temizle

1.10. Supported file formats
1.10.18 go-i18n JSON files

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

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

<table>
<thead>
<tr>
<th>Typical Weblate Component configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dosya maskesi</td>
</tr>
<tr>
<td>Tek dilli taban dil dosyası</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
</tr>
<tr>
<td>Dosya biçimi</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:

JSON, go-i18n, updating-target-files, JSON çıktıını özelleştir, Çeviri dosyaslarını temizle.

1.10.19 ARB File

4.1 sürümünde geldi.

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

<table>
<thead>
<tr>
<th>Typical Weblate Component configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dosya maskesi</td>
</tr>
<tr>
<td>Tek dilli taban dil dosyası</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
</tr>
<tr>
<td>Dosya biçimi</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:


1.10.20 WebExtension JSON

2.16 sürümünde geldi: 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.

**Not:** 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:

```json
{
    "hello": {
        "message": "Ahoj světe!\n",
        "description": "Description",
        "placeholders": {
            "url": {
```
Typical Weblate Component configuration

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>_locales/*/messages.json</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>_locales/en/messages.json</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>WebExtension JSON file</td>
</tr>
</tbody>
</table>

Daha fazla bakınız:

JSON, Google chrome.i18n, Mozilla Extensions Internationalization

1.10.21 .XML resource files

2.3 sürümünde geldi.

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

Typical Weblate Component configuration

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>Resources/Language.*.resx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>Resources/Language.resx</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>.NET kaynak dosyası</td>
</tr>
</tbody>
</table>

Daha fazla bakınız:

.NET Resource files (.resx), updating-target-files, Çeviri dosyası temizle

1.10. Supported file formats

75
1.10.22 CSV files

2.4 sürümünde geldi.

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

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

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

Example file:

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

<table>
<thead>
<tr>
<th>Typical Weblate Component configuration for bilingual CSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dosya masksi</td>
</tr>
<tr>
<td>Tek dilli taban dil dosyası</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
</tr>
<tr>
<td>Dosya biçimi</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Typical Weblate Component configuration for monolingual CSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dosya masksi</td>
</tr>
<tr>
<td>Tek dilli taban dil dosyası</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
</tr>
<tr>
<td>Dosya biçimi</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:
CSV

1.10.23 YAML files

2.9 sürümünde geldi.

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": ""
```
The Weblate Manual, Yayın 4.9

Typical Weblate Component configuration

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>translations/messages.*.yml</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>translations/messages.en.yml</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>Ruby YAML file</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:

YAML, Ruby YAML files

1.10.24 Ruby YAML files

2.9 sürümünde geldi.

Ruby i18n YAML files with language as root node.

Example Ruby i18n YAML file:

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

Typical Weblate Component configuration

<table>
<thead>
<tr>
<th>Dosya maskesi</th>
<th>local/*.dtd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tek dilli taban dil dosyası</td>
<td>locale/en.dtd</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
<td>Empty</td>
</tr>
<tr>
<td>Dosya biçimi</td>
<td>DTD file</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:

YAML, YAML files

1.10.25 DTD files

2.18 sürümünde geldi.

Example DTD file:

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

Ayrıca bakınız:

Mozilla DTD format
1.10.26 Flat XML files

3.9 sürümünde geldi.

Example of a flat XML file:

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

| Typical Weblate Component configuration |
|-----------------|----------------|
| Dosya maskesi   | locale/*.xml   |
| Tek dilli taban dil dosyası | locale/en.xml |
| Yeni çeviriler için şablon     | Empty          |
| Dosya biçimi     | Flat XML file  |

Ayrıca bakınız:
Flat XML

1.10.27 Windows RC files

4.1 sürümünde değişti: Support for Windows RC files has been rewritten.

| Not: Support for this format is currently in beta, feedback from testing is welcome. |

Example Windows RC file:

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

| Typical Weblate Component configuration |
|-----------------|----------------|
| Dosya maskesi   | lang/*.rc      |
| Tek dilli taban dil dosyası | lang/en-US.rc |
| Yeni çeviriler için şablon     | lang/en-US.rc |
| Dosya biçimi     | RC file       |

Ayrıca bakınız:
Windows RC files
1.10.28 Uygulama mağazası üst veri dosyaları

3.5 sürümünde geldi.

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.

<table>
<thead>
<tr>
<th>Typical Weblate Component configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dosya maskesi</td>
</tr>
<tr>
<td>Tek dilli taban dil dosyası</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
</tr>
<tr>
<td>Dosya biçimleri</td>
</tr>
</tbody>
</table>

İpucu: In case you don’t want to translate certain strings (for example changelogs), mark them read-only (see Customizing behavior using flags). This can be automated by the Toplu düzenleme.

1.10.29 Subtitle files

3.7 sürümünde geldi.

Weblate çeşitli altyazı dosyalarını çevirebilir:

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

Ayrıca bakınız: Subtitles

1.10.30 Excel Open XML

3.2 sürümünde geldi.

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. Supported file formats
1.10.31 HTML files

4.1 sürümünde geldi.

Not: 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.

Ayrıca bakınız:

HTML

1.10.32 Metin dosyaları

4.6 sürümünde geldi.

Not: Support for this format is currently in beta, feedback from testing is welcome.

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

Bu biçimin üç çeşidi vardır:

• Düz metin dosyası
• DokuWiki metin dosyası
• MediaWiki metin dosyası

Ayrıca bakınız:

Simple Text Documents

1.10.33 OpenDocument Format

4.1 sürümünde geldi.

Not: 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.

Ayrıca bakınız:

OpenDocument Format

1.10.34 IDML Format

4.1 sürümünde geldi.

Not: Support for this format is currently in beta, feedback from testing is welcome.

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

4.5 sürümünde geldi.

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

<table>
<thead>
<tr>
<th>Typical Weblate Component configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dosya maskesi</td>
</tr>
<tr>
<td>Tek dilli taban dil dosyası</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
</tr>
<tr>
<td>Dosya biçimleri</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:
TBX on Wikipedia, TBX, Sözüklük

1.10.36 Stringsdict biçim

4.8 sürümünde geldi.

**Not:** Support for this format is currently in beta, feedback from testing is welcome.

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

<table>
<thead>
<tr>
<th>Typical Weblate Component configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dosya maskesi</td>
</tr>
<tr>
<td>Tek dilli taban dil dosyası</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
</tr>
<tr>
<td>Dosya biçimleri</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:
Apple iOS strings, Stringsdict File Format,

1.10.37 Fluent biçimi

4.8 sürümünde geldi.

**Not:** Support for this format is currently in beta, feedback from testing is welcome.

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

<table>
<thead>
<tr>
<th>Typical Weblate Component configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dosya maskesi</td>
</tr>
<tr>
<td>Tek dilli taban dil dosyası</td>
</tr>
<tr>
<td>Yeni çeviriler için şablon</td>
</tr>
<tr>
<td>Dosya biçimleri</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:
Project Fluent website

1.10. Supported file formats
1.10.38 Diğer biçimleri destekleme

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

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

Ayrıca bakınız:
Translation Related File Formats

1.11 Sürüm denetimi tümleşimi

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

1.11.1 Accessing repositories

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

Accessing repositories from Hosted Weblate

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

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

Once the weblate user is added, you can configure Kaynak kod deposu and Depo yollama URL'si using the SSH protocol (for example git@github.com:WeblateOrg/weblate.git).

SSH repositories

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

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

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

In case adjustment is needed, do so from the Weblate admin interface:
The Weblate public key is visible to all users browsing the About page.

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

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

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

**Verifying SSH host keys**

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

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

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

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

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

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

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

*Ayrıca bakınız:*

*Accessing repositories from Hosted Weblate*
Weblate internal URLs

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

**Uyarı:** Removing main component also removes linked components.

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

Reasons to use this:

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

**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.

**Not:** If your username or password contains special characters, those have to be URL encoded, for example https://user%40example.com:%24password%23@bitbucket.org/….

**Using proxy**

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

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

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

**Not:** 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.

**Ayrıca bakınız:**

The cURL manpage, Git config documentation
1.11.2 Git

İpucu: Weblate needs Git 2.12 or newer.

Ayrıca bakınız:
See Accessing repositories for info on how to access different kinds of repositories.

Git (zorlama push ile)

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.

Uyarı: 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
```

Uyarı: 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 sürümünde geldi.

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

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

Ayrıca bakınız:
Pushing changes from Weblate
Pushing changes to GitHub as pull requests

If not wanting to push translations to a GitHub repository, they can be sent as either one or many pull requests instead. You need to configure API credentials to make this work.

Ayrıca bakınız:

GITHUB_USERNAME, GITHUB_TOKEN, GITHUB_CREDENTIALS

1.11.4 GitLab

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

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

Ayrıca bakınız:

Pushing changes from Weblate

Pushing changes to GitLab as merge requests

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

You need to configure API credentials to make this work.

Ayrıca bakınız:

GITLAB_USERNAME, GITLAB_TOKEN, GITLAB_CREDENTIALS

1.11.5 Pagure

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

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

Ayrıca bakınız:

Pushing changes from Weblate

Değişiklikleri birleştirme istekleri olarak Pagure’ye gönderme

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

You need to configure API credentials to make this work.

Ayrıca bakınız:

PAGURE_USERNAME, PAGURE_TOKEN, PAGURE_CREDENTIALS
1.11.6 Gerrit

2.2 sürümünde geldi.

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

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

1.11.7 Mercurial

2.1 sürümünde geldi.

Mercurial is another VCS you can use directly in Weblate.

**Not:** It should work with any Mercurial version, but there are sometimes incompatible changes to the command-line interface which breaks Weblate integration.

**Ayrıca bakınız:**

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

1.11.8 Subversion

2.8 sürümünde geldi.

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.

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

2.19 sürümünde değişti: Before this, only repositories using the standard layout were supported.

**Subversion credentials**

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

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

**Ayrıca bakınız:**

`DATA_DIR`
1.11.9 Local files

1.11.10 Git

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

3.8 sürümünde geldi.

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

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

1.12 Weblate’s REST API

2.6 sürümünde geldi: The REST API is available since Weblate 2.6.

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

1.12.1 Authentication and generic parameters

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

ANY /

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

Query Parameters

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

Request Headers

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

Response Headers

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

Response JSON Object

- detail (string) – verbose description of the result (for HTTP status codes other than 200 OK)
- count (int) – total item count for object lists
• **next**(string) – next page URL for object lists
• **previous**(string) – previous page URL for object lists
• **results**(array) – results for object lists
• **url**(string) – URL to access this resource using API
• **web_url**(string) – URL to access this resource using web browser

**Status Codes**

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

**Authentication examples**

**Example request:**

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

**Example response:**

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

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

**CURL example:**

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

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

Form request example:

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

operation = pull
```

JSON request example:

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

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

CURL example:

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

CURL JSON example:

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

API rate limiting

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

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

The status of rate limiting is reported in following headers:

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-RateLimit-Limit</td>
<td>Rate limiting limit of requests to perform</td>
</tr>
<tr>
<td>X-RateLimit-Remaining</td>
<td>Remaining limit of requests</td>
</tr>
<tr>
<td>X-RateLimit-Reset</td>
<td>Number of seconds until ratelimit window resets</td>
</tr>
</tbody>
</table>

4.1 sürümünde değişti: Added ratelimiting status headers.

Ayrıca bakınız:

Rate limiting, Rate limiting

1.12. Weblate’s REST API
1.12.2 API Entry Point

GET /api/

The API root entry point.

Example request:

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

Example response:

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

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

1.12.3 Kullanıcılar

4.0 sürümünde geldi.

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.

Ayrıca bakınız:

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

POST /api/users/

Creates a new user.

Parameters

- **username** (string) – Kullanıcı adı
- **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:

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

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

Changes the user parameters.

Parameters

• **username** *(string)* – User’s username

Response JSON Object

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

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

Changes the user parameters.

Parameters

• **username** *(string)* – User’s username

Response JSON Object

• **username** *(string)* – username of a user
• **full_name** *(string)* – full name of a user
• **email** *(string)* – email of a user
• **is_superuser** *(boolean)* – whether the user is a super user
• **is_active** *(boolean)* – whether the user is active
• **date_joined** *(string)* – date the user is created
DELETE /api/users/(str: username)/
Deletes all user information and marks the user inactive.

Parameters
• username(string) – User’s username

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

Parameters
• username(string) – User’s username

Form Parameters
• string group_id – The unique group ID

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

Parameters
• username(string) – User’s username

Response JSON Object
• translated(int) – Kullanıcı tarafından yapılan çeviri sayısı
• suggested(int) – Kullanıcı tarafından yapılan öneri sayısı
• uploaded(int) – Kullanıcı tarafından yapılan yükleme sayısı
• commented(int) – Kullanıcı tarafından yapılan yorum sayısı
• languages(int) – Kullanıcının çevirebileceği dili sayısı

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) – Kayıtlı bildirim kimliği

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)` – Kayıtlı bildirim kimliği

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)` – Kayıtlı bildirim kimliği

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 Gruplar

4.0 sürümünde geldi.

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.

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

POST /api/groups/
Creates a new group.

Parameters

• `name (string)` – Grup adı
• `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
The Weblate Manual, Yayın 4.9

- **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:

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

**PUT /api/groups/(int: id)/**
Changes the group parameters.

**Parameters**

- **id**(int) – Group’s ID

**Response JSON Object**

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

**PATCH /api/groups/(int: id)/**
Changes the group parameters.

**Parameters**

- **id**(int) – Group’s ID

**Response JSON Object**

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

**DELETE /api/groups/(int: id)/**
Deletes the group.
Parameters

- **id**(int) – Group’s ID

**POST /api/groups/(int: id)/roles/**

Associate roles with a group.

Parameters

- **id**(int) – Group’s ID

Form Parameters

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

**POST /api/groups/(int: id)/components/**

Associate components with a group.

Parameters

- **id**(int) – Group’s ID

Form Parameters

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

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

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/**

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/**

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 Roller

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

Ayrıca bakınız:
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:

```json
{
    "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 Diller

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

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

**POST /api/languages/**
Creates a new language.

**Parameters**
• **code** (*string*) – Dil adı
• **name** (*string*) – Dil adı
• **direction** (*string*) – Metin yönü
• **plural** (*object*) – Language plural formula and number

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

**Parameters**
• **language** (*string*) – Dil kodu

**Response JSON Object**
• **code** (*string*) – Dil kodu
• **direction** (*string*) – Metin yönü
• **plural** (*object*) – Object of language plural information
• **aliases** (*array*) – Array of aliases for language

**Example JSON data:**

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

(sonraki sayfaya devam)
PUT /api/languages/(string: language)/
Changes the language parameters.

Parameters

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

Request JSON Object

- **name**(string) – Dil adı
- **direction**(string) – Metin yönü
- **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) – Dil adı
- **direction**(string) – Metin yönü
- **plural**(object) – Language plural details

DELETE /api/languages/(string: language)/
Dili siler.

Parameters

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

GET /api/languages/(string: language)/statistics/
Returns statistics for a language.

Parameters

- **language**(string) – Dil kodu

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)` – belirsiz (düzenele için işaretlenen) dizgilerin sayısı
• `fuzzy_percent (int)` – percentage of fuzzy (marked for edit) strings
• `failing (int)` – number of failing strings
• `failing` – percentage of failing strings

1.12.7 Projeler

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

Ayrıca bakınız:

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

POST /api/projects/
3.9 sürümünde geldi.
Creates a new project.

Parameters
• `name (string)` – Proje adı
• `slug (string)` – Project slug
• `web (string)` – Proje web sitesi

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

Parameters
• `project (string)` – Proje URL’si kısaltması

Response JSON Object
• `name (string)` – proje adı
• `slug (string)` – project slug
• `web (string)` – project website
• `components_list_url (string)` – URL to components list; see `GET /api/projects/(string:project)/components/`
• `repository_url (string)` – URL to repository status; see `GET /api/projects/(string:project)/repository/`
• `changes_list_url (string)` – URL to changes list; see `GET /api/projects/(string:project)/changes/`
• `translation_review (boolean)` – Gözden geçirmeleri etkinleştir
• **source_review** (boolean) – Kaynak gözden geçirmelerini etkinleştir
• **set_language_team** (boolean) – "Language-Team" (Çeviri-Takımı) başlığını ayarla
• **enable_hooks** (boolean) – Kancaları etkinleştir
• **instructions** (string) – Çeviri talimatları
• **language_aliases** (string) – Dil kod adları

Example JSON data:

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

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

4.3 sürümünde geldi.

Edit a project by a PATCH request.

**Parameters**

• **project** (string) – Proje URL’si kısaltması
• **component** (string) – Bileşen URL’si kısaltması

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

4.3 sürümünde geldi.

Edit a project by a PUT request.

**Parameters**

• **project** (string) – Proje URL’si kısaltması

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

3.9 sürümünde geldi.

Deletes a project.

**Parameters**

• **project** (string) – Proje URL’si kısaltması

**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) – Proje URL’si kısaltması

**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) – Proje URL’si kısaltması
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:

```json
{
  "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) – Proje URL’si kısaltması

Request JSON Object

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

Response JSON Object

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

CURL example:

```bash
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.
• **project (string)** – Proje URL’si kısaltması

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 sürümünden geldi.

4.3 sürümünde değişti: The **zipfile** and **docfile** parameters are now accepted for VCS-less components, see **Local files**.

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

Creates translation components in the given project.

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

**Not:** Most of the component creation happens in the background. Check the **task_url** attribute of created component and follow the progress there.

**Parameters**

• **project (string)** – Proje URL’si kısaltması

**Form Parameters**

• **file zipfile** – ZIP file to upload into Weblate for translations initialization

• **file docfile** – Çevrilecek belge

• **boolean disable_autoshare** – Disables automatic repository sharing via **Weblate internal URLs**.

Response JSON Object

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

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

**CURL form request example:**

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

**CURL JSON request example:**

```json
curl
   --data-binary '{
   "branch": "main",
   "file_format": "po",
   "filemask": "po/*.po",
   "git_export": "",
   "repo_url": "",
   "is_repo": false,
   "branch": "main",
   "ref": "main",
   "project": "hello",
   "slug": "weblate",
   "url": "http://example.com/api/projects/hello/components/"
}'
```

(sonraki sayfaya devam)
The Weblate Manual, Yayın 4.9

```
"license": "",
"license_url": "",
"name": "Weblate",
"slug": "weblate",
"repo": "file:///home/nijel/work/weblate-hello",
"template": "",
"new_base": "",
"vcs": "git"
```

JSON request example:

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

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

JSON response example:

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

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

1.12. Weblate’s REST API 105

(sonraki sayfaya devam)
GET /api/projects/(string: project)/languages/
Returns paginated statistics for all languages within a project.

3.8 sürümünde geldi.

Parameters

- **project (string)** – Proje URL’si kısaltması

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 sürümünde geldi.

Parameters

- **project (string)** – Proje URL’si kısaltması

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 Bileşenler

GET /api/components/
   Returns a list of translation components.
   Ayrıca bakınız:
   Component object attributes are documented at GET /api/components/(string:project)/
   (string:component)/.

GET /api/components/(string: project)/
   Returns information about translation component.
   Parameters
   • project (string) – Proje URL’si kısaltması
   • component (string) – Bileşen URL’si kısaltması

Response JSON Object
   • project (object) – the translation project; see GET /api/projects/
     (string:project)/
   • name (string) – Bileşen adı
   • slug (string) – Component slug
   • vcs (string) – Sürüm denetleme sistemi
   • repo (string) – Kaynak kod deposu
   • git_export (string) – Dışa aktarılmış depo URL’si
   • branch (string) – Depo dalı
   • push_branch (string) – Yollama dalı
   • filemask (string) – File mask
   • template (string) – Tek dilli taban dil dosyası
   • edit_template (string) – Taban dosyası düzenele
   • intermediate (string) – Ara dil dosyası
   • new_base (string) – Yeni çeviriler için şablon
   • file_format (string) – Dosya biçimi
   • license (string) – Çeviri lisansı
   • agreement (string) – Katılımcı sözleşmesi
   • new_lang (string) – Yeni çeviri ekleniyor
   • language_code_style (string) – Dil kodu stili
   • source_language (object) – source language object; see GET /api/
     languages/(string:language)/
   • push (string) – Depo yollama URL’si
   • check_flags (string) – Çeviri işaretleri
   • priority (string) – Öncelik
   • enforced_checks (string) – Zorunlu denetimler
   • restricted (string) – Restricted access
   • repoweb (string) – Depo tarayıcısi
   • report_source_bugs (string) – Kaynak dizgi hata bildirme adresi
• **merge_style**(string) – **Birleştirme stili**
• **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) – **Çeviri yaymaya izin ver**
• **enable_suggestions**(string) – **Önerileri etkinleştir**
• **suggestion_voting**(string) – **Öneri oylaması**
• **suggestion_autoaccept**(string) – **Önerileri kendiliğinden kabul et**
• **push_on_commit**(string) – **İşlemeye yolla**
• **commit_pending_age**(string) – **İşlenecek değişikliklerin yaş**
• **auto_lock_error**(string) – **Hata durumunda kilitle**
• **language_regex**(string) – **Dil süzgeci**
• **variant_regex**(string) – **Çeşitlerin düzenli ifadesi**
• **repository_url**(string) – **URL to repository status; see GET /api/components/(string:project)/(string:component)/repository/**
• **translations_url**(string) – **URL to translations list; see GET /api/components/(string:project)/(string:component)/translations/**
• **lock_url**(string) – **URL to lock status; see GET /api/components/(string:project)/(string:component)/lock/**
• **changes_list_url**(string) – **URL to changes list; see GET /api/components/(string:project)/(string:component)/changes/**
• **task_url**(string) – **URL to a background task (if any); see GET /api/tasks/(str:uuid)/**

**Example JSON data:**

```json
{
    "branch": "main",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "Weblate",
    "slug": "weblate",
    "project": {
        "name": "Hello",
        "slug": "hello",
        "source_language": {
            "code": "en",
            "direction": "ltr",
            "name": "English",
            "url": "http://example.com/api/languages/en/",
            "web_url": "http://example.com/languages/en/"
        },
        "url": "http://example.com/api/projects/hello/",
        "web": "https://weblate.org/"
    }
}
```

(sonraki sayfaya devam)
The Weblate Manual, Yayın 4.9

PATCH /api/components/(string: project)/
string: component/ Edit a component by a PATCH request.

Parameters

- **project (string)** – Proje URL’si kısaltması
- **component (string)** – Bileşen URL’si kısaltması
- **source_language (string)** – Project source language code (optional)

Request JSON Object

- **name (string)** – name of component
- **slug (string)** – slug of component
- **repo (string)** – VCS repository URL

CURL example:

```
curl \
  --data-binary '{"name": "new name"}' \
  -H "Content-Type: application/json" \
  -H "Authorization: Token TOKEN" \ 
  PATCH http://example.com/api/projects/hello/components/
```

JSON request example:

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

{
   "name": "new name"
}
```

JSON response example:

```
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
```

1.12. Weblate’s REST API
PUT /api/components/ (string: project) /
string: component/ Edit a component by a PUT request.

Parameters

• project (string) – Proje URL'si kısaltması
• component (string) – Bileşen URL'si kısaltması

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 sürümünde geldi.
Deletes a component.

Parameters
- `project` *(string)* – Proje URL’si kısaltması
- `component` *(string)* – Bileşen URL’si kısaltması

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)* – Proje URL’si kısaltması
- `component` *(string)* – Bileşen URL’si kısaltması

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)* – Proje URL’si kısaltması
- `component` *(string)* – Bileşen URL’si kısaltması

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)* – Proje URL’si kısaltması
- `component` *(string)* – Bileşen URL’si kısaltması

Response JSON Object
- `locked` *(boolean)* – whether component is locked for updates

Example JSON data:

```json
{
  "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)* – Proje URL’si kısaltması
- `component` *(string)* – Bileşen URL’si kısaltması

Request JSON Object
- `lock` – Boolean whether to lock or not.

CURL example:
curl \
  -d "lock=true" \
  -H "Authorization: Token TOKEN" \
  http://example.com/api/components/hello/weblate/repository/

JSON request example:

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

{"lock": true}

JSON response example:

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

{"locked":true}

GET /api/components/(string: project)/
string: component/repository/ Returns information about VCS repository status.
The response is same as for GET /api/projects/(string:project)/repository/.

Parameters

- project (string) – Proje URL’si kısaltması
- component (string) – Bileşen URL’si kısaltması

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) – Proje URL’si kısaltması
- component (string) – Bileşen URL’si kısaltması

Request JSON Object

- operation (string) – Operation to perform: one of push, pull, commit, re- set, cleanup
Response JSON Object

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

**CURL example:**

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

**JSON request example:**

```json
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
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) – Proje URL’si kısaltması
- **component** (string) – Bileşen URL’si kısaltması

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

**Parameters**

- **project** (string) – Proje URL’si kısaltması
- **component** (string) – Bileşen URL’si kısaltması

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

**Parameters**

- **project** (string) – Proje URL’si kısaltması
- **component** (string) – Bileşen URL’si kısaltması

**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*) – Proje URL’si kısaltması

• **component** (*string*) – Bileşen URL’si kısaltması

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 \n   -d "language_code=cs" \n   -H "Authorization: Token TOKEN" \n   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,
```

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

2.7 sürümüne geldi.

Parameters
- **project (string)** – Proje URL’si kısaltması
- **component (string)** – Bileşen URL’si kısaltması

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

GET /api/components/(string: project)/
string: component/links/ Bir bileşenle bağlantılı projeleri döndürür.

4.5 sürümüne geldi.

Parameters
- **project (string)** – Proje URL’si kısaltması
- **component (string)** – Bileşen URL’si kısaltması

Response JSON Object
- **projects (array)** – ilişkili projeler; bkz GET /api/projects/

POST /api/components/(string: project)/
string: component/links/ Projeyi bir bileşenle ilişkilendir.

4.5 sürümüne geldi.

Parameters
- **project (string)** – Proje URL’si kısaltması
- **component (string)** – Bileşen URL’si kısaltması

Form Parameters
- **string project_slug** – Project slug

DELETE /api/components/(string: project)/
string: component/links/string: project_slug/ Bir projenin bir bileşenle ilişkisini kaldırır.

4.5 sürümüne geldi.

Parameters
- **project (string)** – Proje URL’si kısaltması
The Weblate Manual, Yayım 4.9

- **component**(string) – Bileşen URL'si kısaltması
- **project_slug**(string) – Kaldırılacak projenin kısaltması

### 1.12.9 Çeviriler

**GET /api/translations/**

Returns a list of translations.

Ayrıca bakınız:

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) – Proje URL’si kısaltması
- **component**(string) – Bileşen URL’si kısaltması
- **language**(string) – Translation language code

#### Response JSON Object

- **component**(object) – component object; see **GET /api/components/(string:project)/(string:component)/**
- **failing_checks**(int) – kusur denetimleri olan dizgi sayısı
- **failing_checks_percent**(float) – kusur denetimleri olan dizgi yüzdesi
- **failing_checks_words**(int) – kusur denetimleri olan sözcük sayısı
- **filename**(string) – translation filename
- **fuzzy**(int) – belirsiz (düzenleme içi işaretlenen) dizgilerin sayısı
- **fuzzy_percent**(float) – percentage of fuzzy (marked for edit) strings
- **fuzzy_words**(int) – number of words in fuzzy (marked for edit) strings
- **have_comment**(int) – number of strings with comment
- **have_suggestion**(int) – number of strings with suggestion
- **is_template**(boolean) – çevirinin tek dilli bir taban olup olmadığı
- **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:

```json
{
  "component": {
    "branch": "main",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "Weblate",
    "new_base": "",
    "project": {
      "name": "Hello",
      "slug": "hello",
      "source_language": {
        "code": "en",
        "direction": "ltr",
        "name": "English",
        "url": "http://example.com/api/languages/en/",
        "web_url": "http://example.com/languages/en/"
      },
      "url": "http://example.com/api/projects/hello/",
      "web": "https://weblate.org/",
      "web_url": "http://example.com/projects/hello/"
    },
    "repo": "file:///home/nijel/work/weblate-hello",
    "slug": "weblate",
    "template": "",
    "url": "http://example.com/api/components/hello/weblate/",
    "vcs": "git",
    "web_url": "http://example.com/projects/hello/weblate/"
  },
  "failing_checks": 3,
  "failing_checks_percent": 75.0,
  "failing_checks_words": 11,
  "filename": "po/cs.po",
  "fuzzy": 0,
  "fuzzy_percent": 0.0,
  "fuzzy_words": 0,
  "have_comment": 0,
  "have_suggestion": 0,
  "is_template": false,
  "language": {
```

(sonraki sayfaya devam)
DELETE /api/translations/(string: project) /
string: component/string: language/ 3.9 sürümünde geldi.

Deletes a translation.

Parameters

• project (string) – Proje URL’si kısaltması
• component (string) – Bileşen URL’si kısaltması
• 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) – Proje URL’si kısaltması
• component (string) – Bileşen URL’si kısaltması
• 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) – Proje URL’si kısaltması
• component (string) – Bileşen URL’si kısaltması
• language (string) – Translation language code
• q (string) – Search query string Searching (optional)

Response JSON Object

• results (array) – array of component objects; see GET /api/units/(int:id)/
POST /api/translations/(string: project)/
string: component/string: language/units/ Add new monolingual unit.

Parameters

- **project**(string) – Proje URL’si kısaltması
- **component**(string) – Bileşen URL’si kısaltması
- **language**(string) – Translation language code

Request JSON Object

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

Ayrıca bakınız:

*Dizgileri yönet*, adding-new-strings

POST /api/translations/(string: project)/

Parameters

- **project**(string) – Proje URL’si kısaltması
- **component**(string) – Bileşen URL’si kısaltması
- **language**(string) – Translation language code

Request JSON Object

- **mode**(string) – Otomatik çeviri kipi
- **filter_type**(string) – Automatic translation filter type
- **auto_source**(string) – Otomatik çeviri kaynağı - mt veya others
- **component**(string) – Projenin ek bileşenlere erişebilmesi için yayınlanan çeviri bel- leğine katkıyı açın.
- **engines**(array) – Makine çevirisini motorları
- **threshold**(string) – Skor eşiği

GET /api/translations/(string: project)/
string: component/string: language/file/ Download current translation file as it is stored in the VCS (without the format parameter) or converted to another format (see *Downloading translations*).

**Not:** 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, xlfiff, xlfiff1, tbx, csv, xlsx, json, aresource, strings

Parameters

- **project**(string) – Proje URL’si kısaltması
- **component**(string) – Bileşen URL’si kısaltması
- **language**(string) – Translation language code

POST /api/translations/(string: project)/
Parameters

- **project (string)** – Proje URL'si kısaltması
- **component (string)** – Bileşen URL'si kısaltması
- **language (string)** – Translation language code

Form Parameters

- **string conflict** – How to deal with conflicts (ignore, replace-translated or replace-approved)
- **file file** – Uploaded file
- **string email** – Hazırlayanın e-postası
- **string author** – Hazırlayanın adı
- **string method** – Upload method (translate, approve, suggest, fuzzy, replace, source, add), see [Import methods](#)
- **string fuzzy** – Fuzzy (marked for edit) strings processing (empty, process, approve)

CURL example:

```bash
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)** – Proje URL'si kısaltması
- **component (string)** – Bileşen URL'si kısaltması
- **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)** – Proje URL'si kısaltması
- **component (string)** – Bileşen URL'si kısaltması
- **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)/

2.7 sürümünde geldi.

Parameters

- **project**(string) – Proje URL’si kısaltması
- **component**(string) – Bileşen URL’si kısaltması
- **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) – belirsiz (düzeneleme için işaretlenen) dizgilerin sayısı
- **fuzzy_percent**(float) – percentage of fuzzy (marked for edit) strings
- **total_words**(int) – total number of words
- **translated_words**(int) – number of translated words
- **last_author**(string) – name of last author
- **last_change**(timestamp) – date of last change
- **name**(string) – language name
- **total**(int) – total number of strings
- **translated**(int) – number of translated strings
- **translated_percent**(float) – percentage of translated strings
- **url**(string) – URL to access the translation (engagement URL)
- **url_translate**(string) – URL to access the translation (real translation URL)

1.12.10 Units

A unit is a single piece of a translation which pairs a source string with a corresponding translated string and also contains some related metadata. The term is derived from the Translate Toolkit and XLIFF.

2.10 sürümünde geldi.

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

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

GET /api/units/(int: id)/
4.3 sürümünde değişti: The target and source are now arrays to properly handle plural strings.

Returns information about translation unit.

Parameters

- **id**(int) – Unit ID

Response JSON Object

- **translation**(string) – URL of a related translation object
- **source**(array) – source string
• `previous_source` *(string)* – previous source string used for fuzzy matching
• `target` *(array)* – target string
• `id_hash` *(string)* – unique identifier of the unit
• `content_hash` *(string)* – unique identifier of the source string
• `location` *(string)* – location of the unit in source code
• `context` *(string)* – translation unit context
• `note` *(string)* – translation unit note
• `flags` *(string)* – translation unit flags
• `state` *(int)* – unit state, 0 - not translated, 10 - needs editing, 20 - translated, 30 - approved, 100 - readonly
• `fuzzy` *(boolean)* – birimin belirsiz veya gözden geçirilmek üzere işaretlenmiş olup olmadığını
• `translated` *(boolean)* – birimin çevrilmiş olup olmadığını
• `approved` *(boolean)* – çevirinin onaylanmış olup olmadığını
• `position` *(int)* – unit position in translation file
• `has_suggestion` *(boolean)* – birimin önerileri olup olmadığını
• `has_comment` *(boolean)* – birimin yorumları olup olmadığını
• `has_failing_check` *(boolean)* – birimin kusur denetimleri olup olmadığını
• `num_words` *(int)* – number of source words
• `priority` *(int)* – translation priority; 100 is default
• `id` *(int)* – unit identifier
• `explanation` *(string)* – String explanation, available on source units, see *Additional info on source strings*
• `extra_flags` *(string)* – Additional string flags, available on source units, see *Customizing behavior using flags*
• `web_url` *(string)* – URL where the unit can be edited
• `souce_unit` *(string)* – Source unit link; see *GET /api/units/(int:id)/*

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

4.3 sürümünde geldi.

Çeviri birimi üzerinde kısmını güncellemeye gerçekleştirmiştir.

Parameters

• `id` *(int)* – Unit ID

Request JSON Object

• `state` *(int)* – unit state, 0 - not translated, 10 - needs editing, 20 - translated, 30 - approved (need review workflow enabled, see *Dedicated reviewers*)
• `target` *(array)* – target string
• `explanation` *(string)* – String explanation, available on source units, see *Additional info on source strings*
• `extra_flags` *(string)* – Additional string flags, available on source units, see *Customizing behavior using flags*
PUT /api/units/ (int: id) /
4.3 sürümünde geldi.
Çeviri birimi üzerinde tam güncelleme gerçekleştirir.

Parameters

• id (int) – Unit ID

Request JSON Object

• state (int) – unit state, 0 - not translated, 10 - needs editing, 20 - translated, 30 - approved (need review workflow enabled, see Dedicated reviewers)
• target (array) – target string
• explanation (string) – String explanation, available on source units, see Additional info on source strings
• extra_flags (string) – Additional string flags, available on source units, see Customizing behavior using flags

DELETE /api/units/ (int: id) /
4.3 sürümünde geldi.
Bir çeviri birimini siler.

Parameters

• id (int) – Unit ID

1.12.11 Değişiklikler

2.10 sürümünde geldi.

GET /api/changes/
4.1 sürümünde değişti: Filtering of changes was introduced in the 4.1 release.

Returns a list of translation changes.

Ayrıca bakınız:
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
The Weblate Manual, Yayım 4.9

- **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 Ekran görüntüleri

2.14 sürümünde geldi.

**GET /api/screenshots/**

Returns a list of screenshot string information.

Ayrıca bakınız:

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:

```bash
curl -X POST \n  -F image=@image.png \n  -H "Authorization: Token TOKEN" \n  http://example.com/api/screenshots/1/file/
```

**POST /api/screenshots/(int: id)/units/**

Associate source string with screenshot.
Parameters

- `id` (int) – Screenshot ID

Form Parameters

- `string unit_id` – Unit ID

Response JSON Object

- `name` (string) – name of a screenshot
- `translation` (string) – URL of a related translation object
- `file_url` (string) – URL to download a file; see GET /api/screenshots/(int:id)/file/
- `units` (array) – link to associated source string information; see GET /api/units/(int:id)/

DELETE /api/screenshots/(int:id)/units/

- `int: unit_id` Kaynak dizi ile ekran görüntüsü ilişkilendirmesini kaldır.

Parameters

- `id` (int) – Screenshot ID
- `unit_id` – Kaynak dizi birimi kimliği

POST /api/screenshots/

Yeni bir ekran görüntüsünü oluşturur.

Form Parameters

- `file image` – Uploaded file
- `string name` – Ekran görüntüsü adı
- `string project_slug` – Project slug
- `string component_slug` – Component slug
- `string language_code` – Dil Kodu

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)/

Ekran görüntüsü ile ilgili kısmi bilgileri düzenle.

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)/
Ekran görüntüsü ile ilgili tüm bilgileri düzenle.

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)/
Ekran görüntüsünü sil.

Parameters

• id(int) – Screenshot ID

1.12.13 Eklentiler

4.4.1 sürümünde geldi.

GET /api/addons/
Eklentilerin bir listesini döndürür.

Ayrıca bakınız:
Eklenti nesne nitelikleri GET /api/addons/(int:id)/ adresinde belgelendirilmiştir.

GET /api/addons/(int: id)/
Eklentiyle ilgili bilgileri döndürür.

Parameters

• id(int) – Eklenti kimliği

Response JSON Object

• name(string) – bir eklentinin adı
• component(string) – URL of a related component object
• configuration(object) – İsteğe bağlı eklenti yapılandırması

Ayrıca bakınız:

Eklentiler

POST /api/components/(string: project)/
string: component/addons/ Yeni bir eklenti oluşturur.

Parameters

• project_slug(string) – Project slug
• component_slug(string) – Component slug

Request JSON Object

• name(string) – bir ekleninin adı
• configuration(object) – İsteğe bağlı eklenti yapılandırması
PATCH /api/addons/(int: id)/
Eklenti ile ilgili kısıtlamaları düzenleyin.

**Parameters**
- **id (int)** – Eklenti kimliği

**Response JSON Object**
- **configuration (object)** – İsteğe bağlı eklenti yapılandırması

PUT /api/addons/(int: id)/
Eklenti ile ilgili tüm bilgileri düzenleyin.

**Parameters**
- **id (int)** – Eklenti kimliği

**Response JSON Object**
- **configuration (object)** – İsteğe bağlı eklenti yapılandırması

DELETE /api/addons/(int: id)/
Eklenti silin.

**Parameters**
- **id (int)** – Eklenti kimliği

### 1.12.14 Bileşen listeleri

4.0 sürümünde geldi.

GET /api/component-lists/
Bileşen listelerinin bir listesini döndürür.

**Ayrıca bakınız:**
Bileşen listesi nesne nitelikleri GET /api/component-lists/(str:slug)/ adresinde belgelendi-

GET /api/component-lists/(str: slug)/
Bileşen listesi hakkında bilgi döndürür.

**Parameters**
- **slug (string)** – Bileşen listesi kısaltması

**Response JSON Object**
- **name (string)** – bir bileşen listesinin adı
- **slug (string)** – bir bileşen listesinin kısaltması
- **show_dashboard (boolean)** – bir panelde gösterilip gösterilmeyeceği
- **components (array)** – link to associated components; see GET /api/
  components/(string:project)/(string:component)/
- **auto_assign (array)** – otomatik atama kuralları

PUT /api/component-lists/(str: slug)/
Bileşen listesi parametrelerini değiştirir.

**Parameters**
- **slug (string)** – Bileşen listesi kısaltması

**Request JSON Object**
- **name (string)** – bir bileşen listesinin adı
The Weblate Manual, Yayım 4.9

- **slug**(string) – bir bileşen listesinin kısaltması
- **show_dashboard**(boolean) – bir panelde gösterilip gösterilmeyeceği

**PATCH /api/component-lists/(str: slug)/**
Bileşen listesi parametrelerini değiştirir.

**Parameters**
- **slug**(string) – Bileşen listesi kısaltması

**Request JSON Object**
- **name**(string) – bir bileşen listesinin adı
- **slug**(string) – bir bileşen listesinin kısaltması
- **show_dashboard**(boolean) – bir panelde gösterilip gösterilmeyeceği

**DELETE /api/component-lists/(str: slug)/**
Bileşen listesini siler.

**Parameters**
- **slug**(string) – Bileşen listesi kısaltması

**POST /api/component-lists/(str: slug)/components/**
Bileşeni bir bileşen listesiyle ilişkilendir.

**Parameters**
- **slug**(string) – Bileşen listesi kısaltması

**Form Parameters**
- **string component_id** – Bileşen kimliği

**DELETE /api/component-lists/(str: slug)/components/**
str: component_slug Bileşen listesinden bir bileşenin ilişkisini kaldırır.

**Parameters**
- **slug**(string) – Bileşen listesi kısaltması
- **component_slug**(string) – Component slug

### 1.12.15 Sözlük

4.5 sürümünde değişti: Glossaries are now stored as regular components, translations and strings, please use respective API instead.

### 1.12.16 Görevler

4.4 sürümünde geldi.

**GET /api/tasks/**
Görevlerin listesi şu anda kullanılamıyor.

**GET /api/tasks/(str: uuid)/**
Bir görevle ilgili bilgileri döndürür

**Parameters**
- **uuid**(string) – Görev UUID’si

**Response JSON Object**
- **completed**(boolean) – Görevin tamamlandıp tamamlanmadığı
- **progress**(int) – Yüzde olarak görev ilerlemesi
1.12.17 Metrics

GET /api/metrics/
Returns server metrics.

Response JSON Object
- units (int) – Birim sayısı
- units_translated (int) – Çevrilen birim sayısı
- users (int) – Kullanıcı sayısı
- changes (int) – Değişiklik sayısı
- projects (int) – Proje sayısı
- components" (int) – Bileşen sayısı
- translations" (int) – Çeviri sayısı
- languages" (int) – Kullanılan dil sayısı
- checks" (int) – Tetiklenen kursur denetimleri sayısı
- configuration_errors" (int) – Yapılandırma hataları sayısı
- suggestions" (int) – Bekleyen öneri sayısı
- celery_queues (object) – Lengths of Celery queues, see Background tasks using Celery
- name (string) – Yapılandırılan sunucu adı

1.12.18 Bildirim kancaları

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 sürümünden beri kullanım dışı: 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 sürümünden beri kullanım dışı: 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.

Not: 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.

Ayrıca bakınız:
Automatically receiving changes from GitHub  For instruction on setting up GitHub integration
https://docs.github.com/en/github/extending-github/about-webhooks  Generic information about GitHub Webhooks
ENABLE_HOOKS  For enabling hooks for whole Weblate

POST /hooks/gitlab/
Special hook for handling GitLab notifications and automatically updating matching components.

Ayrıca bakınız:

Automatically receiving changes from GitLab  For instruction on setting up GitLab integration
https://docs.gitlab.com/ee/user/project/integrations/webhooks.html  Generic information about GitLab Webhooks
ENABLE_HOOKS  For enabling hooks for whole Weblate

POST /hooks/bitbucket/
Special hook for handling Bitbucket notifications and automatically updating matching components.

Ayrıca bakınız:

Automatically receiving changes from Bitbucket  For instruction on setting up Bitbucket integration
https://support.atlassian.com/bitbucket-cloud/docs/manage-webhooks/  Generic information about Bitbucket Webhooks
ENABLE_HOOKS  For enabling hooks for whole Weblate

POST /hooks/pagure/
3.3 sürümünde geldi.
Special hook for handling Pagure notifications and automatically updating matching components.

Ayrıca bakınız:

Automatically receiving changes from Pagure  For instruction on setting up Pagure integration
https://docs.pagure.org/pagure/usage/using_webhooks.html  Generic information about Pagure Webhooks
ENABLE_HOOKS  For enabling hooks for whole Weblate

POST /hooks/azure/
3.8 sürümünde geldi.
Special hook for handling Azure Repos notifications and automatically updating matching components.

Ayrıca bakınız:

Automatically receiving changes from Azure Repos  For instruction on setting up Azure integration
ENABLE_HOOKS  For enabling hooks for whole Weblate

POST /hooks/gitea/
3.9 sürümünde geldi.
Special hook for handling Gitea Webhook notifications and automatically updating matching components.

Ayrıca bakınız:

Automatically receiving changes from Gitea Repos  For instruction on setting up Gitea integration
https://docs.gitea.io/en-us/webhooks/_Generic information about Gitea Webhooks

**ENABLE_HOOKS** For enabling hooks for whole Weblate

**POST** /hooks/gitee/

3.9 sürümünde geldi.

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

Ayrıca bakınız:

*Automatically receiving changes from Gitee Repos* For instruction on setting up Gitee integration

https://gitee.com/help/categories/40_Generic information about Gitee Webhooks

**ENABLE_HOOKS** For enabling hooks for whole Weblate

### 1.12.19 Exports

Weblate provides various exports to allow you to further process the data.

**GET** /exports/stats/(string: project)/

**string**: component/

Query Parameters

* format (string) – Output format: either json or csv

2.6 sürümünden beri kullanım dışı: 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:**

```plaintext
GET /exports/stats/weblate/main/ HTTP/1.1
Host: example.com
Accept: application/json, text/javascript
```

**Example response:**

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

[
  {
    "code": "cs",
    "failing": 0,
    "failing_percent": 0.0,
    "fuzzy": 0,
    "fuzzy_percent": 0.0,
    "last_author": "Michal Čihař",
    "last_change": "2012-03-28T15:07:38+00:00",
    "name": "Czech",
    "total": 436,
    "total_words": 15271,
    "translated": 436,
    "translated_percent": 100.0,
    "translated_words": 3201,
    "url": "http://hosted.weblate.org/engage/weblate/cs/",
    "url_translate": "http://hosted.weblate.org/projects/weblate/main/cs/"
  }
]```

(sonraki sayfaya devam)
1.12.20 RSS bildirimleri

Çevirilerdeki değişiklikler RSS bildirimlerinde dışa aktarılır.

**GET /exports/rss/(string: project)/**

Bir çeviri için son değişiklikleri içeren RSS bildirimini alır.

**GET /exports/rss/(string: project)/string: component/string: language/**

Bir bileşeni için son değişiklikleri içeren RSS bildirimini alır.

**GET /exports/rss/(string: project)/**

Bir proje için son değişiklikleri içeren RSS bildirimini alır.

**GET /exports/rss/language/(string: language)/**

Bir dil için son değişiklikleri içeren RSS bildirimini alır.

**GET /exports/rss/**

Weblate örneği için son değişiklikleri içeren RSS bildirimini alır.

**Ayrıca bakınız:**

Wikipedia’da RSS
1.13 Weblate Client

2.7 sürümünde geldi: There has been full wlc utility support ever since Weblate 2.7. If you are using an older version some incompatibilities with the API might occur.

1.13.1 Installation

The Weblate Client is shipped separately and includes the Python module. To use the commands below, you need to install `wlc`:

```
pip3 install wlc
```

1.13.2 Docker kullanımı

The Weblate Client is also available as a Docker image. The image is published on Docker Hub: https://hub.docker.com/r/weblate/wlc

Kurulum:

```
docker pull weblate/wlc
```

The Docker container uses Weblate’s default settings and connects to the API deployed in localhost. The API URL and API_KEY can be configured through the arguments accepted by Weblate.

The command to launch the container uses the following syntax:

```
docker run --rm weblate/wlc [WLC_ARGS]
```

Example:

```
docker run --rm weblate/wlc --url https://hosted.weblate.org/api/ list-projects
```

You might want to pass your `Configuration files` to the Docker container, the easiest approach is to add your current directory as `/home/weblate` volume:

```
docker run --volume $PWD:/home/weblate --rm weblate/wlc show
```

1.13.3 Getting started

The wlc configuration is stored in `~/.config/weblate` (see `Configuration files` for other locations), please create it to match your environment:

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

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

You can then invoke commands on the default server:

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

Ayrıca bakınız:

`Configuration files`
1.13.4 Synopsis

```
wl [arguments] <command> [options]
```

Commands actually indicate which operation should be performed.

1.13.5 Açıklama

Weblate Client is a Python library and command-line utility to manage Weblate remotely using Weblate’s REST API. The command-line utility can be invoked as `wl` and is built-in on `wl`.

Argümanlar

The program accepts the following arguments which define output format or which Weblate instance to use. These must be entered before any command.

```
--format {csv, json, text, html}
    Specify the output format.

--url URL
    Specify the API URL. Overrides any value found in the configuration file, see Configuration files. The URL should end with /api/, for example https://hosted.weblate.org/api/.

--key KEY
    Specify the API user key to use. Overrides any value found in the configuration file, see Configuration files. You can find your key in your profile on Weblate.

--config PATH
    Overrides the configuration file path, see Configuration files.

--config-section SECTION
    Overrides configuration file section in use, see Configuration files.
```

Komutlar

The following commands are available:

```
version
    Prints the current version.

list-languages
    Lists used languages in Weblate.

list-projects
    Lists projects in Weblate.

list-components
    Lists components in Weblate.

list-translations
    Lists translations in Weblate.

show
    Shows Weblate object (translation, component or project).

ls
    Lists Weblate object (translation, component or project).

commit
    Commits changes made in a Weblate object (translation, component or project).
```
pull
Pulls remote repository changes into Weblate object (translation, component or project).

push
Pushes Weblate object changes into remote repository (translation, component or project).

reset
0.7 sürümünde geldi: Supported since wlc 0.7.
Resets changes in Weblate object to match remote repository (translation, component or project).

cleanup
0.9 sürümünde geldi: Supported since wlc 0.9.
Removes any untracked changes in a Weblate object to match the remote repository (translation, component or project).
epo
Displays repository status for a given Weblate object (translation, component or project).

statistics
Displays detailed statistics for a given Weblate object (translation, component or project).

lock-status
0.5 sürümünde geldi: Supported since wlc 0.5.
Displays lock status.

lock
0.5 sürümünde geldi: Supported since wlc 0.5.
Locks component from further translation in Weblate.

unlock
0.5 sürümünde geldi: Supported since wlc 0.5.
Unlocks translation of Weblate component.

changes
0.7 sürümünde geldi: Supported since wlc 0.7 and Weblate 2.10.
Displays changes for a given object.

download
0.7 sürümünde geldi: Supported since wlc 0.7.
Downloads a translation file.

--convert
Converts file format, if unspecified no conversion happens on the server and the file is downloaded as is to the repository.

--output
Specifies file to save output in, if left unspecified it is printed to stdout.

upload
0.9 sürümünde geldi: Supported since wlc 0.9.
Uploads a translation file.

--overwrite
Overwrite existing translations upon uploading.

--input
File from which content is read, if left unspecified it is read from stdin.

İpucu: You can get more detailed information on invoking individual commands by passing --help, for example: wlc ls --help.
1.13.6 Configuration files

`.weblate`, `.weblate.ini`, `weblate.ini` 1.6 sürümünde değişti: The files with `.ini` extension are accepted as well.

Per project configuration file

C:\Users\NAME\AppData\weblate.ini 1.6 sürümünde geldi.

User configuration file on Windows.

`~/.config/weblate` User configuration file

`/etc/xdg/weblate` System wide configuration file

The program follows the XDG specification, so you can adjust placement of config files by environment variables `XDG_CONFIG_HOME` or `XDG_CONFIG_DIRS`. On Windows `APPDATA` directory is preferred location for the configuration file.

Following settings can be configured in the `[weblate]` section (you can customize this by `--config-section`):

key
   API KEY to access Weblate.

url
   API server URL, defaults to `http://127.0.0.1:8000/api/`.

translation
   Path to the default translation - component or project.

The configuration file is an INI file, for example:

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

Additionally API keys can be stored in the `[keys]` section:

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

This allows you to store keys in your personal settings, while using the `.weblate` configuration in the VCS repository so that `wl` knows which server it should talk to.

1.13.7 Examples

Print current program version:

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

List all projects:

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

You can also designate what project `wl` should work on:
With this setup it is easy to commit pending changes in the current project:

```
$ wlc commit
```

# 1.14 Weblate’s Python API

## 1.14.1 Installation

The Python API is shipped separately, you need to install the **Weblate Client**: (wlc) to have it.

```
pip install wlc
```

## 1.14.2 wlc

### WeblateException

**exception** `wlc.WeblateException`

Base class for all exceptions.

### Weblate

**class** `wlc.Weblate (key =", url =None, config =None)`

**Parametreler**

- `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)

Parametreler path (str) – Request path

Dönüştürü object

Performs a single API GET call.

post (path, **kwargs)

Parametreler path (str) – Request path

Dönüştürü object

Performs a single API GET call.

1.14.3 wlc.config

WeblateConfig

class wlc.config.WeblateConfig (section = ’wlc’)

Parametreler section (str) – Configuration section to use

Configuration file parser following XDG specification.

load (path = None)

Parametreler 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)

Parametreler

• 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.
2.1 Configuration instructions

2.1.1 Installing Weblate

Installing using Docker

With dockerized Weblate deployment you can get your personal Weblate instance up and running in seconds. All of Weblate’s dependencies are already included. PostgreSQL is set up as the default database.

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

Not: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.
Installation

The following examples assume you have a working Docker environment, with `docker-compose` installed. Please check the Docker documentation for instructions.

1. Clone the weblate-docker repo:

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

2. Create a `docker-compose.override.yml` file with your settings. See *Docker environment variables* for full list of environment variables.

   ```
   version: '3'
   services:
     weblate:
       ports:
       - 80:8080
       environment:
         WEBLATE_EMAIL_HOST: smtp.example.com
         WEBLATE_EMAIL_HOST_USER: user
         WEBLATE_EMAIL_HOST_PASSWORD: pass
         WEBLATE_SERVER_EMAIL: weblate@example.com
         WEBLATE_DEFAULT_FROM_EMAIL: weblate@example.com
         WEBLATE_SITE_DOMAIN: weblate.example.com
         WEBLATE_ADMIN_PASSWORD: password for the admin user
         WEBLATE_ADMIN_EMAIL: weblate.admin@example.com
   ```

   **Not:** If `WEBLATE_ADMIN_PASSWORD` is not set, the admin user is created with a random password shown on first startup.

   The provided example makes Weblate listen on port 80, edit the port mapping in the `docker-compose.override.yml` file to change it.

3. Start Weblate containers:

   ```
   docker-compose up
   ```

   Enjoy your Weblate deployment, it’s accessible on port 80 of the `weblate` container.

2.15-2 sürümünde değişti: The setup has changed recently, priorly there was separate web server container, since 2.15-2 the web server is embedded in the Weblate container.

3.7.1-6 sürümünde değişti: In July 2019 (starting with the 3.7.1-6 tag), the containers are not running as a root user. This has changed the exposed port from 80 to 8080.

**Ayrıca bakınız:**

*Invoking management commands*
Choosing Docker hub tag

You can use following tags on Docker hub, see https://hub.docker.com/r/weblate/weblate/tags/ for full list of available ones.

<table>
<thead>
<tr>
<th>Etiket adı</th>
<th>Açıklama</th>
<th>Use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>latest</td>
<td>Weblate stable release, matches latest tagged release</td>
<td>Rolling updates in a production environment</td>
</tr>
<tr>
<td>&lt;VERSION&gt;-&lt;PATCH&gt;</td>
<td>Weblate kararlı sürümü</td>
<td>Well defined deploy in a production environment</td>
</tr>
<tr>
<td>edge</td>
<td>Weblate stable release with development changes in the Docker container (for example updated dependencies)</td>
<td>Rolling updates in a staging environment</td>
</tr>
<tr>
<td>edge-&lt;DATE&gt;-&lt;SHA&gt;</td>
<td>Weblate stable release with development changes in the Docker container (for example updated dependencies)</td>
<td>Well defined deploy in a staging environment</td>
</tr>
<tr>
<td>bleeding</td>
<td>Development version Weblate from Git</td>
<td>Rolling updates to test upcoming Weblate features</td>
</tr>
<tr>
<td>bleeding-&lt;DATE&gt;-&lt;SHA&gt;</td>
<td>Development version Weblate from Git</td>
<td>Well defined deploy to test upcoming Weblate features</td>
</tr>
</tbody>
</table>

Every image is tested by our CI before it gets published, so even the bleeding version should be quite safe to use.

Docker container with HTTPS support

Please see Installation for generic deployment instructions, this section only mentions differences compared to it.

Using own SSL certificates

3.8-3 sürümünde geldi.

In case you have own SSL certificate you want to use, simply place the files into the Weblate data volume (see Docker container volumes):

- ssl/fullchain.pem containing the certificate including any needed CA certificates
- ssl/privkey.pem containing the private key

Both of these files must be owned by the same user as the one starting the docker container and have file mask set to 600 (readable and writable only by the owning user).

Additionally, Weblate container will now accept SSL connections on port 4443, you will want to include the port forwarding for HTTPS in docker compose override:

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

If you already host other sites on the same server, it is likely ports 80 and 443 are used by a reverse proxy, such as NGINX. To pass the HTTPS connection from NGINX to the docker container, you can use the following configuration:

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

(sonraki sayfaya devam)
server_name <SITE_URL>;
ssl_certificate /etc/letsencrypt/live/<SITE>/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/<SITE>/privkey.pem;

location / {
    proxy_set_header HOST $host;
    proxy_set_header X-Forwarded-Proto https;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Host $server_name;
    proxy_pass https://127.0.0.1:<EXPOSED_DOCKER_PORT>;
}

Replace <SITE_URL>, <SITE> and <EXPOSED_DOCKER_PORT> with actual values from your environment.

Automatic SSL certificates using Let's Encrypt

In case you want to use Let's Encrypt automatically generated SSL certificates on public installation, you need to add a reverse HTTPS proxy an additional Docker container, https-portal will be used for that. This is made use of in the docker-compose-https.yml file. Then create a docker-compose-https.override.yml file with your settings:

```
version: '3'
services:
    weblate:
        environment:
            WEBLATE_EMAIL_HOST: smtp.example.com
            WEBLATE_EMAIL_HOST_USER: user
            WEBLATE_EMAIL_HOST_PASSWORD: pass
            WEBLATE_SITE_DOMAIN: weblate.example.com
            WEBLATE_ADMIN_PASSWORD: password for admin user
    https-portal:
        environment:
            DOMAINS: 'weblate.example.com -> http://weblate:8080'
```

Whenever invoking `docker-compose` you need to pass both files to it, and then do:

```
docker-compose -f docker-compose-https.yml -f docker-compose-https.override.yml up
```

Upgrading the Docker container

Usually it is good idea to only update the Weblate container and keep the PostgreSQL container at the version you have, as upgrading PostgreSQL is quite painful and in most cases does not bring many benefits.

You can do this by sticking with the existing docker-compose and just pull the latest images and then restart:

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

The Weblate database should be automatically migrated on first startup, and there should be no need for additional manual actions.
Not: Upgrades across 3.0 are not supported by Weblate. If you are on 2.x series and want to upgrade to 3.x, first upgrade to the latest 3.0.1-x (at time of writing this it is the 3.0.1-7) image, which will do the migration and then continue upgrading to newer versions.

You might also want to update the docker-compose repository, though it’s not needed in most case. Please beware of PostgreSQL version changes in this case as it’s not straightforward to upgrade the database, see GitHub issue for more info.

Yönetici olarak oturum açma

After container setup, you can sign in as admin user with password provided in WEBLATE_ADMIN_PASSWORD, or a random password generated on first start if that was not set.

To reset admin password, restart the container with WEBLATE_ADMIN_PASSWORD set to new password.

Ayrıca bakınız:
WEBLATE_ADMIN_PASSWORD, WEBLATE_ADMIN_NAME, WEBLATE_ADMIN_EMAIL

Number of processes and memory consumption

The number of worker processes for both uWSGI and Celery is determined automatically based on number of CPUs. This works well for most cloud virtual machines as these typically have few CPUs and good amount of memory.

In case you have a lot of CPU cores and hit out of memory issues, try reducing number of workers:

```
environment:
    WEBLATE_WORKERS: 2
```

You can also fine-tune individual worker categories:

```
environment:
    UWSGI_WORKERS: 4
    CELERY_MAIN_OPTIONS: --concurrency 2
    CELERY_NOTIFY_OPTIONS: --concurrency 1
    CELERY_TRANSLATE_OPTIONS: --concurrency 1
```

Ayrıca bakınız:
WEBLATE_WORKERS, CELERY_MAIN_OPTIONS, CELERY_NOTIFY_OPTIONS, CELERY_MEMORY_OPTIONS, CELERY_TRANSLATE_OPTIONS, CELERY_BACKUP_OPTIONS, CELERY_BEAT_OPTIONS, UWSGI_WORKERS

Yatay ölçeklendirme

4.6 sürümünde geldi.

Uyarı: Bu özellikle bir teknoloji ön izlemesidir.

You can run multiple Weblate containers to scale the service horizontally. The /app/data volume has to be shared by all containers, it is recommended to use cluster filesystem such as GlusterFS for this. The /app/cache volume should be separate for each container.

Each Weblate container has defined role using WEBLATE_SERVICE environment variable. Please follow carefully the documentation as some of the services should be running just once in the cluster and the ordering of the services matters as well.
You can find example setup in the docker-compose repo as docker-compose-split.yml.

**Docker environment variables**

Many of Weblate’s Configuration can be set in the Docker container using environment variables:

**Generic settings**

**WEBLATE_DEBUG**

Configures Django debug mode using DEBUG.

Example:

```
environment:
  WEBLATE_DEBUG: 1
```

Ayrıca bakınız:

 Disable debug mode

**WEBLATE_LOGLEVEL**

Configures the logging verbosity.

**WEBLATE_SITE_TITLE**

Changes the site-title shown in the header of all pages.

**WEBLATE_SITE_DOMAIN**

Site etki alanını yapılandırır. Bu parametre gereklidir.

Ayrıca bakınız:

 Set correct site domain, SITE_DOMAIN

**WEBLATE_ADMIN_NAME**

**WEBLATE_ADMIN_EMAIL**

Configures the site-admin’s name and e-mail. It is used for both ADMINS setting and creating admin user (see WEBLATE_ADMIN_PASSWORD for more info on that).

Example:

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

Ayrıca bakınız:

 Yönetici olarak oturum açma, Properly configure admins, ADMINS

**WEBLATE_ADMIN_PASSWORD**

Sets the password for the admin user.

- If not set and admin user does not exist, it is created with a random password shown on first container startup.

- If not set and admin user exists, no action is performed.

- If set the admin user is adjusted on every container startup to match WEBLATE_ADMIN_PASSWORD, WEBLATE_ADMIN_NAME and WEBLATE_ADMIN_EMAIL.

**Uyarı:** It might be a security risk to store password in the configuration file. Consider using this variable only for initial setup (or let Weblate generate random password on initial startup) or for password recovery.
Ayrıca bakınız:

**WEBLATE_ADMIN_PASSWORD**
Sets the path to a file containing the password for the **admin** user.

Ayrıca bakınız:

**WEBLATE_ADMIN_PASSWORD_FILE**
Set the path to a file containing the password for the admin user.

**WEBLATE_SERVER_EMAIL**

**WEBLATE_DEFAULT_FROM_EMAIL**
Configures the address for outgoing e-mails.

Ayrıca bakınız:

**WEBLATE_CONTACT_FORM**
Configures contact form behavior, see [CONTACT_FORM](#).

**WEBLATE_ALLOWED_HOSTS**
Configures allowed HTTP hostnames using [ALLOWED_HOSTS](#).

Defaults to * which allows all hostnames.

**WEBLATE_REGISTRATION_OPEN**
Configures whether registrations are open by toggling [REGISTRATION_OPEN](#).

**WEBLATE_REGISTRATION_ALLOW_BACKENDS**
Configure which authentication methods can be used to create new account via [REGISTRATION_ALLOW_BACKENDS](#).

**WEBLATE_TIME_ZONE**
Configures the used time zone in Weblate, see [TIME_ZONE](#).

**Not:** To change the time zone of the Docker container itself, use the TZ environment variable.

**Example:**

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

Ayrıca bakınız:

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

**Example:**

```yaml
environment:
  WEBLATE_REGISTRATION_OPEN: 0

WEBLATE_REGISTRATION_ALLOW_BACKENDS: azuread-oauth2,azuread-tenant-oauth2

WEBLATE_TIME_ZONE: Europe/Prague
```

**WEBLATE_ADMIN_NAME**

**WEBLATE_ADMIN_EMAIL**

**WEBLATE_TIME_ZONE**
Configure the timezone in Weblate, see `TIME_ZONE`.

Not: To change the time zone of the Docker container itself, use the `TZ` environment variable.

**Example:**

```yaml
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.

*İpucu:* Please see `ENABLE_HTTPS` documentation for possible caveats.

*Not:* This does not make the Weblate container accept HTTPS connections, you need to configure that as well, see *Docker container with HTTPS support* for examples.

**Example:**

```
environment:
  WEBLATE_ENABLE_HTTPS: 1
```

**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`.

*Not:* The format must conform to Django's expectations. Django transforms raw HTTP header names as follows:

- converts all characters to uppercase
- replaces any hyphens with underscores
- prepends `HTTP_` prefix

So `X-Forwarded-For` would be mapped to `HTTP_X_FORWARDED_FOR`.

**Example:**

```
environment:
  WEBLATE_IP_PROXY_HEADER: HTTP_X_FORWARDED_FOR
```

**WEBLATE_SECURE_PROXY_SSL_HEADER**

A tuple representing a HTTP header/value combination that signifies a request is secure. This is needed when Weblate is running behind a reverse proxy doing SSL termination which does not pass standard HTTPS headers.

**Example:**

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

**WEBLATE_REQUIRE_LOGIN**

Enables `REQUIRE_LOGIN` to enforce authentication on whole Weblate.

**Example:**

```
environment:
  WEBLATE_REQUIRE_LOGIN: 1
```
WEBLATE_ADD_LOGIN_REQUIRED_URLS_EXCEPTIONS

WEBLATE_REMOVE_LOGIN_REQUIRED_URLS_EXCEPTIONS

Adds URL exceptions for authentication required for the whole Weblate installation using LOGIN_REQUIRED_URLS_EXCEPTIONS.

You can either replace whole settings, or modify default value using ADD and REMOVE variables.

WEBLATE_GOOGLE_ANALYTICS_ID

Configures ID for Google Analytics by changing GOOGLE_ANALYTICS_ID.

WEBLATE_GITHUB_USERNAME

Configures GitHub username for GitHub pull-requests by changing GITHUB_USERNAME.

Ayrıca bkz:

GitHub

WEBLATE_GITHUB_TOKEN

4.3 sürümünde geldi.

Configures GitHub personal access token for GitHub pull-requests via API by changing GITHUB_TOKEN.

Ayrıca bkz:

GitHub

WEBLATE_GITLAB_USERNAME

Configures GitLab username for GitLab merge-requests by changing GITLAB_USERNAME

Ayrıca bkz:

GitLab

WEBLATE_GITLAB_TOKEN

Configures GitLab personal access token for GitLab merge-requests via API by changing GITLAB_TOKEN.

Ayrıca bkz:

GitLab

WEBLATE_PAGURE_USERNAME

Configures Pagure username for Pagure merge-requests by changing PAGURE_USERNAME

Ayrıca bkz:

Pagure

WEBLATE_PAGURE_TOKEN

Configures Pagure personal access token for Pagure merge-requests via API by changing PAGURE_TOKEN

Ayrıca bkz:

Pagure

WEBLATE_SIMPLIFY_LANGUAGES

Configures the language simplification policy, see SIMPLIFY_LANGUAGES.

WEBLATE_DEFAULT_ACCESS_CONTROL

Configures the default Erişim denetimi 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 Çeviri yaymaya izin ver for new components, see DEFAULT_TRANSLATION_PROPAGATION.

WEBLATE_DEFAULT_COMMITER_EMAIL

Configures DEFAULT_COMMITER_EMAIL.
WEBLATE_DEFAULT_COMMITER_NAME
  Configures DEFAULT_COMMITER_NAME.

WEBLATE_DEFAULT_SHARED_TM
  Configures DEFAULT_SHARED_TM.

WEBLATE_AKISMET_API_KEY
  Configures the Akismet API key, see AKISMET_API_KEY.

WEBLATE_GPG_IDENTITY
  Configures GPG signing of commits, see WEBLATE_GPG_IDENTITY.

  Ayrıca bakınız:
  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.

  Ayrıca bakınız:
  Content security policy, CSP_SCRIPT_SRC, CSP_IMG_SRC, CSP_CONNECT_SRC, CSP_STYLE_SRC, CSP_FONT_SRC

WEBLATE_LICENSE_FILTER
  Configures LICENSE_FILTER.

WEBLATE_LICENSE_REQUIRED
  Configures LICENSE_REQUIRED

WEBLATE_WEBSITE_REQUIRED
  Configures WEBSITE_REQUIRED

WEBLATE_HIDE_VERSION
  Configures HIDE_VERSION.

WEBLATE_BASIC_LANGUAGES
  Configures BASIC_LANGUAGES.

WEBLATE_DEFAULT_AUTO_WATCH
  Configures DEFAULT_AUTO_WATCH.

WEBLATE_RATELIMIT_ATTEMPTS
WEBLATE_RATELIMIT_LOCKOUT
WEBLATE_RATELIMIT_WINDOW
  4.6 sürümünde geldi.

  Oran smurlayıcıyı yapılandırmır.

  **İpucu:** You can set configuration for any rate limiter scopes. To do that add WEBLATE_ prefix to any of setting described in Rate limiting.
Ayrıca bakınız:

Rate limiting, RATELIMIT_ATTEMPTS, RATELIMIT_WINDOW, RATELIMIT_LOCKOUT

**WEBLATE_ENABLE_AVATARS**

4.6.1 sürümünde geldi.

Configures ENABLE_AVATARS.

### Machine translation settings

İpucu: Configuring API key for a service automatically configures it in `MT_SERVICES`.

**WEBLATE_MT_APERTIUM_API**

Enables Apertium machine translation and sets `MT_APERTIUM_API`

**WEBLATE_MT_AWS_REGION**

**WEBLATE_MT_AWS_ACCESS_KEY_ID**

**WEBLATE_MT_AWS_SECRET_ACCESS_KEY**

Configures AWS machine translation.

<table>
<thead>
<tr>
<th>environment:</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEBLATE_MT_AWS_REGION: us-east-1</td>
</tr>
<tr>
<td>WEBLATE_MT_AWS_ACCESS_KEY_ID: AKIAIOSFODNN7EXAMPLE</td>
</tr>
<tr>
<td>WEBLATE_MT_AWS_SECRET_ACCESS_KEY: wJalrXUttnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY</td>
</tr>
</tbody>
</table>

**WEBLATE_MT_DEEPL_KEY**

Enables DeepL machine translation and sets `MT_DEEPL_KEY`

**WEBLATE_MT_DEEPL_API_URL**

Configures DeepL API version to use, see `MT_DEEPL_API_URL`.

**WEBLATE_MT_LIBRETRANSLATE_KEY**

Enables LibreTranslate machine translation and sets `MT_LIBRETRANSLATE_KEY`

**WEBLATE_MT_LIBRETRANSLATE_API_URL**

Configures LibreTranslate API instance to use, see `MT_LIBRETRANSLATE_API_URL`.

**WEBLATE_MT_GOOGLE_KEY**

Enables Google Translate and sets `MT_GOOGLE_KEY`

**WEBLATE_MT_MICROSOFT_COGNITIVE_KEY**

Enables Microsoft Cognitive Services Translator and sets `MT_MICROSOFT_COGNITIVE_KEY`

**WEBLATE_MT_MICROSOFT_ENDPOINT_URL**

Sets `MT_MICROSOFT_ENDPOINT_URL`, please note this is supposed to contain domain name only.

**WEBLATE_MT_MICROSOFT_REGION**

Sets `MT_MICROSOFT_REGION`

**WEBLATE_MT_MICROSOFT_BASE_URL**

Sets `MT_MICROSOFT_BASE_URL`

**WEBLATE_MT_MODERNMT_KEY**

Enables ModernMT and sets `MT_MODERNMT_KEY`.

**WEBLATE_MT_MYMEMORY_ENABLED**

Enables MyMemory machine translation and sets `MT_MYMEMORY_EMAIL` to `WEBLATE_ADMIN_EMAIL`.

Example:
The Weblate Manual, Yayım 4.9

WEBLATE_MT_GLOSBE_ENABLED
Enables Glosbe machine translation.

WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED
Enables Microsoft Terminology Service machine translation.

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.

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:

WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE: uid =%%(user)s,ou =People,dc =example,
.. dc =net

2.1. Configuration instructions 151
# map weblate 'full_name' to ldap 'name' and weblate 'email' attribute to ldap 'mail' 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:

```ini
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:

```ini
environment:
WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
WEBLATE_AUTH_LDAP_BIND_DN: CN =ldap,CN =Users,DC =example,DC =com
WEBLATE_AUTH_LDAP_BIND_PASSWORD: password
WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
WEBLATE_AUTH_LDAP_USER_SEARCH_UNION: ou =users,dc =example,ou =otherusers,dc =example,dc =com
```

Example with search and bind against Active Directory:

```ini
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)
```

Ayrıca bakınız:

LDAP authentication

GitHub

WEBLATE_SOCIAL_AUTH_GITHUB_KEY
WEBLATE_SOCIAL_AUTH_GITHUB_SECRET
Enables GitHub authentication.

Bitbucket

WEBLATE_SOCIAL_AUTH_BITBUCKET_KEY
WEBLATE_SOCIAL_AUTH_BITBUCKET_SECRET
Enables Bitbucket authentication.
Facebook

WEBLATE_SOCIAL_AUTH_FACEBOOK_KEY
WEBLATE_SOCIAL_AUTH_FACEBOOK_SECRET
   Enables Facebook OAuth 2.

Google

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_KEY
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_DOMAINS
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_EMAILS
   Enables Google OAuth 2.

GitLab

WEBLATE_SOCIAL_AUTH_GITLAB_KEY
WEBLATE_SOCIAL_AUTH_GITLAB_SECRET
WEBLATE_SOCIAL_AUTH_GITLAB_API_URL
   Enables GitLab OAuth 2.

Azure Active Directory

WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_KEY
WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET
   Enables Azure Active Directory authentication, see Microsoft Azure 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 Microsoft 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 authentication](#).

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.

- **Ayrıca bakınız:**
  *Database setup for Weblate*

PostgreSQL password.

- `POSTGRES_PASSWORD`

Path to the file containing the PostgreSQL password. Use as an alternative to `POSTGRES_PASSWORD`.

- `POSTGRES_USER`

PostgreSQL username.

- `POSTGRES_DATABASE`

PostgreSQL database name.

- `POSTGRES_HOST`

PostgreSQL server hostname or IP address. Defaults to `database`.

- `POSTGRES_PORT`

PostgreSQL server port. Defaults to none (uses the default value).
POSTGRES_SSL_MODE
Configure how PostgreSQL handles SSL in connection to the server, for possible choices see SSL Mode Descriptions.

POSTGRES_ALTER_ROLE
Configures name of role to alter during migrations, see Configuring Weblate to use PostgreSQL.

POSTGRES_CONN_MAX_AGE
4.8.1 sürümünde geldi.

The lifetime of a database connection, as an integer of seconds. Use 0 to close database connections at the end of each request (this is the default behavior).

Enabling connection persistence will typically, cause more open connection to the database. Please adjust your database configuration prior enabling.

**Uyarı:** This is currently experimental, not widely tested and not supported by the Weblate team.

Ayrıca bakınız:

CONN_MAX_AGE, Persistent connections

Database backup settings

Ayrıca bakınız:

Dumped data for backups

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.

Ayrıca bakınız:

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.

2.1. Configuration instructions
Email server setup

To make outgoing e-mail work, you need to provide a mail server.

Example TLS configuration:

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

Example SSL configuration:

```bash
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
```

Ayrıca bakınız:

*Configuring outgoing e-mail*

**WEBLATE_EMAIL_HOST**
Mail server hostname or IP address.

Ayrıca bakınız:

```
WEBLATE_EMAIL_PORT, WEBLATE_EMAIL_USE_SSL, WEBLATE_EMAIL_USE_TLS, EMAIL_HOST
```

**WEBLATE_EMAIL_PORT**
Mail server port, defaults to 25.

Ayrıca bakınız:

```
EMAIL_PORT
```

**WEBLATE_EMAIL_HOST_USER**
E-posta kimlik doğrulama kullanıcı.

Ayrıca bakınız:

```
EMAIL_HOST_USER
```

**WEBLATE_EMAIL_HOST_PASSWORD**
E-posta kimlik doğrulama parolası.

Ayrıca bakınız:

```
EMAIL_HOST_PASSWORD
```

**WEBLATE_EMAIL_HOST_PASSWORD_FILE**
E-posta kimlik doğrulama parolasını içeren dosyanın yolu.

Ayrıca bakınız:

```
WEBLATE_EMAIL_HOST_PASSWORD
```

**WEBLATE_EMAIL_USE_SSL**
Whether to use an implicit TLS (secure) connection when talking to the SMTP server. In most e-mail documentation, this type of TLS connection is referred to as SSL. It is generally used on port 465. If you are experiencing problems, see the explicit TLS setting `WEBLATE_EMAIL_USE_TLS`.

Ayrıca bakınız:

```
WEBLATE_EMAIL_PORT, WEBLATE_EMAIL_USE_TLS, EMAIL_USE_SSL
```
WEBLATE_EMAIL_USE_TLS
Whether to use a TLS (secure) connection when talking to the SMTP server. This is used for explicit TLS connections, generally on port 587 or 25. If you are experiencing connections that hang, see the implicit TLS setting WEBLATE_EMAIL_USE_SSL.

Ayrıca bakınız:
WEBLATE_EMAIL_PORT, WEBLATE_EMAIL_USE_SSL, EMAIL_USE_TLS

WEBLATE_EMAIL_BACKEND
Configures Django back-end to use for sending e-mails.

Ayrıca bakınız:
Configure e-mail sending, EMAIL_BACKEND

Site tümleşimi

WEBLATE_GET_HELP_URL
Configures GETHELP URL.

WEBLATE_STATUS_URL
Configures STATUS URL.

WEBLATE_LEGAL_URL
Configures LEGAL URL.

WEBLATE_PRIVACY_URL
Configures PRIVACY URL.

Error reporting

It is recommended to collect errors from the installation systematically, see Collecting error reports.

To enable support for Rollbar, set the following:

ROLLBAR_KEY
Your Rollbar post server access token.

ROLLBAR_ENVIRONMENT
Your Rollbar environment, defaults to production.

To enable support for Sentry, set following:

SENTRY_DSN
Your Sentry DSN.

SENTRY_ENVIRONMENT
Your Sentry Environment (optional).

Yerelleştirme CDN

WEBLATE_LOCALIZE_CDN_URL

WEBLATE_LOCALIZE_CDN_PATH
4.2.1 sürümündeki geldi.

Configuration for JavaScript yerelleştirme CDN's.

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:
### The Weblate Manual, Yayın 4.9

<table>
<thead>
<tr>
<th>environment:</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEBLATE_LOCALIZE_CDN_URL: <a href="https://cdn.example.com/">https://cdn.example.com/</a></td>
</tr>
<tr>
<td>WEBLATE_LOCALIZE_CDN_PATH: /app/data/l10n-cdn</td>
</tr>
</tbody>
</table>

**Not:** You are responsible for setting up serving of the files generated by Weblate, it only does stores the files in configured location.

**Ayrıca bırakınız:**

weblate-cdn, LOCALIZE_CDN_URL, LOCALIZE_CDN_PATH

---

### Changing enabled apps, checks, addons or autofixes

3.8.5 sürümünde geldi.

The built-in configuration of enabled checks, addons or autofixes can be adjusted by the following variables:

- WEBLATE_ADD_APPS
- WEBLATE_REMOVE_APPS
- WEBLATE_ADD_CHECK
- WEBLATE_REMOVE_CHECK
- WEBLATE_ADD_AUTOFIX
- WEBLATE_REMOVE_AUTOFIX
- WEBLATE_ADD_ADDONS
- WEBLATE_REMOVE_ADDONS

Example:

<table>
<thead>
<tr>
<th>environment:</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEBLATE_REMOVE_AUTOFIX: weblate.trans.autofixes.whitespace.</td>
</tr>
<tr>
<td>WEBLATE_ADD_ADDONS: customize.addons.MyAddon, customize.addons.OtherAddon</td>
</tr>
</tbody>
</table>

**Ayrıca bırakınız:**

CHECK_LIST, AUTOFIX_LIST, WEBLATE_ADDONS, INSTALLED_APPS

---

### Konteyner ayarları

WEBLATE_WORKERS

4.6.1 sürümünde geldi.

Base number of worker processes running in the container. When not set it is determined automatically on container startup based on number of CPU cores available.

It is used to determine CELERY_MAIN_OPTIONS, CELERY_NOTIFY_OPTIONS, CELERY_MEMORY_OPTIONS, CELERY_TRANSLATE_OPTIONS, CELERY_BACKUP_OPTIONS, CELERY_BEAT_OPTIONS, and UWSGI_WORKERS. You can use these settings to fine-tune.

CELE}R}Y_MAIN_OPTIONS
CELE}R}Y_NOTIFY_OPTIONS
CELE}R}Y_MEMORY_OPTIONS
CELE}R}Y_TRANSLATE_OPTIONS
CELE}R}Y_BACKUP_OPTIONS
CELE}R}Y_BEAT_OPTIONS
UWSGI_WORKERS
CELERY_BACKUP_OPTIONS
CELERY_BEAT_OPTIONS
These variables allow you to adjust Celery worker options. It can be useful to adjust concurrency (--concurrency 16) or use different pool implementation (--pool =gevent).

By default, the number of concurrent workers is based on WEBLATE_WORKERS.

Example:

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

Ayrıca bakınız:
Celery worker options, Background tasks using Celery

UWSGI_WORKERS
Configure how many uWSGI workers should be executed.

It defaults to WEBLATE_WORKERS.

Example:

```environment:
UWSGI_WORKERS: 32
```

WEBLATE_SERVICE
Defines which services should be executed inside the container. Use this for Yatay ölçeklendirme.

Aşağıdaki hizmetler tanımlıdır:
- celery-beat Celery task scheduler, only one instance should be running. This container is also responsible for the database structure migrations and it should be started prior others.
- celery-backup Celery worker for backups, only one instance should be running.
- celery-celery Generic Celery worker.
- celery-memory Çeviri Belleği Celery işleyicisi.
- celery-notify Bildirimler Celery işleyicisi.
- celery-translate Otomatik çeviri Celery işleyicisi.
- web Web server.

Docker container volumes

There are two volumes (data and cache) exported by the Weblate container. The other service containers (PostgreSQL or Redis) have their data volumes as well, but those are not covered by this document.

The data volume is used to store Weblate persistent data such as cloned repositories or to customize Weblate installation.

The placement of the Docker volume on host system depends on your Docker configuration, but usually it is stored in /var/lib/docker/volumes/weblate-docker_weblate-data/_data/ (the path consist of name of your docker-compose directory, container, and volume names). In the container it is mounted as /app/data.

The cache volume is mounted as /app/cache and is used to store static files. Its content is recreated on container startup and the volume can be mounted using ephemeral filesystem such as tmpfs.

When creating the volumes manually, the directories should be owned by UID 1000 as that is user used inside the container.

Ayrıca bakınız:
Docker volumes documentation
Further configuration customization

You can further customize Weblate installation in the data volume, see Docker container volumes.

Custom configuration files

You can additionally override the configuration in /app/data/settings-override.py (see Docker container volumes). This is executed at the end of built-in settings, after all environment settings are loaded, and you can adjust or override them.

Replacing logo and other static files

3.8.5 sürümünde geldi.

The static files coming with Weblate can be overridden by placing into /app/data/python/customize/static (see Docker container volumes). For example creating /app/data/python/customize/static/favicon.ico will replace the favicon.

İpucu: The files are copied to the corresponding location upon container startup, so a restart of Weblate is needed after changing the content of the volume.

Alternatively you can also include own module (see Customizing Weblate) and add it as separate volume to the Docker container, for example:

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

Adding own Python modules

3.8.5 sürümünde geldi.

You can place own Python modules in /app/data/python/ (see Docker container volumes) and they can be then loaded by Weblate, most likely by using Custom configuration files.

Ayrıca bakınız:
Customizing Weblate

Installing on Debian and Ubuntu

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space
The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

**Not:** Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

### Installation

#### System requirements

Install the dependencies needed to build the Python modules (see *Software requirements*):

```bash
apt install
libxml2-dev libxslt-dev libfreetype6-dev libjpeg-dev libz-dev libyaml-dev
libcairo-dev gir1.2-pango-1.0 libgirepository1.0-dev libacl1-dev libssl-dev
build-essential python3-gdbm python3-dev python3-pip python3-virtualenv
virtualenv git
```

Install wanted optional dependencies depending on features you intend to use (see *Optional dependencies*):

```bash
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:

```bash
# Web server option 1: NGINX and uWSGI
apt install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with `mod_wsgi`
apt install apache2 libapache2-mod-wsgi-py3

# Caching backend: Redis
apt install redis-server

# Database server: PostgreSQL
apt install postgresql postgresql-contrib

# SMTP server
apt install exim4
```
Python modules

Ipucu: We’re using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:

   ```
   . ~/weblate-env/bin/activate
   ```

3. Install Weblate including all dependencies:

   ```
   pip install Weblate
   ```

4. Install database driver:

   ```
   pip install psycopg2-binary
   ```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check Optional dependencies):

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

Configuring Weblate

Not: 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.


2. Adjust the values in the new settings.py file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see Adjusting configuration.

3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check Database setup for Weblate for production ready setup):

   ```
   weblate migrate
   ```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

   ```
   weblate createadmin
   ```

5. Collect static files for web server (see Running server and Serving static files):

   ```
   weblate collectstatic
   ```

6. Compress JavaScript and CSS files (optional, see Compressing client assets):
The Weblate Manual, Yayım 4.9

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See Background tasks using Celery for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see Running server for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

- You can now access Weblate on http://localhost:8000/.
- Login with admin credentials obtained during installation or register with new users.
- You can now run Weblate commands using weblate command when Weblate virtualenv is active, see Management commands.
- You can stop the test server with Ctrl+C.
- Review potential issues with your installation either on /manage/performance/ URL or using weblate check --deploy, see Production setup.

Adding translation

1. Open the admin interface (http://localhost:8000/create/project/) and create the project you want to translate. See Project configuration for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See Component configuration for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see Supported file formats for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on SUSE and openSUSE

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

2.1. Configuration instructions
The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

**Not:** Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

### Installation

#### System requirements

Install the dependencies needed to build the Python modules (see *Software requirements*):

```bash
zypper install \
  libxslt-devel libxml2-devel freetype-devel libjpeg-devel zlib-devel libyaml-devel \
  cairo-devel typelib-1_0-Pango-1_0 gobject-introspection-devel libacl-devel \
  python3-pip python3-virtualenv python3-devel git
```

Install wanted optional dependencies depending on features you intend to use (see *Optional dependencies*):

```bash
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:

```bash
# 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

**İpucu:** 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:

   ```bash
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:
3. Install Weblate including all dependencies:

```
pip install Weblate
```

4. Install database driver:

```
pip install psycopg2-binary
```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check Optional dependencies):

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

### Configuring Weblate

**Not:** 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`.


2. Adjust the values in the new `settings.py` file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see Adjusting configuration.

3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check Database setup for Weblate for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see Running server and Serving static files):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see Compressing client assets):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See Background tasks using Celery for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see Running server for production setup):

```
weblate runserver
```
After installation

Congratulations, your Weblate server is now running and you can start using it.

- You can now access Weblate on http://localhost:8000/.
- Login with admin credentials obtained during installation or register with new users.
- You can now run Weblate commands using `weblate` command when Weblate virtualenv is active, see Management commands.
- You can stop the test server with Ctrl+C.
- Review potential issues with your installation either on /manage/performance/ URL or using `weblate check --deploy`, see Production setup.

Adding translation

1. Open the admin interface (http://localhost:8000/create/project/) and create the project you want to translate. See Project configuration for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See Component configuration for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see Supported file formats for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on RedHat, Fedora and CentOS

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

Not: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.
Installation

System requirements

Install the dependencies needed to build the Python modules (see *Software requirements*):

```bash
dnf install \n   libxslt-devel libxml2-devel freetype-devel libjpeg-devel zlib-devel libyaml-devel \n   cairo-devel pango-devel gobject-introspection-devel libacl-devel \n   python3-pip python3-virtualenv python3-devel git
```

Install wanted optional dependencies depending on features you intend to use (see *Optional dependencies*):

```bash
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:

```bash
# 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

*İpucu:* 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:

   ```bash
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:

   ```bash
   . ~/weblate-env/bin/activate
   ```

3. Install Weblate including all dependencies:

   ```bash
   pip install Weblate
   ```

4. Install database driver:

   ```bash
   pip install psycopg2-binary
   ```

2.1. Configuration instructions
5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check Optional dependencies):

```bash
pip install ruamel.yaml aedon boto3 zeep chardet tesserocr
```

**Configuring Weblate**

**Not:** 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.


2. Adjust the values in the new settings.py file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see Adjusting configuration.

3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check Database setup for Weblate for production ready setup):

```bash
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```bash
weblate createadmin
```

5. Collect static files for web server (see Running server and Serving static files):

```bash
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see Compressing client assets):

```bash
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:

```bash
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see Running server for production setup):

```bash
weblate runserver
```

**After installation**

Congratulations, your Weblate server is now running and you can start using it.

- You can now access Weblate on http://localhost:8000/.
- Login with admin credentials obtained during installation or register with new users.
- You can now run Weblate commands using weblate command when Weblate virtualenv is active, see Management commands.
- You can stop the test server with Ctrl+C.
• Review potential issues with your installation either on `/manage/performance/` URL or using `weblate check --deploy`, see `Production setup`.

Adding translation

1. Open the admin interface (http://localhost:8000/create/project/) and create the project you want to translate. See `Project configuration` for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See `Component configuration` for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see `Supported file formats` for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on macOS

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2 GB of RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

**Not:** Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

Installation

System requirements

Install the dependencies needed to build the Python modules (see `Software requirements`):

```bash
brew install python pango cairo gobject-introspection libffi glib libyaml
pip3 install virtualenv
```

Make sure pip will be able to find the `libffi` version provided by homebrew — this will be needed during the installation build step.

```bash
export PKG_CONFIG_PATH="/usr/local/opt/libffi/lib/pkgconfig"
```

Install wanted optional dependencies depending on features you intend to use (see `Optional dependencies`):
brew install tesseract

Optionally install software for running production server, see Running server, Database setup for Weblate, Background tasks using Celery. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```bash
# 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

**İpucu:** 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:
   ```bash
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:
   ```bash
   . ~/weblate-env/bin/activate
   ```

3. Install Weblate including all dependencies:
   ```bash
   pip install Weblate
   ```

4. Install database driver:
   ```bash
   pip install psycopg2-binary
   ```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check Optional dependencies):
   ```bash
   pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
   ```

### Configuring Weblate

**Not:** 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.

2. Adjust the values in the new `settings.py` file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see Adjusting configuration.

3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check Database setup for Weblate for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see Running server and Serving static files):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see Compressing client assets):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See Background tasks using Celery for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see Running server for production setup):

```
weblate runserver
```

**After installation**

Congratulations, your Weblate server is now running and you can start using it.

- You can now access Weblate on `http://localhost:8000/`.
- Login with admin credentials obtained during installation or register with new users.
- You can now run Weblate commands using `weblate` command when Weblate virtualenv is active, see Management commands.
- You can stop the test server with Ctrl+C.
- Review potential issues with your installation either on `/manage/performance/` URL or using `weblate check --deploy`, see Production setup.

**Adding translation**

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See Project configuration for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See Component configuration for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see Supported file formats for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.
Installing from sources

1. Please follow the installation instructions for your system first:
   - Installing on Debian and Ubuntu
   - Installing on SUSE and openSUSE
   - Installing on RedHat, Fedora and CentOS

2. Grab the latest Weblate sources using Git (or download a tarball and unpack that):

   ```
git clone https://github.com/WeblateOrg/weblate.git weblate-src
   ```

   Alternatively you can use released archives. You can download them from our website <https://weblate.org/>. Those downloads are cryptographically signed, please see Verifying release signatures.

3. Install current Weblate code into the virtualenv:

   ```
   . ~/weblate-env/bin/activate
   pip install -e weblate-src
   ```

4. Copy `weblate/settings_example.py` to `weblate/settings.py`.

5. Adjust the values in the new `settings.py` file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see Adjusting configuration.

6. Create the database used by Weblate, see Database setup for Weblate.

7. Build Django tables, static files and initial data (see Filling up the database and Serving static files):

   ```
   weblate migrate
   weblate collectstatic
   weblate compress
   weblate compilemessages
   ```

   **Not:** This step should be repeated whenever you update the repository.

Installing on OpenShift

With the OpenShift Weblate template you can get your personal Weblate instance up and running in seconds. All of Weblate’s dependencies are already included. PostgreSQL is set up as the default database and persistent volume claims are used.

You can find the template at <https://github.com/WeblateOrg/openshift/>.

Installation

The following examples assume you have a working OpenShift v3.x environment, with `oc` client tool installed. Please check the OpenShift documentation for instructions.

The `template.yml` is suited for running all components in OpenShift. There is also `template-external-postgresql.yml` which does not start a PostgreSQL server and allows you to configure external PostgreSQL server.
The Weblate Manual, Yayım 4.9

Web Console

Copy the raw content from `template.yml` and import them into your project, then use the Create button in the OpenShift web console to create your application. The web console will prompt you for the values for all of the parameters used by the template.

CLI

To upload the Weblate template to your current project’s template library, pass the `template.yml` file with the following command:

```
   template.yml \
   -n <PROJECT>
```

The template is now available for selection using the web console or the CLI.

Parameters

The parameters that you can override are listed in the parameters section of the template. You can list them with the CLI by using the following command and specifying the file to be used:

```
   openshift/main/template.yml
```

# If the template is already uploaded
$ oc process --parameters -n <PROJECT> weblate

Tanimlama

You can also use the CLI to process templates and use the configuration that is generated to create objects immediately.

```
   template.yml \ 
   -p APPLICATION_NAME=weblate \ 
   -p WEBLATE_VERSION=4.3.1-1 \ 
   -p WEBLATE_SITE_DOMAIN=weblate.app-openshift.example.com \ 
   -p POSTGRESQL_IMAGE=docker-registry.default.svc:5000/openshift/postgresql:9.6 \ 
   -p REDIS_IMAGE=docker-registry.default.svc:5000/openshift/redis:3.2 \ 
   | oc create -f
```

The Weblate instance should be available after successful migration and deployment at the specified `WEBLATE_SITE_DOMAIN` parameter.

After container setup, you can sign in as `admin` user with password provided in `WEBLATE_ADMIN_PASSWORD`, or a random password generated on first start if that was not set.

To reset `admin` password, restart the container with `WEBLATE_ADMIN_PASSWORD` set to new password in the respective Secret.
Kaldırma

```
$ oc delete all -l app=<APPLICATION_NAME>
$ oc delete configmap -l app= <APPLICATION_NAME>
$ oc delete secret -l app=<APPLICATION_NAME>
# ATTENTION! The following command is only optional and will permanently delete... all of your data.
$ oc delete pvc -l app=<APPLICATION_NAME>
$ oc delete all -l app=weblate \ 
  && oc delete secret -l app=weblate \ 
  && oc delete configmap -l app=weblate \ 
  && oc delete pvc -l app=weblate
```

Configuration

By processing the template a respective ConfigMap will be created and which can be used to customize the Weblate image. The ConfigMap is directly mounted as environment variables and triggers a new deployment every time it is changed. For further configuration options, see Docker environment variables for full list of environment variables.

Installing on Kubernetes

**Not:** This guide is looking for contributors experienced with Kubernetes to cover the setup in more details.

With the Kubernetes Helm chart you can get your personal Weblate instance up and running in seconds. All of Weblate’s dependencies are already included. PostgreSQL is set up as the default database and persistent volume claims are used.

You can find the chart at [https://github.com/WeblateOrg/helm/](https://github.com/WeblateOrg/helm/) and it can be displayed at [https://artifacthub.io/packages/helm/weblate/weblate](https://artifacthub.io/packages/helm/weblate/weblate).

**Installation**

```
helm repo add weblate https://helm.weblate.org
helm install my-release weblate/weblate
```

**Configuration**

For further configuration options, see Docker environment variables for full list of environment variables.

Depending on your setup and experience, choose an appropriate installation method for you:

- **Installing using Docker**, recommended for production setups.
- **Virtualenv installation**, recommended for production setups:
  - **Installing on Debian and Ubuntu**
  - **Installing on SUSE and openSUSE**
  - **Installing on RedHat, Fedora and CentOS**
  - **Installing on macOS**
- **Installing from sources**, recommended for development.
- **Installing on OpenShift**
2.1.2 Software requirements

Operating system

Weblate is known to work on Linux, FreeBSD and macOS. Other Unix like systems will most likely work too. Weblate is not supported on Windows. But it may still work and patches are happily accepted.

Other services

Weblate is using other services for its operation. You will need at least following services running:

- PostgreSQL database server, see Database setup for Weblate.
- Redis server for cache and tasks queue, see Background tasks using Celery.
- SMTP server for outgoing e-mail, see Configuring outgoing e-mail.

Python dependencies

Weblate is written in Python and supports Python 3.6 or newer. You can install dependencies using pip or from your distribution packages, full list is available in requirements.txt.

Most notable dependencies:

- Django https://www.djangoproject.com/
- Celery https://docs.celeryproject.org/
- Translate Toolkit https://toolkit.translatehouse.org/
- Python Social Auth https://python-social-auth.readthedocs.io/
- Django REST Framework https://www.django-rest-framework.org/

Optional dependencies

Following modules are necessary for some Weblate features. You can find all of them in requirements-optional.txt.

- Mercurial (optional for Mercurial repositories support) https://www.mercurial-scm.org/
- phply (optional for PHP support) https://github.com/viraptor/phply
- tesserocr (optional for screenshots OCR) https://github.com/sirfz/tesserocr
- akismet (optional for suggestion spam protection) https://github.com/ubernostrum/akismet
- ruamel.yaml (optional for YAML files) https://pypi.org/project/ruamel.yaml/
- Zeep (optional for Microsoft Terminology Service) https://docs.python-zeep.org/
- aeidon (optional for Subtitle files) https://pypi.org/project/aeidon/
Database backend dependencies

Weblate supports PostgreSQL, MySQL and MariaDB, see Database setup for Weblate and backends documentation for more details.

Other system requirements

The following dependencies have to be installed on the system:

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

**Pango, Cairo and related header files and gir introspection data** https://cairographics.org/, https://pango.gnome.org/, see Pango and Cairo

**git-review (optional for Gerrit support)** https://pypi.org/project/git-review/

**git-svn (optional for Subversion support)** https://git-scm.com/docs/git-svn

**tesseract and its data (optional for screenshots OCR)** https://github.com/tesseract-ocr/tesseract

**licensee (optional for detecting license when creating component)** https://github.com/licensee/licensee

Build-time dependencies

To build some of the Python dependencies you might need to install their dependencies. This depends on how you install them, so please consult individual packages for documentation. You won’t need those if using prebuilt wheels while installing using pip or when you use distribution packages.

Pango and Cairo

3.7 sürümünde değişti.

Weblate uses Pango and Cairo for rendering bitmap widgets (see promotion) and rendering checks (see Managing fonts). To properly install Python bindings for those you need to install system libraries first - you need both Cairo and Pango, which in turn need GLib. All those should be installed with development files and GObject introspection data.

2.1.3 Verifying release signatures

Weblate release are cryptographically signed by the releasing developer. Currently this is Michal Čihař. Fingerprint of his PGP key is:

```
63CB 1DF1 EF12 CF2A C0EE 5A32 9C27 B313 42B7 511D
```

and you can get more identification information from <https://keybase.io/nijel>.

You should verify that the signature matches the archive you have downloaded. This way you can be sure that you are using the same code that was released. You should also verify the date of the signature to make sure that you downloaded the latest version.

Each archive is accompanied with .asc files which contain the PGP signature for it. Once you have both of them in the same folder, you can verify the signature:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
```

```
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Ne 3. března 2019, 16:43:15 CET
gpg:   Signature made Ne 3. března 2019, 16:43:15 CET
gpg:   using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
```

```
gpg: Can't check signature: public key not found
```

176 Bölüm 2. Administrator docs
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 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 2009-06-17 [E]
  sub 2015-09-09 [S]
  ```

- Download the keyring from Michal’s server, then import it with:

  ```
  $ gpg --import wmxth3chu9jfxdxywjlskpmhsj311mzm
  ```

- Download and import the key from one of the key servers:

  ```
  $ gpg --keyserver hkp://pgp.mit.edu --recv-keys
  87E673AF83F6C3A0C344C8C3F44A229D4D58C245
  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 cannot 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'
```

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 through 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
  ```

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:
2.1.4 Filesystem permissions

The Weblate process needs to be able to read and write to the directory where it keeps data - `DATA_DIR`. All files within this directory should be owned and writable by the user running all Weblate processes (typically WSGI and Celery, see `Running server` and `Background tasks using Celery`).

The default configuration places them in the same tree as the Weblate sources, however you might prefer to move these to a better location such as: `/var/lib/weblate`.

Weblate tries to create these directories automatically, but it will fail when it does not have permissions to do so.

You should also take care when running `Management commands`, as they should be ran under the same user as Weblate itself is running, otherwise permissions on some files might be wrong.

In the Docker container, all files in the `/app/data` volume have to be owned by weblate user inside the container (UID 1000).

**Ayrıca bakınız:**

`Serving static files`

2.1.5 Database setup for Weblate

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

**Ayrıca bakınız:**

`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.

**Not:** Weblate uses trigram extension which has to be installed separately in some cases. Look for `postgresql-contrib` or a similarly named package.

**Ayrıca bakınız:**

`PostgreSQL notes`

Creating a database in PostgreSQL

It is usually a good idea to run Weblate in a separate database, and separate user account:

```bash
# If PostgreSQL was not installed before, set the main password
sudo -u postgres psql postgres -c "\password postgres"

# Create a database user called "weblate"
sudo -u postgres createuser --superuser --pwprompt weblate

# Create the database "weblate" owned by "weblate"
sudo -u postgres createdb -E UTF8 -O weblate weblate
```
İpucu: 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:

```python
DATABASES = {
    "default": {
        # Database engine
        "ENGINE": "django.db.backends.postgresql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Name of role to alter to set parameters in PostgreSQL, 
        # use in case role name is different than user used for authentication.
        # "ALTER_ROLE": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "database.example.com",
        # Set to empty string for default
        "PORT": ",",
    }
}
```

The database migration performs `ALTER ROLE` on database role used by Weblate. In most cases the name of the role matches username. In more complex setups the role name is different than username and you will get error about non-existing role during the database migration (psycopg2.errors.UndefinedObject: role "weblate@hostname" does not exist). This is known to happen with Azure Database for PostgreSQL, but it’s not limited to this environment. Please set `ALTER_ROLE` to change name of the role Weblate should alter during the database migration.

**MySQL and MariaDB**

İpucu: Some Weblate features will perform better with PostgreSQL. This includes searching and translation memory, which both utilize full-text features in the database and PostgreSQL implementation is superior.

Weblate can be also used with MySQL or MariaDB, please see MySQL notes and MariaDB notes for caveats using Django with those. Because of the limitations it is recommended to use PostgreSQL for new installations.

Weblate requires MySQL at least 5.7.8 or MariaDB at least 10.2.7.

Following configuration is recommended for Weblate:

- Use the `utf8mb4` charset to allow representation of higher Unicode planes (for example emojis).
- Configure the server with `innodb_large_prefix` to allow longer indices on text fields.
- Set the isolation level to `READ COMMITTED`.
- The SQL mode should be set to `STRICT_TRANS_TABLES`.

MySQL 8.x, MariaDB 10.5.x or newer have reasonable default configuration so that no server tweaking should be necessary and all what is needed can be configured on the client side.
Below is an example `/etc/my.cnf.d/server.cnf` for a server with 8 GB of RAM. These settings should be sufficient for most installs. MySQL and MariaDB have tunables that will increase the performance of your server that are considered not necessary unless you are planning on having large numbers of concurrent users accessing the system. See the various vendors documentation on those details.

It is absolutely critical to reduce issues when installing that the setting `innodb_file_per_table` is set properly and MySQL/MariaDB restarted before you start your Weblate install.

```
[mysqld]
character-set-server = utf8mb4
character-set-client = utf8mb4
collation-server = utf8mb4_unicode_ci
datadir=/var/lib/mysql
log-error=/var/log/mariadb/mariadb.log
innodb_large_prefix=1
innodb_file_format=Barracuda
innodb_file_per_table=1
innodb_buffer_pool_size=2G
sql_mode=STRICT_TRANS_TABLES
```

**Ipucu:** In case you are getting `#1071` - Specified key was too long; max key length is 767 bytes error, please update your configuration to include the `innodb` settings above and restart your install.

**Ipucu:** In case you are getting `#2006` - MySQL server has gone away error, configuring `CONN_MAX_AGE` might help.

**Weblate'i MySQL/MariaDB kullanımı için yapılandırma**

The `settings.py` snippet for MySQL and MariaDB:

```
DATABASES = {
    "default": {
        # Database engine
        "ENGINE": "django.db.backends.mysql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "127.0.0.1",
        # Set to empty string for default
        "PORT": "3306",
        # In case you wish to use additional
        # connection options
        "OPTIONS": {},
    }
}
```

You should also create the `weblate` user account in MySQL or MariaDB before you begin the install. Use the commands below to achieve that:
### 2.1.6 Other configurations

#### Configuring outgoing e-mail

Weblate sends out e-mails on various occasions - for account activation and on various notifications configured by users. For this it needs access to an SMTP server.

The mail server setup is configured using these settings: `EMAIL_HOST`, `EMAIL_HOST_PASSWORD`, `EMAIL_USE_TLS`, `EMAIL_USE_SSL`, `EMAIL_HOST_USER` and `EMAIL_PORT`. Their names are quite self-explanatory, but you can find more info in the Django documentation.

**Ipucu:** In case you get error about not supported authentication (for example `SMTP AUTH extension not supported by server`), it is most likely caused by using insecure connection and server refuses to authenticate this way. Try enabling `EMAIL_USE_TLS` in such case.

**Ayrıca bakınız:**
- Not receiving e-mails from Weblate
- Configuring outgoing e-mail in Docker container

#### Running behind reverse proxy

Several features in Weblate rely on being able to get client IP address. This includes Rate limiting, Spam protection or Denetim günlüğü.

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.

**Ayrıca bakınız:**
- Spam protection
- Rate limiting
- Denetim günlüğü
- 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`:

```python
import os

os.environ["http_proxy"] = "http://proxy.example.com:8080"
os.environ["HTTPS_PROXY"] = "http://proxy.example.com:8080"
```

**Ayrıca bakınız:**
- Proxy Environment Variables
2.1.7 Adjusting configuration

Ayrıca bakınız:

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` and `ALLOWED_HOSTS`

- **ADMINS**
  
  List of site administrators to receive notifications when something goes wrong, for example notifications on failed merges, or Django errors.

  Ayrıca bakınız: `ADMINS`

- **ALLOWED_HOSTS**
  
  You need to set this to list the hosts your site is supposed to serve. For example:

  ```python
  ALLOWED_HOSTS = ["demo.weblate.org"]
  ```

  Alternatively you can include wildcard:

  ```python
  ALLOWED_HOSTS = ["*"]
  ```

  Ayrıca bakınız: `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:

```python
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
```

Ayrıca bakınız: `Configuring the session engine, SESSION_ENGINE`

**DATABASES**

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

Ayrıca bakınız: `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.

Ayrıca bakınız: `DEBUG`

**DEFAULT_FROM_EMAIL**

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

Ayrıca bakınız: `DEFAULT_FROM_EMAIL`

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

**Ayrıca bakınız:**

SECRET_KEY

SERVER_EMAIL

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

**Ayrıca bakınız:**

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.

**Ayrıca bakınız:**

Configuration, Yetkiler listesi

2.1.9 Production setup

For a production setup you should carry out adjustments described in the following sections. The most critical settings will trigger a warning, which is indicated by an exclamation mark in the top bar if signed in as a superuser:

It is also recommended to inspect checks triggered by Django (though you might not need to fix all of them):

```
weblate check --deploy
```

You can also review the very same checklist from the Management interface.

**Ayrıca bakınız:**

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.

**Ayrıca bakınız:**

 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"),
]
```

**Ayrıca bakınız:**

 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.

4.2 sürümünde değişti: Prior to the 4.2 release the Django sites framework was used instead, please see The “sites” framework.

**Ayrıca bakınız:**

 allowed hosts setup, correctly configure HTTPS SITE_DOMAIN, WEBLATE_SITE_DOMAIN, ENABLEHTTPS

Correctly configure HTTPS

It is strongly recommended to run Weblate using the encrypted HTTPS protocol. After enabling it, you should set `ENABLEHTTPS` in the settings:

```
ENABLEHTTPS = True
```

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

**Ayrıca bakınız:**

 ENABLEHTTPS, 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.

**Uyarı:** Setting this incorrectly can irreversibly (for some time) break your site. Read the HTTP Strict Transport Security documentation first.

Use a powerful database engine

- Please use PostgreSQL for a production environment, see Database setup for Weblate for more info.
- Use adjacent location for running the database server, otherwise the networking performance or reliability might ruin your Weblate experience.
- Check the database server performance or tweak its configuration, for example using PGTune.

**Ayrıca bakınız:**

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",
        },
    }
}
```

**İpucu:** In case you change Redis settings for the cache, you might need to adjust them for Celery as well, see Background tasks using Celery.

**Ayrıca bakınız:**

Avatar önbellikleme, Django’s cache framework
Avatar önbellkleme

In addition to caching of Django, Weblate performs caching of avatars. It is recommended to use a separate, file-backed cache for this purpose:

```python
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "unix:///var/run/redis/redis.sock?db =0",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
            "PARSER_CLASS": "redis.connection.HiredisParser",
        },
    },
    "avatar": {
        "BACKEND": "django.core.cache.backends.filebased.FileBasedCache",
        "LOCATION": os.path.join(DATA_DIR, "avatar-cache"),
        "TIMEOUT": 604800,
        "OPTIONS": {
            "MAX_ENTRIES": 1000,
        },
    },
}
```

Ayrıca bakınız:

*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:

```python
SERVER_EMAIL = "admin@example.org"
DEFAULT_FROM_EMAIL = "weblate@example.org"
```

**Not:** 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.

Ayrıca bakınız:

*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`.

**İpucu:** On Docker container, this is available as `WEBLATE_ALLOWED_HOSTS`. 
Additionally see:

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.

Additionally see:

SECRET_KEY

---

**Ana dizin**

2.1 sürümünde değişti: 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:

```python
os.environ["HOME"] = os.path.join(BASE_DIR, "configuration")
```

**Not:** 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.

Additionally see:

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):

```python
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": [
```

(sonraki sayfaya devam)
Running maintenance tasks

For optimal performance, it is good idea to run some maintenance tasks in the background. This is now automatically done by *Background tasks using Celery* and covers following tasks:

- Configuration health check (hourly).
- Committing pending changes (hourly), see *Lazy commits* and *commit_pending*.
- Updating component alerts (daily).
- Update remote branches (nightly), see *AUTO_UPDATE*.
- Translation memory backup to JSON (daily), see *dump_memory*.
- Fulltext and database maintenance tasks (daily and weekly tasks), see *cleanuptrans*.

3.2 sürümünde değişti: Since version 3.2, the default way of executing these tasks is using Celery and Weblate already comes with proper configuration, see *Background tasks using Celery*.

System locales and encoding

The system locales should be configured to UTF-8 capable ones. On most Linux distributions this is the default setting. In case it is not the case on your system, please change locales to UTF-8 variant.

For example by editing `/etc/default/locale` and setting there `LANG = "C.UTF-8"`.

In some cases the individual services have separate configuration for locales. This varies between distribution and web servers, so check documentation of your web server packages for that.

Apache on Ubuntu uses `/etc/apache2/envvars`:

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

Apache on CentOS uses `/etc/sysconfig/httpd` (or `/opt/rh/httpd24/root/etc/sysconfig/httpd`):

```bash
LANG='en_US.UTF-8'
```
Using custom certificate authority

Weblate does verify SSL certificates during HTTP requests. In case you are using custom certificate authority which is not trusted in default bundles, you will have to add its certificate as trusted.

The preferred approach is to do this at system level, please check your distro documentation for more details (for example on debian this can be done by placing the CA certificate into /usr/local/share/ca-certificates/ and running `update-ca-certificates`).

Once this is done, system tools will trust the certificate and this includes Git.

For Python code, you will need to configure requests to use system CA bundle instead of the one shipped with it. This can be achieved by placing following snippet to `settings.py` (the path is Debian specific):

```python
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):

```python
COMPRESSION_OFFLINE = True
```

On each deploy you need to compress the files to match current version:

```bash
weblate compress
```

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

Ayrıca bakınız:
Common Deployment Scenarios, Serving static files

2.1.10 Running server

İpucu: In case you are not experienced with services described below, you might want to try Installing using Docker.

You will need several services to run Weblate, the recommended setup consists of:

- Database server (see Database setup for Weblate)
- Cache server (see Enable caching)
- Frontend web server for static files and SSL termination (see Serving static files)
- WSGI server for dynamic content (see Sample configuration for NGINX and uWSGI)
- Celery for executing background tasks (see Background tasks using Celery)

2.1. Configuration instructions
**Not:** 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`.

**Not:** The WSGI process has to be executed under the same user the Celery process, otherwise files in the `DATA_DIR` will be stored with mixed ownership, leading to runtime issues.

See also *Filesystem permissions* and *Background tasks using Celery*.

---

**Running web server**

Running Weblate is not different from running any other Django based program. Django is usually executed as uWSGI or fcgi (see examples for different webservers below).

For testing purposes, you can use the built-in web server in Django:

```
weblate runserver
```

**Uyarı:** 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`.

**İpucu:** 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**

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`.

**Ayrıca bakınız:**

*Sample configuration for NGINX and uWSGI*, *Sample configuration for Apache*, *Sample configuration for Apache and Gunicorn*, *Compressing client assets*, *Deploying Django*, *Deploying static files*.

---

190 Bölüm 2. Administrator docs
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.

Ayrıca bakımınız:

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):

```bash
server {
    listen 80;
    server_name weblate;
    # Not used
    root /var/www/html;

    location ~^{/favicon.ico$} {
        # DATA_DIR/static/favicon.ico
        alias /home/weblate/data/static/favicon.ico;
        expires 30d;
    }

    location /static/ {  
        # DATA_DIR/static/
        alias /home/weblate/data/static/;
        expires 30d;
    }

    location /media/ { 
        # DATA_DIR/media/
        alias /home/weblate/data/media/;
        expires 30d;
    }

    location / {  
        include uwsgi_params;
        # Needed for long running operations in admin interface
        uwsgi_read_timeout 3600;
        # Adjust based to uwsgi configuration:
        uwsgi_pass unix:///run/uwsgi/app/weblate/socket;
        # uwsgi_pass 127.0.0.1:8080;
    }
}
```

Configuration for uWSGI (also available as weblate/examples/weblate.uwsgi.ini):

---

2.1. Configuration instructions 191
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your setup.

[uwsgi]
plugins = python3
master = true
protocol = uwsgi
socket = 127.0.0.1:8080
wsgi-file = /home/weblate/weblate-env/lib/python3.9/site-packages/weblate/wsgi.py

# Add path to Weblate checkout if you did not install
# Weblate by pip
# python-path = /path/to/weblate

# In case you’re using virtualenv uncomment this:
virtualenv = /home/weblate/weblate-env

# Needed for OAuth/OpenID
buffer-size = 8192

# Reload when consuming too much of memory
reload-on-rss = 250

# Increase number of workers for heavily loaded sites
workers = 8

# Enable threads for Sentry error submission
enable-threads = true

close-on-exec = true

# Avoid default 0000 umask
umask = 0022

# Run as weblate user
uid = weblate
gid = weblate

# Enable harakiri mode (kill requests after some time)
# harakiri = 3600
# harakiri-verbose = true

# Enable uWSGI stats server
# stats = :1717
# stats-http = true

# Do not log some errors caused by client disconnects
ignore-sigpipe = true
ignore-write-errors = true
disable-write-exception = true

Ayrıca bakınız:
How to use Django with uWSGI
Sample configuration for Apache

It is recommended to use prefork MPM when using WSGI with Weblate.

The following configuration runs Weblate as WSGI, you need to have enabled mod_wsgi (available as weblate/examples/apache.conf):

```plaintext
# # 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...
#
<VirtualHost *:80>
  ServerAdmin admin@weblate.example.org
  ServerName weblate.example.org

  # DATA_DIR/static/favicon.ico
  Alias /favicon.ico /home/weblate/data/static/favicon.ico

  # DATA_DIR/static/
  Alias /static/ /home/weblate/data/static/
  <Directory /home/weblate/data/static/>
    Require all granted
  </Directory>

  # DATA_DIR/media/
  Alias /media/ /home/weblate/data/media/
  <Directory /home/weblate/data/media/>
    Require all granted
  </Directory>

  # Path to your Weblate virtualenv
  WSGIDaemonProcess weblate python-home =/home/weblate/weblate-env user=weblate
  WSGIProcessGroup weblate
  WSGIApplicationGroup %{GLOBAL}

  WSGIScriptAlias / /home/weblate/weblate-env/lib/python3.7/site-packages/
  weblate/wsgi.py process-group =weblate request-timeout =600
  WSGIPassAuthorization On

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

</VirtualHost>
```

**Not:** 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`.

**Ayrıca bakınız:**
*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
    Alias /favicon.ico /home/weblate/data/static/favicon.ico
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>
    Alias /static/ /home/weblate/data/static/
    Require all granted
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/https_cert.cert
    SSLCertificateKeyFile /etc/apache2/ssl/https_key.pem
    SSLProxyEngine On
    ProxyPass /favicon.ico /
    ProxyPass /static/ /
    ProxyPass /media/ /
    ProxyPass / http://localhost:8000/
    ProxyPassReverse / http://localhost:8000/
    ProxyPreserveHost On
</VirtualHost>
```

Ayrıca bakınız:
How to use Django with Gunicorn

Running Weblate under path

1.3 sürümünde geldi.

It is recommended to use prefork MPM when using WSGI with Weblate.

A sample Apache configuration to serve Weblate under /weblate. Again using mod_wsgi (also available as weblate/examples/apache-path.conf):

```
# VirtualHost for Weblate, running under /weblate path
```

(sonraki sayfaya devam)
# 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...
<VirtualHost *:80>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org
    # DATA_DIR/static/favicon.ico
    Alias /weblate/favicon.ico /home/weblate/data/static/favicon.ico
    # DATA_DIR/static/
    Alias /weblate/static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>
    # DATA_DIR/media/
    Alias /weblate/media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>
    # Path to your Weblate virtualenv
    WSGIDaemonProcess weblate python-home =/home/weblate/weblate-env user=weblate
    WSGIProcessGroup weblate
    WSGIApplicationGroup %{GLOBAL}
    WSGIScriptAlias /weblate /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/wsgi.py
    WSGIPassAuthorization On
    <Directory /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/>
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>
</VirtualHost>

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

```
URL_PREFIX = ""/weblate"
```

## 2.1.11 Background tasks using Celery

3.2 sürümünde geldi.

Weblate uses Celery to process background tasks. A typical setup using Redis as a backend looks like this:

```
CELERY_TASK_ALWAYS_EAGER = False
CELERY_BROKER_URL = "redis://localhost:6379"
CELERY_RESULT_BACKEND = CELERY_BROKER_URL
```

Ayrıca bakınız:
Redis broker configuration in Celery

For development, you might want to use eager configuration, which does process all tasks in place, but this will have performance impact on Weblate:
CELERY_TASK_ALWAYS_EAGER = True
CELERY_BROKER_URL = "memory://"
CELERY_TASK_EAGER_PROPAGATES = True

You should also start the Celery worker to process the tasks and start scheduled tasks, this can be done directly on the command line (which is mostly useful when debugging or developing):

./weblate/examples/celery start
./weblate/examples/celery stop

Not: The Celery process has to be executed under the same user as the WSGI process, otherwise files in the DATA_DIR will be stored with mixed ownership, leading to runtime issues.

See also Filesystem permissions and Running server.

Running Celery as system service

Most likely you will want to run Celery as a daemon and that is covered by Daemonization. For the most common Linux setup using systemd, you can use the example files shipped in the examples folder listed below.

Systemd unit to be placed as /etc/systemd/system/celery-weblate.service:

```ini
[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) 
-A $(CELERY_APP) --pidfile=$(CELERYD_PID_FILE) 
--logfile=$(CELERYD_LOG_FILE) --loglevel=$(CELERYD_LOG_LEVEL) $(CELERYD_OPTS)'
ExecReload=/bin/sh -c '$(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:

```ini
# Name of nodes to start
CELERYD_NODES="celery notify memory backup translate"

# Absolute or relative path to the 'celery' command:
CELERY_BIN="/home/weblate/weblate-env/bin/celery"

# App instance to use
# comment out this line if you don't use an app
CELERY_APP="weblate.utils"
```

(sonraki sayfaya devam)
# Extra command-line arguments to the worker,
# increase concurrency if you get weblate.E019
CELERYD_OPTS="--beat:celery --queues:celery --prefetch-
  --multiplier:celery =4 \ 
  --queues:notify --prefetch-multiplier:notify =10 \ 
  --queues:memory --prefetch-multiplier:memory =10 \ 
  --queues:translate --prefetch-multiplier:translate =4 \ 
  --concurrency:backup =1 --queues:backup --backup \ 
  --prefetch-multiplier:backup =2 \`

# Logging configuration
# - %n will be replaced with the first part of the nodename.
# - %I will be replaced with the current child process index
#   and is important when using the prefork pool to avoid race conditions.
CELERYD_PID_FILE="/run/celery/weblate-%n.pid"
CELERYD_LOG_FILE="/var/log/celery/weblate-%n%I.log"
CELERYD_LOG_LEVEL="INFO"

# Internal Weblate variable to indicate we're running inside Celery
CELERY_WORKER_RUNNING="1"

Additional configuration to rotate Celery logs using `logrotate` to be placed as `/etc/logrotate.d/celery`:

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

## Periodic tasks using Celery beat

Weblate comes with built-in setup for scheduled tasks. You can however define additional tasks in `settings.py`, for example see `Lazy commits`.

The tasks are supposed to be executed by Celery beats daemon. In case it is not working properly, it might not be running or its database was corrupted. Check the Celery startup logs in such case to figure out root cause.

## Monitoring Celery status

You can use `celery_queues` to see current length of Celery task queues. In case the queue will get too long, you will also get configuration error in the admin interface.

**Uyarı:** The Celery errors are by default only logged into Celery log and are not visible to user. In case you want to have overview on such failures, it is recommended to configure `Collecting error reports`.

**Ayrıca bakınız:**
2.1.12 Monitoring Weblate

Weblate provides the /healthz/ URL to be used in simple health checks, for example using Kubernetes. The Docker container has built-in health check using this URL.

For monitoring metrics of Weblate you can use GET /api/metrics/ API endpoint.

Ayrıca bakınız:
Munin için Weblate eklentisi

2.1.13 Collecting error reports

Weblate, as any other software, can fail. In order to collect useful failure states we recommend to use third party services to collect such information. This is especially useful in case of failing Celery tasks, which would otherwise only report error to the logs and you won’t get notified on them. Weblate has support for the following services:

Sentry

Weblate has built-in support for Sentry. To use it, it’s enough to set SENTRY_DSN in the settings.py:

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

Rollbar

Weblate has built-in support for Rollbar. To use it, it’s enough to follow instructions for Rollbar notifier for Python.

In short, you need to adjust settings.py:

```python
# Add rollbar as last middleware:
MIDDLEWARE = [
    # ... other middleware classes ...
    "rollbar.contrib.django.middleware.RollbarNotifierMiddleware",
]

# Configure client access
ROLLBAR = {
    "access_token": "POST_SERVER_ITEM_ACCESS_TOKEN",
    "client_token": "POST_CLIENT_ITEM_ACCESS_TOKEN",
    "environment": "development" if DEBUG else "production",
    "branch": "main",
    "root": "/absolute/path/to/code/root",
}
```

Everything else is integrated automatically, you will now collect both server and client side errors.

2.1.14 Migrating Weblate to another server

Migrating Weblate to another server should be pretty easy, however it stores data in few locations which you should migrate carefully. The best approach is to stop Weblate for the migration.
Migrating database

Depending on your database backend, you might have several options to migrate the database. The most straightforward one is to dump the database on one server and import it on the new one. Alternatively you can use replication in case your database supports it.

The best approach is to use database native tools, as they are usually the most effective (e.g. `mysqldump` or `pg_dump`). If you want to migrate between different databases, the only option might be to use Django management to dump and import the database:

```
# Export current data
weblate dumpdata > /tmp/weblate.dump
# Import dump
weblate loaddata /tmp/weblate.dump
```

Migrating VCS repositories

The VCS repositories stored under `DATA_DIR` need to be migrated as well. You can simply copy them or use `rsync` to do the migration more effectively.

Other notes

Don’t forget to move other services Weblate might have been using like Redis, Cron jobs or custom authentication backends.

2.2 Weblate deployments

Weblate can be easily installed in your cloud. Please find detailed guide for your platform:

- Installing using Docker
- Installing on OpenShift
- Installing on Kubernetes

2.2.1 Third-party deployments for Weblate

**Not:** Following deployments are not developed or supported by Weblate team. Parts of the setup might vary from what is described in this documentation.

Bitnami Weblate stack

Bitnami provides a Weblate stack for many platforms at <https://bitnami.com/stack/weblate>. The setup will be adjusted during installation, see <https://bitnami.com/stack/weblate/README.txt> for more documentation.
Weblate Cloudron Paketi

Cloudron is a platform for self-hosting web applications. Weblate installed with Cloudron will be automatically kept up-to-date. The package is maintained by the Cloudron team at their Weblate package repo.

Weblate in YunoHost

The self-hosting project YunoHost provides a package for Weblate. Once you have your YunoHost installation, you may install Weblate as any other application. It will provide you with a fully working stack with backup and restoration, but you may still have to edit your settings file for specific usages.

You may use your administration interface, or this button (it will bring you to your server):

It also is possible to use the commandline interface:

```
yunohost app install https://github.com/YunoHost-Apps/weblate_ynh
```

2.3 Upgrading Weblate

2.3.1 Docker image upgrades

The official Docker image (see Installing using Docker) has all upgrade steps integrated. There are no manual step besides pulling latest version.

2.3.2 Generic upgrade instructions

Before upgrading, please check the current Software requirements as they might have changed. Once all requirements are installed or updated, please adjust your settings.py to match changes in the configuration (consult settings_example.py for correct values).

Always check Version specific instructions before upgrade. In case you are skipping some versions, please follow instructions for all versions you are skipping in the upgrade. Sometimes it’s better to upgrade to some intermediate version to ensure a smooth migration. Upgrading across multiple releases should work, but is not as well tested as single version upgrades.

**Not:** 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 Backing up and moving 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:
3. Upgrade configuration file, refer to settings_example.py or Version specific instructions for needed steps.

4. Upgrade database structure:

   ```bash
   weblate migrate --noinput
   ```

5. Collect updated static files (see Running server and Serving static files):

   ```bash
   weblate collectstatic --noinput
   ```

6. Compress JavaScript and CSS files (optional, see Compressing client assets):

   ```bash
   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:

   ```bash
   weblate compilemessages
   ```

8. Verify that your setup is sane (see also Production setup):

   ```bash
   weblate check --deploy
   ```

9. Restart Celery worker (see Background tasks using Celery).

### 2.3.3 Version specific instructions

**Upgrade from 2.x**

If you are upgrading from 2.x release, always first upgrade to 3.0.1 and then continue upgrading in the 3.x series. Upgrades skipping this step are not supported and will break.

**Ayrıca bakınız:**
Upgrade from 2.20 to 3.0 in Weblate 3.0 documentation

**Upgrade from 3.x**

If you are upgrading from 3.x release, always first upgrade to 4.0.4 or 4.1.1 and then continue upgrading in the 4.x series. Upgrades skipping this step are not supported and will break.

**Ayrıca bakınız:**
Upgrade from 3.11 to 4.0 in Weblate 4.0 documentation
Upgrade from 4.0 to 4.1

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- There are several changes in `settings_example.py`, most notable middleware changes, please adjust your settings accordingly.
- There are new file formats, you might want to include them in case you modified the `WEBLATE_FORMATS`.
- There are new quality checks, you might want to include them in case you modified the `CHECK_LIST`.
- There is change in `DEFAULT_THROTTLE_CLASSES` setting to allow reporting of rate limiting in the API.
- There are some new and updated requirements.
- There is a change in `INSTALLED_APPS`.
- The `MT_DEEPL_API_VERSION` setting has been removed in Version 4.7. The DeepL machine translation now uses the new `MT_DEEPL_API_URL` instead. You might need to adjust `MT_DEEPL_API_URL` to match your subscription.

Ayrıca bakınız:

*Generic upgrade instructions*

Upgrade from 4.1 to 4.2

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- Upgrade from 3.x releases is not longer supported, please upgrade to 4.0 or 4.1 first.
- There are some new and updated requirements.
- There are several changes in `settings_example.py`, most notable new middleware and changed application ordering.
- The keys for JSON based formats no longer include leading dot. The strings are adjusted during the database migration, but external components might need adjustment in case you rely on keys in exports or API.
- The Celery configuration was changed to no longer use memory queue. Please adjust your startup scripts and `CELERY_TASK_ROUTES` setting.
- The Weblate domain is now configured in the settings, see `SITE_DOMAIN` (or `WEBLATE_SITE_DOMAIN`). You will have to configure it before running Weblate.
- The username and email fields on user database now should be case insensitive unique. It was mistakenly not enforced with PostgreSQL.

Ayrıca bakınız:

*Generic upgrade instructions*
Upgrade from 4.2 to 4.3

Please follow Generic upgrade instructions in order to perform update.

Notable configuration or dependencies changes:

- There are some changes in quality checks, you might want to include them in case you modified the CHECK_LIST.

- The source language attribute was moved from project to a component what is exposed in the API. You will need to update Weblate Client in case you are using it.

- The database migration to 4.3 might take long depending on number of strings you are translating (expect around one hour of migration time per 100,000 source strings).

- There is a change in INSTALLED_APPS.

- There is a new setting SESSION_COOKIE_AGE_AUTHENTICATED which complements SESSION_COOKIE_AGE.

- In case you were using hub or lab to integrate with GitHub or GitLab, you will need to reconfigure this, see GITHUB_CREDENTIALS and GITLAB_CREDENTIALS.

4.3.1 sürümünde değişti:

- The Celery configuration was changed to add memory queue. Please adjust your startup scripts and CELERY_TASK_ROUTES setting.

4.3.2 sürümünde değişti:

- The post_update method of addons now takes extra skip_push parameter.

Ayrıca bakınız:

Generic upgrade instructions

Upgrade from 4.3 to 4.4

Please follow Generic upgrade instructions in order to perform update.

Notable configuration or dependencies changes:

- There is a change in INSTALLED_APPS, weblate.configuration has to be added there.

- Django 3.1 is now required.

- In case you are using MySQL or MariaDB, the minimal required versions have increased, see MySQL and MariaDB.

4.4.1 sürümünde değişti:

- Monolingual gettext now uses both msgid and msgctxt when present. This will change identification of translation strings in such files breaking links to Weblate extended data such as screenshots or review states. Please make sure you commit pending changes in such files prior upgrading and it is recommended to force loading of affected component using loadpo.

- Increased minimal required version of translate-toolkit to address several file format issues.

Ayrıca bakınız:

Generic upgrade instructions

2.3. Upgrading Weblate
Upgrade from 4.4 to 4.5

Please follow Generic upgrade instructions in order to perform update.

Notable configuration or dependencies changes:

- The migration might take considerable time if you had big glossaries.
- Sözlükler artık normal bileşenler olarak saklanıyor.
- The glossary API is removed, use regular translation API to access glossaries.
- There is a change in INSTALLED_APPS - weblate.metrics should be added.

4.5.1 sürümünde değişti:

- There is a new dependency on the pyahocorasick module.

Ayrıca bakınız:
Generic upgrade instructions

Upgrade from 4.5 to 4.6

Please follow Generic upgrade instructions in order to perform update.

Notable configuration or dependencies changes:

- There are new file formats, you might want to include them in case you modified the WEBLATE_FORMATS.
- API for creating components now automatically uses Weblate internal URLs, see POST /api/projects/ (string:project)/components/.
- There is a change in dependencies and PASSWORD_HASHERS to prefer Argon2 for passwords hashing.

Ayrıca bakınız:
Generic upgrade instructions

Upgrade from 4.6 to 4.7

Please follow Generic upgrade instructions in order to perform update.

Notable configuration or dependencies changes:

- There are several changes in settings_example.py, most notable middleware changes (MIDDLEWARE), please adjust your settings accordingly.
- The DeepL machine translation now has a generic MT_DEEPL_API_URL setting to adapt to different subscription models more flexibly. The MT_DEEPL_API_VERSION setting is no longer used.
- Django 3.2 is now required.

Ayrıca bakınız:
Generic upgrade instructions
Upgrade from 4.7 to 4.8

Please follow *Generic upgrade instructions* in order to perform update.

There are no additional upgrade steps needed in this release.

**Ayrıca bakınız:**
*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 database than PostgreSQL, you should consider migrating to PostgreSQL as Weblate performs best with it. The following steps will guide you in migrating your data between the databases. Please remember to stop both web and Celery servers prior to the migration, otherwise you might end up with inconsistent data.

#### Creating a database in PostgreSQL

It is usually a good idea to run Weblate in a separate database, and separate user account:

```
# If PostgreSQL was not installed before, set the main password
sudo -u postgres psql postgres -c \"\password postgres\"

# Create a database user called "weblate"
sudo -u postgres createuser -D -P weblate

# Create the database "weblate" owned by "weblate"
sudo -u postgres createdb -E UTF8 -O weblate weblate
```

#### Migrating using Django JSON dumps

The simplest approach for migration is to utilize Django JSON dumps. This works well for smaller installations. On bigger sites you might want to use pgloader instead, see *Migrating to PostgreSQL using pgloader*.

1. Add PostgreSQL as additional database connection to the *settings.py*:

```
DATABASES = {
    "default": {
        # Database engine
        "ENGINE": "django.db.backends.mysql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "database.example.com",
        # Set to empty string for default
        "PORT": ",
        # Additional database options
    }
```

(sonraki sayfaya devam)
"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:

```bash
weblate migrate --database=postgresql
weblate sqlflush --database=postgresql | weblate dbshell --database=postgresql
```

3. Dump legacy database and import to PostgreSQL

```bash
weblate dumpdata --all --output weblate.json
weblate loaddata weblate.json --database=postgresql
```

4. Adjust DATABASES to use just PostgreSQL database as default, remove legacy connection.

Weblate should be now ready to run from the PostgreSQL database.

**Migrating to PostgreSQL using pgloader**

The pgloader is a generic migration tool to migrate data to PostgreSQL. You can use it to migrate Weblate database.

1. Adjust your settings.py to use PostgreSQL as a database.

2. Migrate the schema in the PostgreSQL database:

```bash
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:

```sql
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
```

(sonraki sayfaya devam)
2.3.6 Migrating from Pootle

As Webate 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 Backing up and moving Weblate

2.4.1 Automated backup using BorgBackup

3.9 sürümünde geldi.

Webate has built-in support for creating service backups using `BorgBackup`. Borg creates space-effective encrypted backups which can be safely stored in the cloud. The backups can be controlled in the management interface from the `Backups` tab.

4.4.1 sürümünde değişti: Both PostgreSQL and MySQL/MariaDB databases are included in the automated backups. The backups using Borg are incremental and Weblate is configured to keep following backups:

- 14 gün geriye dönük günlük yedekler
- Weekly backups for 8 weeks back
- Monthly backups for 6 months back
### Backup Details

**Backup service:** /tmp/ptsf/iw-4cweblate

**Backup repository credentials:**

- **/tmp/ptsf/iw-4cweblate**

**Passphrase:**

```
g615rh1izkosy(qRbuq'q'd)nRpsb2NGYwWx9QXAXc3Ht8-AgsJ
```

The passphrase is used to encrypt the backups and is necessary to restore them.

**SSH key:**

- **Download private key**

The private key is needed to access the remote backup repository.

- **Deleted the oldest backups**
- **Aug. 21, 2021**
- **Backup performed**
- **Aug. 21, 2021**
- **Repository initialization**
- **Aug. 21, 2021**

### Activate Support Package

The support packages include priority email support, or cloud backups of your Weblate installation.

**Activation token**

Please enter the activation token obtained when making the subscription.

- **Activate**
- **Purchase support package**

### Add Backup Service

**Backup repository URL**

Use `/path/to/repo` for local backups or `user@host:/path/tt/repo` for remote SSH backups.

- **Add**
Borg encryption key

BorgBackup creates encrypted backups and you wouldn’t be able to restore them without the passphrase. The passphrase is generated when adding a new backup service and you should copy it and keep it in a secure place.

If you are using Weblate provisioned backup storage, please backup your private SSH key too, as it’s used to access your backups.

Ayrıca bakınız:

borg init

2.4.2 Weblate provisioned backup storage

The easiest way of backing up your Weblate instance is purchasing the backup service at weblate.org. This is how you get it running:

1. Purchase the Backup service on https://weblate.org/support/#backup.
2. Enter the obtained key in the management interface, see Integrating support.
3. Weblate connects to the cloud service and obtains access info for the backups.
4. Turn on the new backup configuration from the Backups tab.
5. Yedekleri geri yükleyebilmek için Borg kimlik bilgilerini yedekleyin, bkz: Borg encryption key.

İpucu: The manual step of turning everything on is there for your safety. Without your consent no data is sent to the backup repository obtained through the registration process.

2.4.3 Using custom backup storage

You can also use your own storage for the backups. SSH can be used to store backups in the remote destination, the target server needs to have BorgBackup installed.

Ayrıca bakınız:

General in the Borg documentation

Local filesystem

It is recommended to specify the absolute path for the local backup, for example /path/to/backup. The directory has to be writable by the user running Weblate (see Filesystem permissions). If it doesn’t exist, Weblate attempts to create it but needs the appropriate permissions to do so.

İpucu: When running Weblate in Docker, please ensure the backup location is exposed as a volume from the Weblate container. Otherwise the backups will be discarded by Docker upon restarting the container it is in.

One option is to place backups into an existing volume, for example /app/data/borgbackup. This is an existing volume in the container.

You can also add a new container for the backups in the Docker Compose file for example by using /borgbackup:

```
services:
  weblate:
    volumes:
    - /home/weblate/data:/app/data
    - /home/weblate/borgbackup:/borgbackup
```
The directory where backups will be stored have to be owned by UID 1000, otherwise Weblate won’t be able to write the backups there.

**Uzaktan yedeklemeler**

For creating remote backups, you will have to install BorgBackup onto another server that’s accessible for your Weblate deployment via SSH using the Weblate SSH key:

1. Prepare a server where your backups will be stored.
2. Install the SSH server on it (you will get it by default with most Linux distributions).
3. Install BorgBackup on that server; most Linux distributions have packages available (see Installation).
4. Choose an existing user or create a new user that will be used for backing up.
5. Add Weblate SSH key to the user so that Weblate can SSH to the server without a password (see Weblate SSH key).
6. Configure the backup location in Weblate as user@host:/path/to/backups.

**İpucu:** Weblate provisioned backup storage provides you automated remote backups without any effort.

**Ayrıca bakınız:**

Weblate SSH key

### 2.4.4 Restoring from BorgBackup

1. Restore access to your backup repository and prepare your backup passphrase.
2. List all the backups on the server using borg list REPOSITORY.
3. Restore the desired backup to the current directory using borg extract REPOSITORY::ARCHIVE.
4. Restore the database from the SQL dump placed in the backup directory in the Weblate data dir (see Dumped data for backups).
5. Copy the Weblate configuration (backups/settings.py, see Dumped data for backups) to the correct location, see Adjusting configuration.

    When using Docker container, the settings file is already included in the container and you should restore the original environment variables. The environment.yml file might help you with this (see Dumped data for backups).

6. Copy the whole restored data dir to the location configured by DATA_DIR.

    When using Docker container place the data into the data volume, see Docker container volumes.

    Please make sure the files have correct ownership and permissions, see Filesystem permissions.

The Borg session might look like this:

```bash
$ 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:
```

**Ayrıca bakınız:**

borg list, borg extract
2.4.5 Manual backup

Depending on what you want to save, back up the type of data Weblate stores in each respective place.

**İpucu:** If you are doing the manual backups, you might want to silence Weblate’s warning about a lack of backups by adding `weblate.I028` to `SILENCED_SYSTEM_CHECKS` in `settings.py` or `WEBLATE_SILENCED_SYSTEM_CHECKS` for Docker.

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

Database

The actual storage location depends on your database setup.

**İpucu:** The database is the most important storage. Set up regular backups of your database. Without the database, all the translations are gone.

Native database backup

The recommended approach is to save a dump of the database using database-native tools such as `pg_dump` or `mysqldump`. It usually performs better than Django backup, and it restores complete tables with all their data.

You can restore this backup in a newer Weblate release, it will perform all the necessary migrations when running in `migrate`. Please consult *Upgrading Weblate* on more detailed info on how to upgrade between versions.

Django database backup

Alternatively, you can back up your database using Django’s `dumpdata` command. That way the backup is database agnostic and can be used in case you want to change the database backend.

Prior to restoring the database you need to be running exactly the same Weblate version the backup was made on. This is necessary as the database structure does change between releases and you would end up corrupting the data in some way. After installing the same version, run all database migrations using `migrate`.

Afterwards some entries will already be created in the database and you will have them in the database backup as well. The recommended approach is to delete such entries manually using the management shell (see *Invoking management commands*):

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

Dosyalar

If you have enough backup space, simply back up the whole `DATA_DIR`. This is a safe bet even if it includes some files you don’t want. The following sections describe what you should back up and what you can skip in detail.
Dumped data for backups

4.7 sürümünde değişti: The environment dump was added as `environment.yml` to help in restoring in the Docker environments.

Stored in `DATA_DIR/backups`.

Weblate dumps various data here, and you can include these files for more complete backups. The files are updated daily (requires a running Celery beats server, see Background tasks using Celery). Currently, this includes:

- Weblate settings as `settings.py` (there is also expanded version in `settings-expanded.py`).
- PostgreSQL database backup as `database.sql`.
- Environment dump as `environment.yml`.

The database backups are saved as plain text by default, but they can also be compressed or entirely skipped using `DATABASE_BACKUP`.

To restore the database backup load it using database tools, for example:

```
psql --file=database.sql weblate
```

Version control repositories

Stored in `DATA_DIR/vcs`.

The version control repositories contain a copy of your upstream repositories with Weblate changes. If you have İşlemeye yolla enabled for all your translation components, all Weblate changes are included upstream. No need to back up the repositories on the Weblate side as they can be cloned again from the upstream location(s) with no data loss.

SSH and GPG keys

Stored in `DATA_DIR/ssh` and `DATA_DIR/home`.

If you are using SSH or GPG keys generated by Weblate, you should back up these locations. Otherwise you will lose the private keys and you will have to regenerate new ones.

User uploaded files

Stored in `DATA_DIR/media`.

You should back up all user uploaded files (e.g. Visual context for strings).

Celery tasks

The Celery task queue might contain some info, but is usually not needed for a backup. At most you will lose updates not yet been processed to translation memory. It is recommended to perform the fulltext or repository update upon restoration anyhow, so there is no problem in losing these.

Ayrıca bakınız:

Background tasks using Celery
**Command line for manual backup**

Using a cron job, you can set up a Bash command to be executed on a daily basis, for example:

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

The string between the quotes after `XZ_OPT` allows you to choose your xz options, for instance the amount of memory used for compression; see https://linux.die.net/man/1/xz

You can adjust the list of folders and files to your needs. To avoid saving the translation memory (in backups folder), you can use:

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

### 2.4.6 Restoring manual backup

1. Restore all data you have backed up.
2. Update all repositories using `updatetgit`.

```
weblate updatetgit --all
```

### 2.4.7 Moving a Weblate installation

Relocate your installation to a different system by following the backing up and restoration instructions above.

**Ayrıca bakınız:**

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

### 2.5 Kimlik doğrulama

#### 2.5.1 Kullanıcı kaydı

The default setup for Weblate is to use python-social-auth, a form on the website to handle registration of new users. After confirming their e-mail a new user can contribute or authenticate by using one of the third party services.

You can also turn off registration of new users using `REGISTRATION_OPEN`.

The authentication attempts are subject to `Rate limiting`.

#### 2.5.2 Authentication backends

The built-in solution of Django is used for authentication, including various social options to do so. Using it means you can import the user database of other Django-based projects (see *Migrating from Pootle*).

Django can additionally be set up to authenticate against other means too.

**Ayrıca bakınız:**

*Authentication settings* describes how to configure authentication in the official Docker image.
2.5.3 Social authentication

Thanks to Welcome to Python Social Auth’s documentation!, Weblate support authentication using many third party services such as GitLab, Ubuntu, Fedora, etc.

Please check their documentation for generic configuration instructions in Django Framework.

**Not:** By default, Weblate relies on third-party authentication services to provide a validated e-mail address. If some of the services you want to use don’t support this, please enforce e-mail validation on the Weblate side by configuring `FORCE_EMAIL_VALIDATION` for them. For example:

```python
SOCIAL_AUTH_OPENSUSE_FORCE_EMAIL_VALIDATION = True
```

Ayrıca bakınız:

Pipeline

Enabling individual backends is quite easy, it’s just a matter of adding an entry to the `AUTHENTICATION_BACKENDS` setting and possibly adding keys needed for a given authentication method. Please note that some backends do not provide user e-mail by default, you have to request it explicitly, otherwise Weblate will not be able to properly credit contributions users make.

İpucu: Most of the authentication backends require HTTPS. Once HTTPS is enabled in your web server please configure Weblate to report it properly using `ENABLE_HTTPS`, or by `WEBLATE_ENABLE_HTTPS` in the Docker container.

Ayrıca bakınız:

Python Social Auth backend

OpenID authentication

For OpenID-based services it’s usually just a matter of enabling them. The following section enables OpenID authentication for OpenSUSE, Fedora and Ubuntu:

```python
# 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",
)
```

Ayrıca bakınız:

OpenID
GitHub authentication

You need to register an OAuth application on GitHub and then tell Weblate all its secrets:

```
# 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"]
```

The GitHub should be configured to have callback URL as https://example.com/accounts/complete/github/.

**Not:** Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see **Set correct site domain.**

Ayrıca bakınız:

GitHub

Bitbucket authentication

You need to register an application on Bitbucket and then tell Weblate all its secrets:

```
# 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
```

**Not:** Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see **Set correct site domain.**

Ayrıca bakınız:

Bitbucket
Google OAuth 2

To use Google OAuth 2, you need to register an application on <https://console.developers.google.com/> and enable the Google+ API.

The redirect URL is https://WEBLATE_SERVER/accounts/complete/google-oauth2/

```python
# 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"
```

**Note:** Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see *Set correct site domain.*

Ayrıca bakınız:

Google

Facebook OAuth 2

As per usual with OAuth 2 services, you need to register your application with Facebook. Once this is done, you can set up Weblate to use it:

The redirect URL is https://WEBLATE_SERVER/accounts/complete/facebook/

```python
# 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"]
```

**Note:** Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see *Set correct site domain.*

Ayrıca bakınız:

Facebook
GitLab OAuth 2

For using GitLab OAuth 2, you need to register an application on <https://gitlab.com/profile/applications>. The redirect URL is https://WEBLATE SERVER/accounts/complete/gitlab/ and ensure you mark the read_user scope.

```python
# 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/'
```

**Note:** Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see *Set correct site domain*.

**Ayrıca bakınız:**

GitLab

Microsoft Azure Active Directory

Weblate can be configured to use common or specific tenants for authentication.


```python
# 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
```

(sonraki sayfaya devam)
Social Authentication Configuration

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

**Not:** Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see *Set correct site domain.*

**Ayrıca baktınız:**

Microsoft Azure Active Directory

### Slack

For using Slack OAuth 2, you need to register an application on <https://api.slack.com/apps>.

The redirect URL is `https://WEBLATE SERVER/accounts/complete/slack/`

```python
# 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 = ""
```

**Not:** Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see *Set correct site domain.*

**Ayrıca baktınız:**

Slack

### Overriding authentication method names and icons

You can override the authentication method display name and icon using settings as `SOCIAL_AUTH_<NAME>_IMAGE` and `SOCIAL_AUTH_<NAME>_TITLE`. For example, overriding naming for Auth0 would look like:

```python
SOCIAL_AUTH_AUTH0_IMAGE = "custom.svg"
SOCIAL_AUTH_AUTH0_TITLE = "Custom auth"
```
Turning off password authentication

E-mail and password authentication can be turned off by removing `social_core.backends.email.EmailAuth` from `AUTHENTICATION_BACKENDS`. Always keep `weblate.accounts.auth.WeblateUserBackend` there, it is needed for core Weblate functionality.

**Tıyo:** You can still use password authentication for the admin interface, for users you manually create there. Just navigate to `/admin/`.

For example authentication using only the openSUSE Open ID provider can be achieved using the following:

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

### 2.5.4 Password authentication

The default settings.py comes with a reasonable set of `AUTH_PASSWORD_VALIDATORS`:

- Passwords can’t be too similar to your other personal info.
- Passwords must contain at least 10 characters.
- Passwords can’t be a commonly used password.
- Passwords can’t be entirely numeric.
- Passwords can’t consist of a single character or only whitespace.
- Passwords can’t match a password you have used in the past.

You can customize this setting to match your password policy.

Additionally you can also install `django-zxcvbn-password` which gives quite realistic estimates of password difficulty and allows rejecting passwords below a certain threshold.

### 2.5.5 SAML authentication

4.1.1 sürümünde geldi.

Please follow the Python Social Auth instructions for configuration. Notable differences:

- Weblate supports single IDP which has to be called `weblate` in `SOCIAL_AUTH_SAML_ENABLED_IDPS`.
- The SAML XML metadata URL is `/accounts/metadata/saml/`.
- Following settings are automatically filled in: `SOCIAL_AUTH_SAML_SP_ENTITY_ID`, `SOCIAL_AUTH_SAML_TECHNICAL_CONTACT`, `SOCIAL_AUTH_SAML_SUPPORT_CONTACT`

Example configuration:

```python
# 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_ENTITY_ID = f"https://{SITE_DOMAIN}/accounts/metadata/saml/"
```

2.5. Kimlik doğrulama
The default configuration extracts user details from following attributes, configure your IDP to provide them:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>SAML URI reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad soyad</td>
<td>urn:oid:2.5.4.3</td>
</tr>
<tr>
<td>Ad</td>
<td>urn:oid:2.5.4.42</td>
</tr>
<tr>
<td>Last name</td>
<td>urn:oid:2.5.4.4</td>
</tr>
<tr>
<td>E-posta</td>
<td>urn:oid:0.9.2342.19200300.100.1.3</td>
</tr>
<tr>
<td>Kullanıcı adı</td>
<td>urn:oid:0.9.2342.19200300.100.1.1</td>
</tr>
</tbody>
</table>

İpucu: The example above and the Docker image define an IDP labelled `weblate`. You might need to configure this string as `Relay` in your IDP.

Ayrıca bakınız:

*Configuring SAML in Docker, SAML*

## 2.5.6 LDAP authentication

LDAP authentication can be best achieved using the `django-auth-ldap` package. You can install it via usual means:

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

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

İpucu: This package is included in the Docker container, see `Installing using Docker`. 
Not: There are some incompatibilities in the Python LDAP 3.1.0 module, which might prevent you from using that version. If you get error `AttributeError: 'module' object has no attribute '_trace_level'`, downgrading python-ldap to 3.0.0 might help.

Once you have the package installed, you can hook it into the Django authentication:

```python
# Add LDAP backed, keep Django one if you want to be able to sign in
# even without LDAP for admin account
AUTHENTICATION_BACKENDS = (
    "django_auth_ldap.backend.LDAPBackend",
    "weblate.accounts.auth.WeblateUserBackend",
)

# LDAP server address
AUTH_LDAP_SERVER_URI = "ldaps://ldap.example.net"

# DN to use for authentication
AUTH_LDAP_USER_DN_TEMPLATE = "cn =%(user)s,o =Example"
# Depending on your LDAP server, you might use a different DN
# like:
# AUTH_LDAP_USER_DN_TEMPLATE = 'ou =users,dc =example,dc =com'

# List of attributes to import from LDAP upon sign in
# Weblate stores full name of the user in the full_name attribute
AUTH_LDAP_USER_ATTR_MAP = {
    "full_name": "name",
    # Use the following if your LDAP server does not have full name
    # Weblate will merge them later
    # 'first_name': 'givenName',
    # 'last_name': 'sn',
    # Email is required for Weblate (used in VCS commits)
    "email": "mail",
}

# Hide the registration form
REGISTRATION_OPEN = False
```

Not: You should remove 'social_core.backends.email.EmailAuth' from the AUTHENTICATION_BACKENDS setting, otherwise users will be able to set their password in Weblate, and authenticate using that. Keeping 'weblate.accounts.auth.WeblateUserBackend' is still needed in order to make permissions and facilitate anonymous users. It will also allow you to sign in using a local admin account, if you have created it (e.g. by using `createadmin`).

**Using bind password**

If you cannot use direct bind for authentication, you will need to use search, and provide a user to bind for the search. For example:

```python
import ldap
from django_auth_ldap.config import LDAPSearch

AUTH_LDAP_BIND_DN = ""
AUTH_LDAP_BIND_PASSWORD = ""
AUTH_LDAP_USER_SEARCH = LDAPSearch(
    "ou =users,dc =example,dc =com", ldap.SCOPE_SUBTREE, "(uid =%(user)s)"
)
```
Active Directory bütünleştirmesi

```python
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": "CN =weblate_AdminUsers,OU =Groups,DC =example,DC =com",
}

# Map groups from AD to Weblate
AUTH_LDAP_GROUP_SEARCH = LDAPSearch("OU =Groups,DC =example,DC =com", ldap.SCOPE_SUBTREE, "(objectClass =group)"

AUTH_LDAP_GROUP_TYPE = NestedActiveDirectoryGroupType()
AUTH_LDAP_FIND_GROUP_PERMS = True

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

Ayrıca bakınız:

Django Authentication Using LDAP, Authentication

### 2.5.7 CAS kimlik doğrulaması

CAS authentication can be achieved using a package such as `django-cas-ng`.

Step one is disclosing the e-mail field of the user via CAS. This has to be configured on the CAS server itself, and requires you run at least CAS v2 since CAS v1 doesn’t support attributes at all.

Step two is updating Weblate to use your CAS server and attributes.

To install `django-cas-ng`:

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

Once you have the package installed you can hook it up to the Django authentication system by modifying the `settings.py` file:

```python
# 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")
```
Finally, a signal can be used to map the e-mail field to the user object. For this to work you have to import the signal from the `django-cas-ng` package and connect your code with this signal. Doing this in settings file can cause problems, therefore it’s suggested to put it:

- In your app config’s `django.apps.AppConfig.ready()` method
- In the project’s `urls.py` file (when no models exist)

```python
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.get("email")
    user.save()
```

Ayrıca bakınız:

Django CAS NG

### 2.5.8 Configuring third party Django authentication

Generally any Django authentication plugin should work with Weblate. Just follow the instructions for the plugin, just remember to keep the Weblate user backend installed.

Ayrıca bakınız:

*LDAP authentication*, *CAS kimlik doğrulaması*

Typically the installation will consist of adding an authentication backend to `AUTHENTICATION_BACKENDS` and installing an authentication app (if there is any) into `INSTALLED_APPS`:

```python
AUTHENTICATION_BACKENDS = ({
    # Add authentication backend here
    "weblate.accounts.auth.WeblateUserBackend",
})

INSTALLED_APPS += ({
    # Install authentication app here
})
```

### 2.6 Erişim denetimi

Weblate, tüm örnek için veya sınırlı bir kapsamında kullanıcı izinlerini atamak için ayrıntılı bir yetki sistemiyle birlikte gelmektedir.

3.0 sürümünde değişti: Weblate 3.0’dan önce, yetki sistemleri yalnızca Django ayrıcalık sistemine dayanıyordu, ancak şimdi özellikle Weblate için oluşturuldu. Daha eski bir şey kullanıyorsanız, lütfen kullandığınız belirli sürümünüz belirli sürümü ait belgelere bakın.
2.6.1 Basit erişim kontrolü

Eğer tüm Weblate kurulumunu yönetmiyorsanız ve sadece belirli projelere erişiminiz varsa (Hosted Weblate gibi), erişim denetim yönetim seçeneklerini aşağıdaki ayarlarla kısıtlırsınız. Eğer herhangi bir karışık kuruluma ihtiyacını yoksa, bu ayarlar sizin için yeterlidir.

Proje erişim kontrolü

Not: Bu özellik, Barındırılan Weblate üzerinde Özgür planını kullanan projeler için kullanılabilir değildir.

Farklı bir Erişim denetimi ayarı seçerek kullanıcının bireysel projelere erişimini sınırlayabilirsiniz. Kullanılabilebilir seçenekler şunlardır:

- **Açık** Herkes görebilir, tüm oturum açmış kullanıcılar çevirebilir.
- **Korumalı** Herkes görebilir, ancak yalnızca seçili kullanıcılar çevirebilir.
- **Özel** Yalnızca seçili kullanıcılar görebilir ve çevirebilir.
- **Özel Kullanıcı yönetimi** özellikleri devre dışı bırakılacak; varsayılan olarak tüm kullanıcıların proje üzerinde herhangi bir işlem yapması yasaktır. Tüm izinleri Custom access control kullanarak ayarlanmanız gerekecektir.

Erişim denetimi, ilgili her projenin yapılandırmasına (Yönet ↓ Ayarlar) Erişim sekmesinde değiştirilebilir.

Varsayılan değer ::ayarlar:`VARSAYILAN_ERİŞİM_KONTROLÜ` ile değiştirilebilir.


**Uyarı:** ‘Özel’ erişim kontrolünü açarak, Weblate seçilen bir proje için oluşturduğu tüm :ref:`özel <manage-acl>` gruplarını kaldıracaktır. Bu, tüm Weblate örneği için yönetici izni olmadan yapıyorsanız, projeyi yönetme erişiminizi anında kaybedersiniz.

**Ayrıca bakınız:**

**Erişim denetimi**

**Proje başına erişim denetimini yönetme**

Proje erişimini yönet amacıyla sahip kullanıcılar (bkz. ayrıca açıklıklar), projelerdeki kullanıcıları Özel olmayan erişim kontrolü ile yönetebilir. Kullanıcıları aşağıdaki gruplardan birine atayabilirler.

“Genel”, “Korumalı” ve “Özel” projeler için:

**Yönetim** Proje için kullanılabilen tüm izinleri içerir.

**Gözden geçir (yalnızca inceleme iş akışı açıksa)** Gözden geçirme sırasında çevirileri onaylayabilir.

Yalnızca “Korumalı” ve “Özel” projeler için:

**Çevir** Projeyi çevirebilir ve çevrim dışı yapılan çevirileri yükleyebilir.

**Kaynaklar** *(project settings içinde izin verilirse)* kaynak dizgileri ve kaynak dizgi bilgilerini düzenleyebilir.

**Diller** Çevilen dilleri yönetebilir (çeviri ekleyebilir veya kaldıatabilir).

**Sözlük** Sözlük yönetebilir (girişler ekleyebilir veya kaldıatabilir, ayrıca yükleyebilir).

**Bellek** Çeviri belleğini yönetebilir.

**Ekran görüntüleri** Ekran görüntülerini yönetebilir (ekleyip kaldıatabilir ve kaynak dizgilerle ilişkilendirebilir).

**VCS** VCS’yi yönetebilir ve dışa aktarılan depoya erişebilir.

**Faturalandırma** Faturalandırma bilgilerine ve ayarlarına erişebilir (bkz: *Faturalandırma*).

Ne yazık ki, bu önceden tanımlanmış grup setini değiştirmek şu an için mümkün değil. Ayrıca bu şekilde tüm kullanıcılar sadece bazı ek izinler vermek mümkün değildir.

**Not:** For non-*Custom* access control an instance of each group described above is actually defined for each project. The actual name of those groups will be Project@Group, also displayed in the Django admin interface this way. Although they can’t be edited from Weblate user-interface.
These features are available on the Access control page, which can be accessed from the project’s menu Manage ↓ Users.

Yeni kullanııcı davetiyesi

Also, besides adding an existing user to the project, it is possible to invite new ones. Any new user will be created immediately, but the account will remain inactive until signing in with a link in the invitation sent via an e-mail. It is not required to have any site-wide privileges in order to do so, access management permission on the project’s scope (e.g. a membership in the Administration group) would be sufficient.

İpucu: If the invited user missed the validity of the invitation, they can set their password using invited e-mail address in the password reset form as the account is created already.

3.11 sürümünde geldi: It is possible to resend the e-mail for user invitations (invalidating any previously sent invitation).

The same kind of invitations are available site-wide from the management interface on the Users tab.
Kullanıcıları engelleme

4.7 sürümünde geldi.

In case some users behave badly in your project, you have an option to block them from contributing. The blocked user will still be able to see the project if he has permissions for that, but he won’t be able to contribute.

Proje başına izin yönetimi

Projelerinizi Korumalı veya Özel olarak ayarlayabilir ve Weblate kullanıcı arayüzünde her proje için manage users.

By default this prevents Weblate from granting access provided by Users and Viewers default groups due to these groups’ own configuration. This doesn’t prevent you from granting permissions to those projects site-wide by altering default groups, creating a new one, or creating additional custom settings for individual component as described in Custom access control below.

One of the main benefits of managing permissions through the Weblate user interface is that you can delegate it to other users without giving them the superuser privilege. In order to do so, add them to the Administration group of the project.

2.6.2 Custom access control

Not: Bu özellik, Barındırılan Weblate üzerinde Özgür planını kullanan projeler için kullanılabılır değildir.

Izin sistemi gruplara ve rollere dayanmaktadır, roller bir dizi izin tanımlamakta ve gruplar bunları kullanıcılara ve çevirilere bağlamaktadır, daha fazla ayrıntı için Kullanıcılar, roller, gruplar ve izinler bölümüne bakın.

The most powerful features of the Weblate’s access control system for now are available only through the Django admin interface. You can use it to manage permissions of any project. You don’t necessarily have to switch it to Custom access control to utilize it. However you must have superuser privileges in order to use it.

If you are not interested in details of implementation, and just want to create a simple-enough configuration based on the defaults, or don’t have a site-wide access to the whole Weblate installation (like on Hosted Weblate), please refer to the Basit erişim kontrolü section.

Yaygın kurumlar

This section contains an overview of some common configurations you may be interested in.

Site çapında izin yönetimi

To manage permissions for a whole instance at once, add users to appropriate default groups:

- Users (this is done by default by the automatic group assignment).
- Reviewers (if you are using review workflow with dedicated reviewers).
- Managers (if you want to delegate most of the management operations to somebody else).

You should keep all projects configured as Public (see Proje erişim kontrolü), otherwise the site-wide permissions provided by membership in the Users and Reviewers groups won’t have any effect.

You may also grant some additional permissions of your choice to the default groups. For example, you may want to give a permission to manage screenshots to all the Users.

You can define some new custom groups as well. If you want to keep managing your permissions site-wide for these groups, choose an appropriate value for the Project selection (e.g. All projects or All public projects).
Diller, bileşenler veya projeler için özel izinler

You can create your own dedicated groups to manage permissions for distinct objects such as languages, components, and projects. Although these groups can only grant additional privileges, you can’t revoke any permission granted by site-wide or per-project groups by adding another custom group.

Example:

If you want (for whatever reason) to allow translation to a specific language (let's say Czech) only to a closed set of reliable translators while keeping translations to other languages public, you will have to:

1. Remove the permission to translate Czech from all the users. In the default configuration this can be done by altering the Users default group.

<table>
<thead>
<tr>
<th>Dil seçimi</th>
<th>Tanımlandığı gibi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diller</td>
<td>All but Czech</td>
</tr>
</tbody>
</table>

2. Add a dedicated group for Czech translators.

<table>
<thead>
<tr>
<th>Roller</th>
<th>Uzman kullanıcılar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proje seçimi</td>
<td>Tüm genel projeler</td>
</tr>
<tr>
<td>Dil seçimi</td>
<td>Tanımlandığı gibi</td>
</tr>
<tr>
<td>Diller</td>
<td>Çekçe</td>
</tr>
</tbody>
</table>

3. Add users you wish to give the permissions to into this group.

As you can see, permissions management this way is powerful, but can be quite a tedious job. You can’t delegate it to another user, unless granting superuser permissions.

Kullanıcılar, roller, gruplar ve izinler

The authentication models consist of several objects:

Permission Individual permission defined by Weblate. Permissions cannot be assigned to users. This can only be done through assignment of roles.

Role A role defines a set of permissions. This allows reuse of these sets in several places, making the administration easier.

User User can belong to several groups.

Group Group connect roles, users, and authentication objects (projects, languages, and component lists).
Bir projeye göz atmak için erişim

A user has to be a member of a group linked to the project, or any component inside that project. Having membership is enough, no specific permissions are needed to browse the project (this is used in the default Viewers group, see Grup listesi).

Bir bileşene göz atmak için erişim

A user can access unrestricted components once able to access the components’ project (and will have all the permissions the user was granted for the project). With Restricted access turned on, access to the component requires explicit permissions for the component (or a component list the component is in).

Grupların kapsımı

The scope of the permission assigned by the roles in the groups are applied by the following rules:

- If the group specifies any Component list, all the permissions given to members of that group are granted for all the components in the component lists attached to the group, and an access with no additional permissions is granted for all the projects these components are in. Components and Projects are ignored.

- If the group specifies any Components, all the permissions given to the members of that group are granted for all the components attached to the group, and an access with no additional permissions is granted for all the projects these components are in. Projects are ignored.

- Otherwise, if the group specifies any Projects, either by directly listing them or by having Projects selection set to a value like All public projects, all those permissions are applied to all the projects, which effectively grants the same permissions to access all projects unrestricted components.

- The restrictions imposed by a group’s Languages are applied separately, when it’s verified if a user has an access to perform certain actions. Namely, it’s applied only to actions directly related to the translation process itself like reviewing, saving translations, adding suggestions, etc.
İpucu: Use Language selection or Project selection to automate inclusion of all languages or projects.

Example:

Let's say there is a project foo with the components: foo/bar and foo/baz and the following group:

<table>
<thead>
<tr>
<th>Roller</th>
<th>Dizgileri gözden geçir, Depoya yönet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bileşenler</td>
<td>foo/bar</td>
</tr>
<tr>
<td>Diller</td>
<td>İspanyolca</td>
</tr>
</tbody>
</table>

Tablo 3: Group Spanish Admin-Reviewers

Members of that group will have following permissions (assuming the default role settings):

- General (browsing) access to the whole project foo including both components in it: foo/bar and foo/baz.
- Review strings in foo/bar Spanish translation (not elsewhere).
- Manage VCS for the whole foo/bar repository e.g. commit pending changes made by translators for all languages.

Kendiliğinden grup atamaları

On the bottom of the Group editing page in the Django admin interface, you can specify Automatic group assignments, which is a list of regular expressions used to automatically assign newly created users to a group based on their e-mail addresses. This assignment only happens upon account creation.

The most common use-case for the feature is to assign all new users to some default group. In order to do so, you will probably want to keep the default value (^.*$) in the regular expression field. Another use-case for this option might be to give some additional privileges to employees of your company by default. Assuming all of them use corporate e-mail addresses on your domain, this can be accomplished with an expression like ^.*@mycompany.com.

Not: Automatic group assignment to Users and Viewers is always recreated when upgrading from one Weblate version to another. If you want to turn it off, set the regular expression to ^$ (which won’t match anything).

Not: As for now, there is no way to bulk-add already existing users to some group via the user interface. For that, you may resort to using the REST API.

Varsayılan gruplar ve roller

After installation, a default set of groups is created (see Grup listesi).

These roles and groups are created upon installation. The built-in roles are always kept up to date by the database migration when upgrading. You can’t actually change them, please define a new role if you want to define your own set of permissions.
Yetkiler listesi

Faturalandırma (bkz: Faturalandırma) Faturalandırma bilgisini görüntüle [Yönetim, Faturalandırma]

Değişiklikler Değişiklikleri indir [Yönetim]

Yorumlar Yorum gönder [Yönetim, Kaynağı düzenle, Uzman kullanıc的认知, Çevir]

Bileşen Bileşen ayarlarını düzenleme [Yönetim]

Sözlük Sözlük girisi ekle [Yönetim, Sözlüğü yönet, Uzman kullanıc认知]

Otomatik öneriler Otomatik önerileri kullan [Yönetim, Kaynağı düzenleme, Uzman kullanıc认知, Çevir]

Çeviri Belleği Çeviri belleğini düzenleme [Yönetim, Çeviri belleğini yönet]

Projeler Proje ayarlarını düzenleme [Yönetim]

Raporlar Raporları indir [Yönetim]

Ekran görüntüler Ekran görüntüsünü ekle [Yönetim, Ekran görüntülerini yönet]

Kaynak dizgiler Ek dizgi bilgilerini düzenleme [Yönetim, Kaynağı düzenleme]

Dizgiler Yeni dizgi ekle [Yönetim]

Öneriler Öneri kabul et [Yönetim, Kaynağı düzenleme, Uzman kullanıc认知, Dizgileri gözden geçir, Çevir]

Çeviriler Çeviri için dil ekle [Yönetim, Uzman kullanıc认知, Dilleri yönet]

2.6. Erişim denetimi

Otomatik önerileri gerçekleştir [Yönetim, Dilleri yönet]

Mevcut çeviriyi sil [Yönetim, Dilleri yönet]

Çeviri için birkaç dil ekle [Yönetim, Dilleri yönet]
Yüklemeler  Yüklenen çevirinin yazarına tanımla [Yönetim]

Yükleme ile varolan dizgilerin üzerine yaz [Yönetim, Kaynaağ düzenle, Uzman kullanıcı, Dizgileri gözden geçir, Çevir]
Çevirileri yükle [Yönetim, Kaynağ düzenle, Uzman kullanıcı, Dizgileri gözden geçir, Çevir]

VCS  İç depoya eriş [Yönetim, Depoya eriş, Uzman kullanıcı, Depoyu yönet]

Değişiklikleri iç depoya işle [Yönetim, Depoyu yönet]
İç depodan değişikliği yolla [Yönetim, Depoyu yönet]
İç depodaki değişiklikleri sıfırla [Yönetim, Depoyu yönet]
Yukarı akış depo konumunu görüntüle [Yönetim, Depoya eriş, Uzman kullanıcı, Depoyu yönet]
İç depoyu güncelle [Yönetim, Depoyu yönet]

Site çapında yetkiler  Yönetimin arayüzünü kullan

Yeni projeler ekle
Dil tanımları ekle
Dil tanımlarını yönet
Grupları yönet
Kullancıları yönet
Rolleri yönet
Duyuruları yönet
Çeviri bellegini yönet
Bileşen listelerini yönet

Not:  Site çapında yetkiler herhangi bir varsayılan role verilmez. Bunlar güçlüdür ve yetkili kullanıcı (superuser) durumuna oldukça yakındır. Çoğu Weblate kurulumuzdaki tüm projelerini etkiler.

Grup listesi

Aşağıdaki gruplar kurulumun ardından (veya setupgroups çalıştırılduktan sonra) oluşturulur ve bunları istediğinizi gibi değiştirebilirsiniz. Ancak, silerseniz veya yeniden adlandırırsanız taşma işlemi onları yeniden oluşturacaktır.

Konuklar  Kimliği doğrulanmamış kullanıcılar için izinleri tanımlar.

Bu grup yalnızca anonim kullanıcıları içerir (bkz: ANONYMOUS_USER_NAME).

Kimliği doğrulanmamış kullanıcıların izinlerini sınırlamak için bu gruptaki rolleri kalıdırabilirsiniz.

Varsayılan roller: Öneri ekle, Depoya eriş

Göüntüleyiciler  Bu rol, ortak projelerin tüm kullanıcılar için görünür olduğundan emin olunmasını sağlar. Varsayılan olarak tüm kullanıcılar bu grubun üyesidir.

By default, automatic group assignment makes all new accounts members of this group when they join.

Varsayılan roller: yok

Kullancılar  Tüm kullanıcılar için varsayılan grup.

By default, automatic group assignment makes all new accounts members of this group when they join.

Varsayılan roller: Uzman kullanıcı

Gözden geçirici  Gözden geçirici için grup (bkz: Translation workflows).

Varsayılan roller: Dizgileri gözden geçir
2.6.3 Ekerişim kısıtlamaları

If you want to use your Weblate installation in a less public manner, i.e. allow new users on an invitational basis only, it can be done by configuring Weblate in such a way that only known users have an access to it. In order to do so, you can set `REGISTRATION_OPEN` to `False` to prevent registrations of any new users, and set `REQUIRE_LOGIN` to `/.*` to require logging-in to access all the site pages. This is basically the way to lock your Weblate installation.

İpucu: You can use built-in invitations to add new users.

2.7 Çeviri projeleri

2.7.1 Translation organization

Weblate organizes translatable VCS content of project/components into a tree-like structure.

- The bottom level object is `Project configuration`, which should hold all translations belonging together (for example translation of an application in several versions and/or accompanying documentation).
- On the level above, `Component configuration`, which is actually the component to translate, you define the VCS repository to use, and the mask of files to translate.
- Above `Component configuration` there are individual translations, handled automatically by Weblate as translation files (which match `File mask` defined in `Component configuration`) appear in the VCS repository.

Weblate supports a wide range of translation formats (both bilingual and monolingual ones) supported by Translate Toolkit, see `Supported file formats`.

Not: You can share cloned VCS repositories using Weblate internal URLs. Using this feature is highly recommended when you have many components sharing the same VCS. It improves performance and decreases required disk space.

2.7.2 Adding translation projects and components

3.2 sürümünde değişti: An interface for adding projects and components is included, and you no longer have to use `The Django admin interface`.

3.4 sürümünde değişti: The process of adding components is now multi staged, with automated discovery of most parameters.

Based on your permissions, new translation projects and components can be created. It is always permitted for users with the `Add new projects` permission, and if your instance uses billing (e.g. like `https://hosted.weblate.org/` see `Faturalandırma`), you can also create those based on your plans allowance from the user account that manages billing.

You can view your current billing plan on a separate page:
The project creation can be initiated from there, or using the menu in the navigation bar, filling in basic info about the translation project to complete addition of it:
After creating the project, you are taken directly to the project page:

Creating a new translation component can be initiated via a single click there. The process of creating a component is multi-staged and automatically detects most translation parameters. There are several approaches to creating component:

**Sürüm denetiminden**  Creates component from remote version control repository.

**Mevcut bileşenden**  Creates additional component to existing one by choosing different files.

**Ek dal**  Creates additional component to existing one, just for different branch.

**Çeviri dosyalarını yükle**  Upload translation files to Weblate in case you do not have version control or do not want to integrate it with Weblate. You can later update the content using the web interface or Weblate’s REST API.

**Belgeyi çevir**  Upload single document or translation file and translate that.

**Sıfırdan başla**  Create blank translation project and add strings manually.

Once you have existing translation components, you can also easily add new ones for additional files or branches using same repository.

First you need to fill in name and repository location:
On the next page, you are presented with a list of discovered translatable resources:

As a last step, you review the translation component info and fill in optional details:
2.7. Çeviriprojeleri
2.7.3 Project configuration

Create a translation project and then add a new component for translation in it. The project is like a shelf, in which real
translations are stacked. All components in the same project share suggestions and their dictionary; the translations
are also automatically propagated through all components in a single project (unless turned off in the component
configuration), see Çeviri Belleği.

Ayrıca bakınız:
/devel/integration

These basic attributes set up and inform translators of a project:

**Proje adı**

Verbose project name, used to display the project name.

**URL kısaltması**

Project name suitable for URLs.

**Proje web sitesi**

URL where translators can find more info about the project.
This is a required parameter unless turned off by WEBSITE_REQUIRED.

**Çeviri talimatları**

URL to more site with more detailed instructions for translators.

**“Language-Team” (Çeviri-Takımı) başlığını ayarla**

Whether Weblate should manage the Language-Team header (this is a GNU gettext only feature right now).

**Paylaşılan çeviri belleği kullan**

Whether to use shared translation memory, see Paylaşılan çeviri belleği for more details.
Default value is determined by DEFAULT_SHARED_TM.
Paylaşılan çeviri belleğine katkıda bulunun

Whether to contribute to shared translation memory, see Paylaşılan çeviri belleği for more details.
Default value is determined by DEFAULT_SHARED_TM.

Erişim denetimi

Configure per project access control, see Proje erişim kontrolü for more details.
Default value can be changed by DEFAULT_ACCESS_CONTROL.

Gözden geçirmeleri etkinleştir

Enable review workflow for translations, see Dedicated reviewers.

Kaynak gözden geçirmelerini etkinleştir

Enable review workflow for source strings, see Source strings reviews.

Ayrıca bakınız:
report-source, Yorumlar

Kancaları etkinleştir

Whether unauthenticated Bildirim kancaları are to be used for this repository.

Ayrıca bakınız:
Ara dil dosyası, Quality gateway for the source strings, Bilingual and monolingual formats, Dil tanımları

Dil kod adları

Define language codes mapping when importing translations into Weblate. Use this when language codes are inconsistent in your repositories and you want to get a consistent view in Weblate or in case you want to use non-standard naming of your translation files.

The typical use case might be mapping American English to English: en_US:en
Multiple mappings to be separated by comma: en_GB:en, en_US:en
Using non standard code: ia_FOO:ia

İpucu: 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.

Ayrıca bakınız:
Dil kodlarını ayırıştırma

2.7. Çeviri projeleri
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.

Ayrıca bakınız: /devel/integration

You can find some examples of typical configurations in the Supported file formats.

Not: 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:

**Bileşen adı**

Verbose component name, used to display the component name.

**Component slug**

Component name suitable for URLs.

**Component project**

*Project configuration* where the component belongs.

**Sürüm denetleme sistemi**

VCS to use, see *Sürüm denetimi tümleştimi* for details.

Ayrıca bakınız: *Pushing changes from Weblate*

**Kaynak kod deposu**

VCS repository used to pull changes.

Ayrıca bakınız: See *Accessing repositories* for more details on specifying URLs.

İpucu: 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.
**Depo yollama URL’si**

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.

**Ayrıca bakınız:**
See Accessing repositories for more details on how to specify a repository URL and Pushing changes from Weblate for more details on pushing changes from Weblate.

**Depo tarayıcısı**

URL of repository browser used to display source files (location of used messages). When empty, no such links will be generated. You can use Template markup.

For example on GitHub, use something like: https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename}}#L{{line}}

In case your paths are relative to different folder (path contains ..), you might want to strip leading directory by parentdir filter (see Template markup): https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename|parentdir}}#L{{line}}

**Dişa aktarılmış depo URL’si**

URL where changes made by Weblate are exported. This is important when Sürekli yerelleştirme is not used, or when there is a need to manually merge changes. You can use Git exporter to automate this for Git repositories.

**Depo dalı**

Which branch to checkout from the VCS, and where to look for translations.

**Yollama dalı**

Branch for pushing changes, leave empty to use Depo dalı.

**Not:** This is currently only supported for Git, GitLab and GitHub, it is ignored for other VCS integrations.

**Ayrıca bakınız:**
Pushing changes from Weblate

**File mask**

Mask of files to translate, including path. It should include one “*” replacing language code (see Dil tanımları for info on how this is processed). In case your repository contains more than one translation file (e.g. more gettext domains), you need to create a component for each of them.

For example po/*.po or locale/*/LC_MESSAGES/django.po.

In case your filename contains special characters such as [, ], these need to be escaped as [ ] or [ ].

**Ayrıca bakınız:**
Bilingual and monolingual formats, What does mean “There are more files for the single language (en)”?
Tek dilli taban dil dosyası

Base file containing string definitions for Monolingual components.

Ayrıca bakınız:
Bilingual and monolingual formats, What does mean “There are more files for the single language (en)”?

Taban dosyası düzenle

Whether to allow editing the base file for Monolingual components.

Ara dili dosyası

Intermediate language file for Monolingual components. In most cases this is a translation file provided by developers and is used when creating actual source strings.

When set, the source strings are based on this file, but all other languages are based on Tek dilli taban dil dosyası. In case the string is not translated into the source language, translating to other languages is prohibited. This provides Quality gateway for the source strings.

Ayrıca bakınız:
Quality gateway for the source strings, Bilingual and monolingual formats, What does mean “There are more files for the single language (en)”?

Yeni çeviri şablon

Base file used to generate new translations, e.g. .pot file with gettext.

İpucu: 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.

Ayrıca bakınız:
adding-translation, Adding new translations, Yeni çeviri ekleniyor, Bilingual and monolingual formats, What does mean “There are more files for the single language (en)”?

Dosya biçimi

Translation file format, see also Supported file formats.

Kaynak dizgi hata bildirme adresi

Email address used for reporting upstream bugs. This address will also receive notification about any source string comments made in Weblate.
Çeviri yaymaya izin ver

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`.

Ayrıca bakınız:

Keeping translations same across components

Önerileri etkinleştir

Whether translation suggestions are accepted for this component.

Öneri oylaması

Turns on vote casting for suggestions, see Öneri oylaması.

Önerileri kendiliğinden kabul et

Automatically accept voted suggestions, see Öneri oylaması.

Çeviri işaretleri

Customization of quality checks and other Weblate behavior, see Customizing behavior using flags.

Zorunlu denetimler

List of checks which can not be ignored, see Enforcing checks.

**Not:** Enforcing the check does not automatically enable it, you still should enabled it using Customizing behavior using flags in Çeviri işaretleri or Additional info on source strings.

Çeviri lisansı

License of the translation (does not need to be the same as the source code license).

Katılımcı sözleşmesi

Bir kullanıcının bu bileşeni çevirmeden önce onaylaması gereken kullanıcı sözleşmesi.
Yeni çeviri ekleniyor

How to handle requests for creation of new languages. Available options:

İletişim sorumluşu User can select desired language and the project maintainers will receive a notification about this. It is up to them to add (or not) the language to the repository.

Çeviri talimatları URL’sini göster User is presented a link to page which describes process of starting new translations. Use this in case more formal process is desired (for example forming a team of people before starting actual translation).

Yeni dil dosyası oluştur User can select language and Weblate automatically creates the file for it and translation can begin.

Yeni çeviri ekleme devre dışı There will be no option for user to start new translation.

İpucu: The project admins can add new translations even if it is disabled here when it is possible (either Yeni çeviriler için şablon or the file format supports starting from an empty file).

Ayrıca bakınız: adding-translation, Adding new translations

Dizgileri yönet

4.5 sürümünde geldi.

Configures whether users in Weblate will be allowed to add new strings and remove existing ones. Adjust this to match your localization workflow - how the new strings are supposed to be introduced.

For bilingual formats, the strings are typically extracted from the source code (for example by using xgettext) and adding new strings in Weblate should be disabled (they would be discarded next time you update the translation files). In Weblate you can manage strings for every translation and it does not enforce the strings in all translations to be consistent.

For monolingual formats, the strings are managed only on source language and are automatically added or removed in the translations. The strings appear in the translation files once they are translated.

Ayrıca bakınız: Bilingual and monolingual formats, adding-new-strings, POST /api/translations/(string:project)/(string:component)/(string:language)/units/

Dil kodu stili

Weblate tarafından oluşturulan çeviriler için dosya adım oluşturmak için kullanılan dil kodunu özelleştirin.

Ayrıca bakınız: Adding new translations, Dil kodu, Dil kodlarını ayırırma
Birleştirme stili

You can configure how updates from the upstream repository are handled. This might not be supported for some VCSs. See Merge or rebase for more details.
Default value can be changed by DEFAULT_MERGE_STYLE.

Commit, add, delete, merge and addon messages

Message used when committing a translation, see Template markup.
Default value can be changed by DEFAULT_ADD_MESSAGE, DEFAULT_ADDON_MESSAGE, DEFAULT_COMMIT_MESSAGE, DEFAULT_DELETE_MESSAGE, DEFAULT_MERGE_MESSAGE.

İşlemeye yolla

Whether committed changes should be automatically pushed to the upstream repository. When enabled, the push is initiated once Weblate commits changes to its underlying repository (see Lazy commits). To actually enable pushing Repository push URL has to be configured as well.

İşlenecek değişikliklerin yaşası

Sets how old (in hours) changes have to be before they are committed by background task or the commit_pending management command. All changes in a component are committed once there is at least one change older than this period.
Default value can be changed by COMMIT_PENDING_HOURS.

İpucu: There are other situations where pending changes might be committed, see Lazy commits.

Hata durumunda kilitle

Locks the component (and linked components, see Weblate internal URLs) upon the first failed push or merge into its upstream repository, or pull from it. This avoids adding another conflicts, which would have to be resolved manually.
The component will be automatically unlocked once there are no repository errors left.

Kaynak dil

Language used for source strings. Change this if you are translating from something else than English.

İpucu: In case you are translating bilingual files from English, but want to be able to do fixes in the English translation as well, choose English (Developer) as a source language to avoid conflict between the name of the source language and the existing translation.
For monolingual translations, you can use intermediate translation in this case, see Ara dil dosyası.
Dil süzgeci

Dosya maskesi tararırken çeviri süzme için kullanılan düzenli ifade. Weblate tarafından yönetilen dillerin listesini sınırlamak için kullanılabilir.

**Not:** You need to list language codes as they appear in the filename.

Some examples of filtering:

<table>
<thead>
<tr>
<th>Filter description</th>
<th>Düzenli ifade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected languages only</td>
<td>^(cs</td>
</tr>
<tr>
<td>Exclude languages</td>
<td>^(!it</td>
</tr>
<tr>
<td>Filter two letter codes only</td>
<td>^..$</td>
</tr>
<tr>
<td>Exclude non language files</td>
<td>^(!blank)$+.+$</td>
</tr>
<tr>
<td>Include all files (default)</td>
<td>^[^.]+$</td>
</tr>
</tbody>
</table>

Çeşitlerin düzenli ifadesi

Regular expression used to determine the variants of a string, see variants.

**Not:** Most of the fields can be edited by project owners or managers, in the Weblate interface.

Ayrıca bakınız:

*Does Weblate support other VCSes than Git and Mercurial?*, alerts

Öncelik

Yüksek öncelikli bileşenler çevirmenlere ilk olarak sunulur.

Restricted access

By default the component is visible to anybody who has access to the project, even if the person cannot perform any changes in the component. This makes it easier to keep translation consistency within the project.

Restricting access at a component, or component-list level takes over access permission to a component, regardless of project-level permissions. You will have to grant access to it explicitly. This can be done through granting access to a new user group and putting users in it, or using the default *custom or private* access control groups.

The default value can be changed in `DEFAULT_RESTRICTED_COMPONENT`.

İpucu: This applies to project admins as well — please make sure you will not lose access to the component after toggling the status.
Projelerde paylaş

You can choose additional projects where the component will be visible. Useful for shared libraries which you use in several projects.

**Not:** Sharing a component doesn’t change its access control. It only makes it visible when browsing other projects. Users still need access to the actual component to browse or translate it.

Sözlük olarak kullan

4.5 sürümünde geldi.
Bu bileşenin sözlük olarak kullanılmasına izin verir. Nasıl listeleniceğini Sözlük rengi kullanarak yapılandırabilirsiniz.
The glossary will be accessible in all projects defined by Projelerde paylaş.
It is recommended to enable Dizgileri yönet on glossaries in order to allow adding new words to them.

**Ayrıca bakınız:**
Sözlük

Sözlük rengi

Display color for a glossary used when showing word matches.

2.7.5 Template markup

Weblate uses simple markup language in several places where text rendering is needed. It is based on The Django template language, so it can be quite powerful.
Currently it is used in:

- Commit message formatting, see Component configuration
- Several addons
  - Bileşen keşfi
  - İstatistik oluştürücü
  - Eklentiden betikleri çalıştırma

There following variables are available in the component templates:

```{{ language_code }}
Dil kodu
```

```{{ language_name }}
Dil adı
```

```{{ component_name }}
Bileşen adı
```

```{{ component_slug }}
Component slug
```

```{{ project_name }}
Proje adı
```

```{{ project_slug }}
Project slug
```

```{{ url }}
Translation URL
```

```{{ filename }}
Çeviri dosya adı
```

```{{ stats }}
Translation stats, this has further attributes, examples below.
```

```{{ stats.all }}
Total strings count
```

```{{ stats.fuzzy }}
Count of strings needing review
```
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\ \{\{\\ file-name parentdir \}\}\n
You can combine them with filters:

- \{\{ component|title \}\}\n
You can use conditions:

- \{\% if stats.translated_percent > 80 \%\}Well\ translated!\{\% endif \%\}\n
There is additional tag available for replacing characters:

- \{\% replace component "-" " " \%\}\n
You can combine it with filters:

- \{\% replace component|capfirst "-" " " \%\}\n
There are also additional filter to manipulate with filenames:

Directory of a file: \{\{ filename|dirname \}\}\nFile without extension: \{\{ filename|stripext \}\}\nFile in parent dir: \{\{ filename|parentdir \}\}\nIt can be used multiple times: \{\{ filename|parentdir|parentdir \}\}\n
...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](Production setup) for more details, especially:

- Configure Celery for executing background tasks (see [Background tasks using Celery](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 Bileşen keşfi 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.

Ayrıca bakınız:

Management commands, Bileşen keşfi

2.8 Dil tanımları

Farklı çevirileri doğru bir şekilde sunmak için dil adı, metin yönu, çoğul tanımları ve dil kodu hakkında bilgiler gereklidir.

2.8.1 Dil kodlarını ayırıştırma

Çevirileri ayırıştırırken, Weblate dil kodunu (genellikle ISO 639-1 olanı) herhangi bir mevcut dil nesnesine eşlemeye çalışır.

Dil kod adları ile bu eşleme proje düzeyinde daha fazla ayarlayabilirsiniz.

Tam eşleme bulunamazsa, mevcut bir dile en iyi şekilde uydurulmaya çalışılır. Aşağıdaki adımlar denenir:

- Büyük/küçük harfe duyarlı olmayan aramalar.
- Alt çizgileri ve kısa çizgileri normalleştirmeye.
- Yerleşik dil kod adları arasında arama.
- Dil adına göre arama.
- Belirli bir dil için varsayan ülke kodunu yok sayma—cs_CZ yerine cs seçme.
The Weblate Manual, Yayım 4.9

Bunun da başarısız olması durumunda, varsayılanlar kullanılarak (soldan sağa metin yönü, bir tane çoğul) yeni bir dil tanıımı oluşturulacaktır. xx_XX koduyla otomatik olarak oluşturululan dil xx_XX (generated) olarak adlandırılacaktır. Bunu daha sonra yönetici arayüzünde değiştirmek (bkz: Changing language definitions) ve sorun izleyiciye bildirmek (bkz: Weblate’e Katkıda Bulunma) isteyebilirsiniz, böylece uygun tanımın gelecek Weblate sürümüne eklenmesini sağlayabilirsiniz.

İpucu: Dil olarak istenmeyen bir şey görürseniz, çevirileri ayrıtırırken bir tür bir dosayı yok saymak için Dil düzgeci ayarlamak isteyebilirsiniz.

Ayrıca bakınız:
Dil kodu, Adding new translations

2.8.2 Changing language definitions

You can change language definitions in the languages interface (/languages/ URL).

While editing, make sure all fields are correct (especially plurals and text direction), otherwise translators will be unable to properly edit those translations.

2.8.3 Yerleşik dil tanımları

Definitions for about 600 languages are included in Weblate and the list is extended in every release. Whenever Weblate is upgraded (more specifically whenever weblate migrate is executed, see Generic upgrade instructions) the database of languages is updated to include all language definitions shipped in Weblate.

This feature can be disable using UPDATE_LANGUAGES. You can also enforce updating the database to match Weblate built-in data using setuplang.

Ayrıca bakınız:
Yerleşik dil tanımlarını genişletme

2.8.4 Belirsiz dil kodları ve makro diller

In many cases it is not a good idea to use macro language code for a translation. The typical problematic case might be Kurdish language, which might be written in Arabic or Latin script, depending on actual variant. To get correct behavior in Weblate, it is recommended to use individual language codes only and avoid macro languages.

Ayrıca bakınız:
Macrolanguages definition, List of macrolanguages

2.8.5 Dil tanımları

Each language consists of following fields:
Dil kodu

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.

Ayrıca bakınız:
Dil kodlarını ayrıştırma, Adding new translations

Dil adı

Visible name of the language. The language names included in Weblate are also being localized depending on user interface language.

Metin yönü

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.

Çoğul formülü

Gettext compatible plural formula used to determine which plural form is used for given count.

Ayrıca bakınız:
Çoğullar, GNU gettext utilities: Plural forms, Language Plural Rules by the Unicode Consortium

2.8.6 Adding new translations

2.18 sürümünde değişti: 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 Yeni çeviriler için şablon 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 Yeni çeviriler için şablon is empty and the file format supports it, an empty file is created where new strings will be added once they are translated.

The Dil kodu stili allows you to customize language code used in generated filenames:

Dosya biçimine göre varsayılan Dependent on file format, for most of them POSIX is used.

Ayrıca olarak alt çizgi kullanılan POSIX stili Typically used by gettext and related tools, produces language codes like pt_BR.

Ülke kodu da dahil olmak üzere ayrıca olarak alt çizgi kullanılan POSIX stili POSIX style language code including the country code even when not necessary (for example cs_CZ).

Ayrıca olarak kısa çizgi kullanılan BCP stili Typically used on web platforms, produces language codes like pt_BR.
BCP style language code including the country code even when not necessary (for example cs-CZ).

Android stili Only used in Android apps, produces language codes like pt-br.

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

Additionally, any mappings defined in Dil kod adları are applied in reverse.

Not: Weblate recognizes any of these when parsing translation files, the above settings only influences how new files are created.

Ayrıca bakınız:
Dil kodu, Dil kodlarını ayrıştırma

2.9 Sürekli yerelleştirme

There is infrastructure in place so that your translation closely follows development. This way translators can work on translations the entire time, instead of working through huge amount of new text just prior to release.

Ayrıca bakınız:
/devel/integration describes basic ways to integrate your development with Weblate.

This is the process:

1. Developers make changes and push them to the VCS repository.
2. Optionally the translation files are updated (this depends on the file format, see Why does Weblate still show old translation strings when I’ve updated the template?).
3. Weblate pulls changes from the VCS repository, see Updating repositories.
4. Once Weblate detects changes in translations, translators are notified based on their subscription settings.
5. Translators submit translations using the Weblate web interface, or upload offline changes.
6. Once the translators are finished, Weblate commits the changes to the local repository (see Lazy commits) and pushes them back if it has permissions to do so (see Pushing changes from Weblate).
2.9.1 Updating repositories

You should set up some way of updating backend repositories from their source.

- Use Bildirim kancaları to integrate with most of common code hosting services:
  - Automatically receiving changes from GitHub
  - Automatically receiving changes from GitLab
  - Automatically receiving changes from Bitbucket
  - Automatically receiving changes from Pagure
  - Automatically receiving changes from Azure Repos

- Manually trigger update either in the repository management or using Weblate’s REST API or Weblate Client

- Enable AUTO_UPDATE to automatically update all components on your Weblate instance

- Execute `updategit` (with selection of project or --all to update all)

Whenever Weblate updates the repository, the post-update addons will be triggered, see Eklentiler.
Avoiding merge conflicts

The merge conflicts from Weblate arise when the same file was changed both in Weblate and outside it. There are two approaches to deal with that - avoid edits outside Weblate or integrate Weblate into your updating process, so that it flushes changes prior to updating the files outside Weblate.

The first approach is easy with monolingual files - you can add new strings within Weblate and leave whole editing of the files there. For bilingual files, there is usually some kind of message extraction process to generate translatable files from the source code. In some cases this can be split into two parts - one for the extraction generates template (for example gettext POT is generated using `xgettext`) and then further process merges it into actual translations (the gettext PO files are updated using `msgmerge`). You can perform the second step within Weblate and it will make sure that all pending changes are included prior to this operation.

The second approach can be achieved by using Weblate's REST API to force Weblate to push all pending changes and lock the translation while you are doing changes on your side.

The script for doing updates can look like this:

```bash
# 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
# Push changes to upstream repository
git push
# Tell Weblate to pull changes (not needed if Weblate follows your repo automatically)
wlc pull
# Unlock translations
wlc unlock
```

If you have multiple components sharing the same repository, you need to lock them all separately:

```bash
wlc lock foo/bar
wlc lock foo/baz
wlc lock foo/baj
```

**Note:** The example uses Weblate Client, which needs configuration (API keys) to be able to control Weblate remotely. You can also achieve this using any HTTP client instead of wlc, e.g. curl, see Weblate's REST API.

Ayrıca bakınız:

- [Weblate Client](#)

Automatically receiving changes from GitHub

Weblate comes with native support for GitHub.

If you are using Hosted Weblate, the recommended approach is to install the Weblate app, that way you will get the correct setup without having to set much up. It can also be used for pushing changes back.

To receive notifications on every push to a GitHub repository, add the Weblate Webhook in the repository settings (Webhooks) as shown on the image below:
For the payload URL, append /hooks/github/ to your Weblate URL, for example for the Hosted Weblate service, this is https://hosted.weblate.org/hooks/github/.

You can leave other values at default settings (Weblate can handle both content types and consumes just the push event).

*Ayırca bakınız:*

**POST /hooks/github/, Accessing repositories from Hosted Weblate**

**Automatically receiving changes from Bitbucket**

Weblate has support for Bitbucket webhooks, add a webhook which triggers upon repository push, with destination to /hooks/bitbucket/ URL on your Weblate installation (for example https://hosted.weblate.org/hooks/bitbucket/).
Ayrıca bakınız:

POST /hooks/bitbucket/, Accessing repositories from Hosted Weblate

Automatically receiving changes from GitLab

Weblate has support for GitLab hooks, add a project webhook with destination to /hooks/gitlab/ URL on your Weblate installation (for example https://hosted.weblate.org/hooks/gitlab/).

Ayrıca bakınız:

POST /hooks/gitlab/, Accessing repositories from Hosted Weblate

Automatically receiving changes from Pagure

3.3 sürümünde geldi.

Weblate has support for Pagure hooks, add a webhook with destination to /hooks/pagure/ URL on your Weblate installation (for example https://hosted.weblate.org/hooks/pagure/). This can be done in Activate Web-hooks under Project options:
Additionally, you can also use Azure Repos webhooks to automatically receive changes. To do this, go to the "Service hooks" under "Project settings" and add a webhook for the "Code pushed" event with destination URL /hooks/azure/ on your Weblate installation (for example, https://hosted.weblate.org/hooks/azure/). This can be done in Service hooks under Project settings.

**Ayrıca bakınız:**

POST /hooks/pagure/, Accessing repositories from Hosted Weblate

**Automatically receiving changes from Azure Repos**

3.8 version with support for Azure Repos webhooks. To add a webhook for the "Code pushed" event, go to /hooks/azure/ URL on your Weblate installation (for example, https://hosted.weblate.org/hooks/azure/). This can be done in Service hooks under Project settings.

**Ayrıca bakınız:**

Webhooks in Azure DevOps manual, POST /hooks/azure/, Accessing repositories from Hosted Weblate

---

2.9. Sürekli yerelleştirme 257
Automatically receiving changes from Gitea Repos

3.9 sürümünde geldi.

Weblate has support for Gitea webhooks, add a Gitea Webhook for Push events event with destination to /hooks/gitea/ URL on your Weblate installation (for example https://hosted.weblate.org/hooks/gitea/). This can be done in Webhooks under repository Settings.

Ayrıca bakınız:

Webhooks in Gitea manual, POST /hooks/gitea/, Accessing repositories from Hosted Weblate

Automatically receiving changes from Gitee Repos

3.9 sürümünde geldi.

Weblate has support for Gitee webhooks, add a WebHook for Push event with destination to /hooks/gitee/ URL on your Weblate installation (for example https://hosted.weblate.org/hooks/gitee/). This can be done in WebHooks under repository Management.

Ayrıca bakınız:

Webhooks in Gitee manual, POST /hooks/gitee/, Accessing repositories from Hosted Weblate

Automatically updating repositories nightly

Weblate automatically fetches remote repositories nightly to improve performance when merging changes later. You can optionally turn this into doing nightly merges as well, by enabling AUTO_UPDATE.

2.9.2 Pushing changes from Weblate

Each translation component can have a push URL set up (see Depo yollama URL’si), and in that case Weblate will be able to push change to the remote repository. Weblate can be also be configured to automatically push changes on every commit (this is default, see İşlemeye yolla). If you do not want changes to be pushed automatically, you can do that manually under Repository maintenance or using API via wlc push.

The push options differ based on the Sürüm denetimi tümleşimi used, more details are found in that chapter.

In case you do not want direct pushes by Weblate, there is support for GitHub, GitLab, Pagure pull requests or Gerrit reviews, you can activate these by choosing GitHub, GitLab, Gerrit or Pagure as Sürüm denetleme sistemi in Component configuration.

Overall, following options are available with Git, GitHub and GitLab:

<table>
<thead>
<tr>
<th>Desired setup</th>
<th>Sürüm denetleme sistemi</th>
<th>Depo yollama URL’si</th>
<th>Yollama dali</th>
</tr>
</thead>
<tbody>
<tr>
<td>No push</td>
<td>Git</td>
<td>empty</td>
<td>empty</td>
</tr>
<tr>
<td>Push directly</td>
<td>Git</td>
<td>SSH URL</td>
<td>empty</td>
</tr>
<tr>
<td>Ayırt bir dala yolla</td>
<td>Git</td>
<td>SSH URL</td>
<td>Branch name</td>
</tr>
<tr>
<td>GitHub pull request from fork</td>
<td>GitHub</td>
<td>empty</td>
<td>empty</td>
</tr>
<tr>
<td>GitHub pull request from branch</td>
<td>GitHub</td>
<td>SSH URL¹</td>
<td>Branch name</td>
</tr>
<tr>
<td>GitLab merge request from fork</td>
<td>GitLab</td>
<td>empty</td>
<td>empty</td>
</tr>
<tr>
<td>GitLab merge request from branch</td>
<td>GitLab</td>
<td>SSH URL¹</td>
<td>Branch name</td>
</tr>
<tr>
<td>Pagure merge request from fork</td>
<td>Pagure</td>
<td>empty</td>
<td>empty</td>
</tr>
<tr>
<td>Pagure merge request from branch</td>
<td>Pagure</td>
<td>SSH URL¹</td>
<td>Branch name</td>
</tr>
</tbody>
</table>

1 Can be empty in case Kaynak kod deposu supports pushing.

Not: You can also enable automatic pushing of changes after Weblate commits, this can be done in İşlemeye yolla.
Ayrıca bakınız:

See *Accessing repositories* for setting up SSH keys, and *Lazy commits* for info about when Weblate decides to commit changes.

**Protected branches**

If you are using Weblate on protected branch, you can configure it to use pull requests and perform actual review on the translations (what might be problematic for languages you do not know). An alternative approach is to waive this limitation for the Weblate push user.

For example on GitHub this can be done in the repository configuration:

2.9.3 **Merge or rebase**

By default, Weblate merges the upstream repository into its own. This is the safest way in case you also access the underlying repository by other means. In case you don't need this, you can enable rebasing of changes on upstream, which will produce a history with fewer merge commits.

**Not:** Rebasing can cause you trouble in case of complicated merges, so carefully consider whether or not you want to enable them.
2.9.4 Interacting with others

Weblate makes it easy to interact with others using its API.

Ayrıca bakınız:
Weblate’s REST API

2.9.5 Lazy commits

The behaviour of Weblate is to group commits from the same author into one commit if possible. This greatly reduces the number of commits, however you might need to explicitly tell it to do the commits in case you want to get the VCS repository in sync, e.g. for merge (this is by default allowed for the Managers group, see Yetkiler listesi).

The changes in this mode are committed once any of the following conditions are fulfilled:

- Somebody else changes an already changed string.
- A merge from upstream occurs.
- An explicit commit is requested.
- Change is older than period defined as İşlenecek değişikliklerin yaşısı on Component configuration.

İpucu: Commits are created for every component. So in case you have many components you will still see lot of commits. You might utilize Git işlemelerini sıkıştır addon in that case.

If you want to commit changes more frequently and without checking of age, you can schedule a regular task to perform a commit:

```
CELERY_BEAT_SCHEDULE = {
    # Unconditionally commit all changes every 2 minutes
    "commit": {
        "task": "weblate.trans.tasks.commit_pending",
        # Committing hours will honor per component settings,
        # otherwise components with no changes older than this
        # won't be committed
        "kwargs": {"hours": 0},
        # How frequently to execute the job in seconds
        "schedule": 120,
    }
}
```

2.9.6 Processing repository with scripts

The way to customize how Weblate interacts with the repository is Eklentiler. Consult Eklentiden betikleri çalıştırma for info on how to execute external scripts through addons.

2.9.7 Keeping translations same across components

Once you have multiple translation components, you might want to ensure that the same strings have same translation. This can be achieved at several levels.
Translation propagation

With Çeviri yaymaya izin ver enabled (what is the default, see Component configuration), all new translations are automatically done in all components with matching strings. Such translations are properly credited to currently translating user in all components.

**Not:** The translation propagation requires the key to be match for monolingual translation formats, so keep that in mind when creating translation keys.

Consistency check

The Tutarsız check fires whenever the strings are different. You can utilize this to review such differences manually and choose the right translation.

Kendiliğinden çeviri

Automatic translation based on different components can be way to synchronize the translations across components. You can either trigger it manually (see Kendiliğinden çeviri) or make it run automatically on repository update using addon (see Kendiliğinden çeviri).

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 Katılımcı sözleşmesi

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:

Please pay attention to the licensing info, as this specifies how translations can be used.

By registering you agree to use your name and e-mail in the commits, and provide your contribution under the license defined by each localization project.

You have agreed to the following as a contributor:

- WebateOrg/Language names
2.11 Çeviri işleyişi

2.11.1 Öneri oylaması

Everyone can add suggestions by default, to be accepted by signed in users. Suggestion voting can be used to make use of a string when more than one signed-in user agrees, by setting up the Component configuration with Suggestion voting to turn on voting, and Autoaccept suggestions to set a threshold for accepted suggestions (this includes a vote from the user making the suggestion if it is cast).

**Not:** Once automatic acceptance is set up, normal users lose the privilege to directly save translations or accept suggestions. This can be overridden with the Edit string when suggestions are enforced permission.

You can combine these with access control into one of the following setups:

- Users suggest and vote for suggestions and a limited group controls what is accepted. - Turn on voting. - Turn off automatic acceptance. - Don’t let users save translations.

- Users suggest and vote for suggestions with automatic acceptance once the defined number of them agree. - Turn on voting. - Set the desired number of votes for automatic acceptance.

- Optional voting for suggestions. (Can optionally be used by users when they are unsure about a translation by making multiple suggestions.) - Only turn on voting.

2.11.2 Additional info on source strings

Enhance the translation process by adding additional info to the strings including explanations, string priorities, check flags and visual context. Some of that info may be extracted from the translation files and some may be added by editing the additional string info:
Access this directly from the translation interface by clicking the “Edit” icon next to **Screenshot context** or **Flags**.
2.11. Çeviri işleşi
Strings prioritization

2.0 sürümünde geldi.

String priority can be changed to offer higher priority strings for translation earlier by using the `priority` flag.

İpucu: This can be used to order the flow of translation in a logical manner.

Ayrıca bakınız:

Quality checks

Çeviri işaretleri

2.4 sürümünde geldi.

3.3 sürümünde değişti: 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.

Ayrıca bakınız:

Quality checks, Customizing behavior using flags

Açıklama

4.1 sürümünde değişti: In previous versions this has been called Extra context.

Use the explanation to clarify scope or usage of the translation. You can use Markdown to include links and other markup.
Visual context for strings

2.9 sürümünde geldi.

You can upload a screenshot showing a given source string in use within your program. This helps translators understand where it is used, and how it should be translated.

The uploaded screenshot is shown in the translation context sidebar:

In addition to Additional info on source strings, screenshots have a separate management interface under the Tools menu. Upload screenshots, assign them to source strings manually, or use optical character recognition to do so.

Once a screenshot is uploaded, this interface handles management and source string association:
## 2. Administratordocs

### Screenshot had been uploaded, you can now assign it to source strings.

**Assigned source strings**

<table>
<thead>
<tr>
<th>Language</th>
<th>Location</th>
<th>Assigned screenshots</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No matching strings found.</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Screenshot is shown to add visual context for all listed source strings.

### Assign source strings

<table>
<thead>
<tr>
<th>Language</th>
<th>Location</th>
<th>Assigned screenshots</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No matching strings found.</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Source string search

- **Search**
- **Automatically recognize**

### Image

#### Source string

- Hello, world!
- One. Chang tian has had banana.
- Other. Chang tian has had bananas.
- *By Webate at [http://demo.webate.org](http://demo.webate.org)*;
- *Thank you for using Webate.*

Screenshot is shown to add visual context for all listed source strings.

### Edit screenshot

**Screenshot name**

- Automatic translation

**Image**

- Currently: screenshot/screenshot.png
- Change: **Choose File** (No file chosen)
- Upload JPEG or PNG images up to 2000x2000 pixels.

**Save**

**Screenshot details**

- **Created**:
- **Uploaded by**:
- **Language**: English

### Delete screenshot

Deleting screenshot will remove it from all associated source strings.

- **Delete**
2.12 Checks and fixups

2.12.1 Custom automatic fixups

You can also implement your own automatic fixup in addition to the standard ones and include them in `AUTOFIX_LIST`.

The automatic fixes are powerful, but can also cause damage; be careful when writing one.

For example, the following automatic fixup would replace every occurrence of the string `foo` in a translation with `bar`:

```python
# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.
#
from django.utils.translation import gettext_lazy as _
from weblate.trans.autofixes.base import AutoFix

class ReplaceFooWithBar(AutoFix):
    """Replace foo with bar."""

    name = _("Foobar")

    def fix_single_target(self, target, source, unit):
        if "foo" in target:
            return target.replace("foo", "bar"), True
    return target, False
```

To install custom checks, provide a fully-qualified path to the Python class in the `AUTOFIX_LIST`, see Custom quality checks, addons and auto-fixes.

2.12.2 Customizing behavior using flags

You can fine-tune the behavior of Weblate (mostly checks) for each source string (in source strings review, see Additional info on source strings) or in the Component configuration (Çeviri işaretleri). Some file formats also allow to specify flags directly in the format (see Supported file formats).

The flags are comma-separated, the parameters are separated with colon. You can use quotes to include whitespace or special chars in the string. For example:

```text
placeholders:"special:value":"other value", regex:.*
```

Here is a list of flags currently accepted:
rst-text  Treat a text as an reStructuredText document, affects Değiştirilmemiş çeviri.
dos-eol  Uses DOS end-of-line markers instead of Unix ones (\r\n instead of \n).
read-only  The string is read-only and should not be edited in Weblate, see Salt okunur dizgiler.
priority: N  Priority of the string. Higher priority strings are presented first for translation. The default priority is 100, the higher priority a string has, the earlier it is offered for translation.
max-length: N  Limit the maximal length for a string to N characters, see Çevirinin en çok uzunluğu.
xml-text  Treat text as XML document, affects XML sözdizimi and XML işaretlemesi.
font-family: NAME  Define font-family for rendering checks, see Managing fonts.
font-weight: WEIGHT  Define font-weight for rendering checks, see Managing fonts.
font-size: SIZE  Define font-size for rendering checks, see Managing fonts.
font-spacing: SPACING  Define letterspacing for rendering checks, see Managing fonts.
placeholders: NAME:NAME2:...  Placeholder strings expected in translation, see Yer tutucular.
replacements: FROM:TO:FROM2:TO2...  Replacements to perform when checking resulting text parameters (for example in Çeviri maksimum boyutu or Çevirinin en çok uzunluğu). The typical use case for this is to expand placeables to ensure that the text fits even with long values, for example: replacements:%s:"John Doe".
variants: SOURCE  Mark this string as a variant of string with matching source. See variants.
regex: REGEX  Çevir dosyalarını eşleştirme için düzenli ifade, bakınız Düzenli ifade.
forbidden  Indicates forbidden translation in a glossary, see Yasaklı çeviriler.
strict-same  Make “Unchanged translation” avoid using built-in words blacklist, see Değiştirilmemiş çeviri.
check-glossary  Sözlüğü takip etme nitelikten etkinleştir.
angularjs-format  AngularJS ilişkilendirme dizgisi nitelikden etkinleştir.
c-format  C biçimi nitelikden etkinleştir.
c-sharp-format  C# biçimi nitelikden etkinleştir.
es-format  ECMAScript şablon değişmezleri nitelikden etkinleştir.
i18next-interpolation  Enable the i18next ilişkilendirmesi quality check.
java-format  Java biçimi nitelikden etkinleştir.
java-messageformat  Java MessageFormat nitelikden etkinleştir.
javascript-format  JavaScript biçimi nitelikden etkinleştir.
lua-format  Lua biçimi nitelikden etkinleştir.
object-pascal-format  Object Pascal biçimi nitelikden etkinleştir.
percent-placeholders  Enable the Yüzde yer tutucuları quality check.
perl-format  Perl biçimi nitelikden etkinleştir.
php-format  PHP biçimi nitelikden etkinleştir.
python-brace-format  Enable the Python ayracı biçimi quality check.
python-format  Enable the Python biçimi quality check.
qt-format  Qt biçimi nitelikden etkinleştir.
qt-plural-format  Qt çoğul biçimi nitelikden etkinleştir.
ruby-format  Ruby biçimi nitelikden etkinleştir.
scheme-format  Şema biçimi nitelikden etkinleştir.
vue-format Vue I18n biçimlendirmesi nitelik denetimini etkinleştir.

md-text Treat text as a Markdown document. Enable Markdown bağlantıları, Markdown başvuruları, and Markdown sözdizimi quality checks.

safe-html Güvenli olmayan HTML nitelik denetimini etkinleştir.

url The string should consist of only a URL. Enable the URL quality check.

ignore-bbcode BBcode işaretlemesi nitelik denetimini atla.
ignore-duplicate Art arda yineleden sözcükler nitelik denetimini atla.
ignore-check-glossary Sözlüğü takip etme nitelik denetimini atla.
ignore-double-space Çift boşluk nitelik denetimini atla.
ignore-angularjs-format AngularJS ilişkilendirme dizgisi nitelik denetimini atla.
ignore-c-format C biçimi nitelik denetimini atla.
ignore-c-sharp-format C# biçimi nitelik denetimini atla.
ignore-es-format ECMAScript şablon değişmezleri nitelik denetimini atla.
ignore-il18next-interpolation check-il18next-format nitelik denetimini atla.
ignore-java-format Java biçimi nitelik denetimini atla.
ignore-java-messageformat Java MessageFormat nitelik denetimini atla.
ignore-javascript-format JavaScript biçimi nitelik denetimini atla.
ignore-lua-format Lua biçimi nitelik denetimini atla.
ignore-object-pascal-format Object Pascal biçimi nitelik denetimini atla.
ignore-percent-placeholders Yüzde yer tutucuları nitelik denetimini atla.
ignore-perl-format Perl biçimi nitelik denetimini atla.
ignore-php-format PHP biçimi nitelik denetimini atla.
ignore-python-brace-format Python ayracı biçimi nitelik denetimini atla.
ignore-python-format Python biçimi nitelik denetimini atla.
ignore-qt-format Qt biçimi nitelik denetimini atla.
ignore-qt-plural-format Qt çoğul biçimi nitelik denetimini atla.
ignore-ruby-format Ruby biçimi nitelik denetimini atla.
ignore-scheme-format Şema biçim nitelik denetimini atla.
ignore-vue-format Vue I18n biçimlendirmesi nitelik denetimini atla.
ignore-translated Çevrilmış nitelik denetimini atla.
ignore-inconsistent Tutarsız nitelik denetimini atla.
ignore-kashida Kashida harf kullanıldı nitelik denetimini atla.
ignore-md-link Markdown bağlantıları nitelik denetimini atla.
ignore-md-reflink Markdown başvuruları nitelik denetimini atla.
ignore-md-syntax Markdown sözdizimi nitelik denetimini atla.
ignore-max-length Çevirinin en çok uzunluğu nitelik denetimini atla.
ignore-max-size Çeviri maksimum boyutu nitelik denetimini atla.
ignore-escaped-newline \n uyuşmadı nitelik denetimini atla.
ignore-end-colon İki nokta üst üste uyuşmadı nitelik denetimini atla.
ignore-end-ellipsis Üç nokta uyuşmadı nitelik denetimini atla.
ignore-end-exclamation Ünlem işareti uyuşmadı nitelik denetimini atla.
ignore-end-stop Nokta işareti uyuşmadı nitelik denetimini atla.
ignore-end-question Soru işareti uyuşmadı nitelik denetimini atla.
ignore-end-semicolon Noktalı virgül uyuşmadı nitelik denetimini atla.
ignore-newline-count Uyuşmayan satır sonuçları nitelik denetimini atla.
ignore-plurals Eksik çoğul nitelik denetimini atla.
ignore-placeholders Yer tutucular nitelik denetimini atla.
ignore-punctuation-spacing Noktalama aralığı nitelik denetimini atla.
ignore-regex Düzenli ifade nitelik denetimini atla.
ignore-same-plurals Aynı çoğullar nitelik denetimini atla.
ignore-begin-newline Baştaki yeni satır nitelik denetimini atla.
ignore-begin-space Baştaki boşluk nitelik denetimini atla.
ignore-end-newline Sondaki yeni satır nitelik denetimini atla.
ignore-end-space Sondaki boşluk nitelik denetimini atla.
ignore-same Değişirmemiş çeviri nitelik denetimini atla.
ignore-safe-html Güvenli olmayan HTML nitelik denetimini atla.
ignore-url URL nitelik denetimini atla.
ignore-xml-tags XML işaretlemesi nitelik denetimini atla.
ignore-xml-invalid XML sözdizimi nitelik denetimini atla.
ignore-zero-width-space Sıfır genişlikli boşluk nitelik denetimini atla.
ignore-ellipsis Üç nokta nitelik denetimini atla.
ignore-long-untranslated Uzun süre çevrilmeyen nitelik denetimini atla.
ignore-multiple-failures Birden çok kusur denetimi nitelik denetimini atla.
ignore-unnamed-format Birden çok adsız değişken nitelik denetimini atla.
ignore-optional-plural Çoğulsuz nitelik denetimini atla.

**Not:** Generally the rule is named `ignore-*` for any check, using its identifier, so you can use this even for your custom checks.

These flags are understood both in *Component configuration* settings, per source string settings and in the translation file itself (for example in GNU gettext).

### 2.12.3 Enforcing checks

3.11 sürümünde geldi.

You can configure a list of checks which can not be ignored by setting `Zorunlu denetimler` in *Component configuration*. Each listed check can not be ignored in the user interface and any string failing this check is marked as *Needs editing* (see *Translation states*).
2.12.4 Managing fonts

3.7 sürümünde geldi.

İpucu: Fonts uploaded into Weblate are used purely for purposes of the Çeviri maksimum boyutu check, they do not have an effect in Weblate user interface.

The Çeviri maksimum boyutu check used to calculate dimensions of the rendered text needs font to be loaded into Weblate and selected using a translation flag (see Customizing behavior using flags).

Weblate font management tool in Fonts under the Manage menu of your translation project provides interface to upload and manage fonts. TrueType or OpenType fonts can be uploaded, set up font-groups and use those in the check.

The font-groups allow you to define different fonts for different languages, which is typically needed for non-latin languages:

![Font Management Interface](image)

The font-groups are identified by name, which can not contain whitespace or special characters, so that it can be easily used in the check definition:

```
Language: 
...........

Font: 
...........
```

2.12. Checks and fixups
Font-family and style is automatically recognized after uploading them:

You can have a number of fonts loaded into Weblate:
To use the fonts for checking the string length, pass it the appropriate flags (see Customizing behavior using flags). You will probably need the following ones:

- **max-size:500** Defines maximal width.
- **font-family:ubuntu** Defines font group to use by specifying its identifier.
- **font-size:22** Defines font size.

## 2.12.5 Writing own checks

A wide range of quality checks are built-in, (see Quality checks), though they might not cover everything you want to check. The list of performed checks can be adjusted using CHECK_LIST, and you can also add custom checks.

1. Subclass the `weblate.checks.Check`
2. Set a few attributes.
3. Implement either the `check` (if you want to deal with plurals in your code) or the `check_single` method (which does it for you).

Some examples:

To install custom checks, provide a fully-qualified path to the Python class in the CHECK_LIST, see Custom quality checks, addons and auto-fixes.

### Checking translation text does not contain “foo”

This is a pretty simple check which just checks whether the translation is missing the string “Foo”.

```python
# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
```

(sonraki sayfa devam)
```python
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
```

### Checking that Czech translation text plurals differ

Check using language info to verify the two plural forms in Czech language are not same.
class PluralCzechCheck(TargetCheck):
    # Used as identifier for check, should be unique
    # Has to be shorter than 50 characters
    check_id = "foo"

    # Short name used to display failing check
    name = "Foo check"

    # Description for failing check
    description = "Your translation is foo"

    # Real check code
    def check_target_unit(self, sources, targets, unit):
        if self.is_language(unit, ("cs",)):
            return targets[1] == targets[2]
        return False

    def check_single(self, source, target, unit):
        """We don't check target strings here."""
        return False

2.13 Makine çevirisi

Built-in support for several machine translation services and can be turned on by the administrator using `MT_SERVICES` for each one. They come subject to their terms of use, so ensure you are allowed to use them how you want.

The source language can be configured at Project configuration.

2.13.1 amaGama

Special installation of `tmserver` run by the authors of Virtaal.

Turn on this service by adding `weblate.machinery.tmserver.AmagamaTranslation` to `MT_SERVICES`.

Ayrıca bkz:
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`.

Ayrıca bkz:
`MT_APERTIUM_APY`, Apertium website, Apertium APy documentation
2.13.3 AWS

3.1 sürümünde geldi.

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.

Ayrıca bakınız:

- `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 sürümünde geldi.

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`.

Ayrıca bakınız:

- `MT_BAIDU_ID`, `MT_BAIDU_SECRET` Baidu Translate API

2.13.5 DeepL

2.20 sürümünde geldi.

DeepL is a 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`.

İpucu: In case you have subscription for CAT tools, you are supposed to use “v1 API” instead of default “v2” used by Weblate (it is not really an API version in this case). In case you are on a free instead of a paid plan, you have to use `https://api-free.deepl.com/` instead of `https://api.deepl.com/`. You can adjust both parameters by `MT_DEEPL_API_URL`.

Ayrıca bakınız:

- `MT_DEEPL_KEY`, `MT_DEEPL_API_URL`, DeepL website, DeepL pricing, DeepL API documentation
### 2.13.6 LibreTranslate

4.7.1 sürümünde geldi.

LibreTranslate is a free and open-source service for machine translations. The public instance requires an API key, but LibreTranslate can be self-hosted and there are several mirrors available to use the API for free.

Turn on this service by adding `weblate.machinery.libretranslate.LibreTranslateTranslation` to `MT_SERVICES` and set `MT_LIBRETRANSLATE_API_URL`. If your instance requires an API key, you must also set `MT_LIBRETRANSLATE_KEY`.

Ayrıca bakınız:

- `MT_LIBRETRANSLATE_KEY`, `MT_LIBRETRANSLATE_API_URL`, LibreTranslate website, LibreTranslate repository, LibreTranslate mirrors

### 2.13.7 Glosbe

Free dictionary and translation memory for almost every living language.

The API is gratis to use, but subject to the used data source license. There is a limit of calls that may be done from one IP in a set period of time, to prevent abuse.

Turn on this service by adding `weblate.machinery.glosbe.GlosbeTranslation` to `MT_SERVICES`.

Ayrıca bakınız:

- Glosbe website

### 2.13.8 Google Translate

Machine translation service provided by Google.

This service uses the Google Translation API, and you need to obtain an API key and turn on billing in the Google API console.

To turn on this service, add `weblate.machinery.google.GoogleTranslation` to `MT_SERVICES` and set `MT_GOOGLE_KEY`.

Ayrıca bakınız:

- `MT_GOOGLE_KEY`, Google translate documentation

### 2.13.9 Google Translate API V3 (Advanced)

Machine translation service provided by Google Cloud services.

This service differs from the former one in how it authenticates. To enable service, add `weblate.machinery.googlev3.GoogleV3Translation` to `MT_SERVICES` and set

- `MT_GOOGLE_CREDENTIALS`
- `MT_GOOGLE_PROJECT`

If `location` fails, you may also need to specify `MT_GOOGLE_LOCATION`.

Ayrıca bakınız:

- `MT_GOOGLE_CREDENTIALS`, `MT_GOOGLE_PROJECT`, `MT_GOOGLE_LOCATION` Google translate documentation
2.13.10 Microsoft Cognitive Services Translator

2.10 sürümünde geldi.

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.

Ayrıca bakınız:
MT_MICROSOFT_COGNITIVE_KEY, MT_MICROSOFT_REGION, Cognitive Services - Text Translation API, Microsoft Azure Portal

2.13.11 Microsoft Terminology Service

2.19 sürümünde geldi.

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.

Ayrıca bakınız:
Microsoft Terminology Service API

2.13.12 ModernMT

4.2 sürümünde geldi.

Turn this service on by adding weblate.machinery.modernmt.ModernMTTranslation to MT_SERVICES and configure MT_MODERNMT_KEY.

Ayrıca bakınız:
ModernMT API, MT_MODERNMT_KEY, MT_MODERNMT_URL
2.13.13 MyMemory

Huge translation memory with machine translation.

Free, anonymous usage is currently limited to 100 requests/day, or to 1000 requests/day when you provide a contact e-mail address in `MT_MYMEMORY_EMAIL`. You can also ask them for more.

Turn on this service by adding `weblate.machinery.mymemory.MyMemoryTranslation` to `MT_SERVICES` and set `MT_MYMEMORY_EMAIL`.

Ayrıca bakınız:

`MT_MYMEMORY_EMAIL, MT_MYMEMORY_USER, MT_MYMEMORY_KEY`, MyMemory website

2.13.14 NetEase Sight API machine translation

3.3 sürümünde geldi.

NetEase tarafından sağlanan makine çevirisini hizmeti.

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`.

Ayrıca bakınız:

`MT_NETEASE_KEY, MT_NETEASE_SECRET` NetEase Sight Translation Platform

2.13.15 tmserver

You can run your own translation memory server by using the one bundled with Translate-toolkit and let Weblate talk to it. You can also use it with an amaGama server, which is an enhanced version of tmserver.

1. First you will want to import some data to the translation memory:

2. Turn on this service by adding `weblate.machinery.tmserver.TMServerTranslation` to `MT_SERVICES`.

   ```
   build_tmdb -d /var/lib/tm/db -s en -t cs locale/cs/LC_MESSAGES/django.po
   build_tmdb -d /var/lib/tm/db -s en -t de locale/de/LC_MESSAGES/django.po
   build_tmdb -d /var/lib/tm/db -s en -t fr locale/fr/LC_MESSAGES/django.po
   ```

3. Start tmserver to listen to your requests:

   ```
   tmserver -d /var/lib/tm/db
   ```

4. Configure Weblate to talk to it:

   ```
   MT_TMSERVER = "http://localhost:8888/tmserver/
   ```

Ayrıca bakınız:

`MT_TMSERVER`, tmserver Installing amaGama, Amagama, Amagama Translation Memory
2.13.16 Yandex Translate

Machine translation service provided by Yandex.
This service uses a Translation API, and you need to obtain an API key from Yandex.
Turn on this service by adding `weblate.machinery.yandex.YandexTranslation` to `MT_SERVICES`, and set `MT_YANDEX_KEY`.

Ayrıca bakınız:
`MT_YANDEX_KEY`, Yandex Translate API, Powered by Yandex.Translate

2.13.17 Youdao Zhiyun API machine translation

3.2 sürümünde geldi.
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`.

Ayrıca bakınız:
`MT_YOUDAO_ID, MT_YOUDAO_SECRET` Youdao Zhiyun Natural Language Translation Service

2.13.18 Weblate

Weblate can be the source of machine translations as well. It is based on the Woosh fulltext engine, and provides both exact and inexact matches.
Turn on these services by adding `weblate.machinery.weblatetm.WeblateTranslation` to `MT_SERVICES`.

2.13.19 Weblate Translation Memory

2.20 sürümünde geldi.
The Çeviri Belleği can be used as a source for machine translation suggestions as well.
Turn on these services by adding `weblate.memory.machine.WeblateMemory` to the `MT_SERVICES`. This service is turned on by default.

2.13.20 SAP Translation Hub

Machine translation service provided by SAP.
You need to have a SAP account (and the SAP Translation Hub enabled in the SAP Cloud Platform) to use this service.
Turn on this service by adding `weblate.machinery.saptrnslationhub.SAPTranslationHub` to `MT_SERVICES` and set the appropriate access to either the sandbox or the production API.

Not: 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`. 
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 the `dictionary` Python module:

```python
# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.
#
"""Machine translation example."""
import dictionary
from weblate.machinery.base import MachineTranslation

class SampleTranslation(MachineTranslation):
    """Sample machine translation interface."""
    name = "Sample"

def download_languages(self):
    """Return list of languages your machine translation supports."""
    return {"cs"}

def download_translations(self, source, language, text: str, unit, user,
                           search: bool, threshold: int = 75,):
    """Return tuple with translations."""
    for t in dictionary.translate(text):
        yield {"text": t, "quality": 100, "service": self.name, "source": text}
```

You can list your own class in `MT_SERVICES` and Weblate will start using that.

2.13.21 Custom machine translation
2.14 Eklentiler

2.19 sürümünde geldi.

Eklentiler, çeviri iş akışını özelleştirmenin ve otomatikleştirmenin yollarını sağlar. Yöneticiler, her bir çeviri bileşeninin Yönet ↓ Eklentiler menüsünden ekleni ekleyebilir ve onları yönetebilir.

İpucu: You can also configure add-ons using API, DEFAULT_ADDONS, or install_addon.
## 2.14. Add-ons

### Available add-ons

<table>
<thead>
<tr>
<th>Add-on</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Automatic translation</strong></td>
<td>Automatically translates strings using machine translation or other components.</td>
</tr>
<tr>
<td><strong>Add missing languages</strong></td>
<td>Ensures a consistent set of languages is used for all components within a project.</td>
</tr>
<tr>
<td><strong>Component discovery</strong></td>
<td>Automatically adds or removes project components based on file changes in the version control system.</td>
</tr>
<tr>
<td><strong>Bulk edit</strong></td>
<td>Bulk edit flags, labels, or states of strings.</td>
</tr>
<tr>
<td><strong>Statistics generator</strong></td>
<td>Generates a file containing detailed info about the translation status.</td>
</tr>
<tr>
<td><strong>Pseudolocale generation</strong></td>
<td>Generates a translation by adding prefix and suffix to source strings automatically.</td>
</tr>
<tr>
<td><strong>Contributors in comment</strong></td>
<td>Updates the comment part of the PO file header to include contributor names and years of contributions.</td>
</tr>
<tr>
<td><strong>Customize gettext output</strong></td>
<td>Allows customization of gettext output behavior, for example line wrapping.</td>
</tr>
<tr>
<td><strong>Generate MO files</strong></td>
<td>Automatically generates a MO file for every changed PO file.</td>
</tr>
<tr>
<td><strong>Update PO files to match POT (msgmerge)</strong></td>
<td>Updates all PO files (as configured by &quot;Filenames&quot;) to match the POT file (as configured by &quot;Template for new translations&quot;) using msgmerge.</td>
</tr>
<tr>
<td><strong>Squash Git commits</strong></td>
<td>Squash Git commits prior to pushing changes.</td>
</tr>
<tr>
<td><strong>Stale comment removal</strong></td>
<td>Set a timeframe for removal of comments.</td>
</tr>
<tr>
<td><strong>Stale suggestion removal</strong></td>
<td>Set a timeframe for removal of suggestions.</td>
</tr>
</tbody>
</table>

Some add-ons will ask for additional configuration during installation.
2.14.1 Yerleşik eklentiler

Kendiliğinden çeviri

3.9 sürümünde geldi.

**Eklenti kimliği** weblate.autotranslate.autotranslate

**Configuration**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>Otomatik çeviri kipi Kullanabilir seçenekler: suggest – Öneri olarak ekle, translate – Çeviri olarak ekle, fuzzy – Düzenleme ihtiyacı var olarak ekle</td>
</tr>
<tr>
<td>auto_source</td>
<td>Otomatik çeviri kaynağı Kullanabilir seçenekler: others – Diğer çeviri bileşenleri, mt – Makine çevirisı</td>
</tr>
<tr>
<td>component</td>
<td>Bileşenler Projenin her bileşenlerine erişebilmesi için paylaşılın çeviri belleğine katkı için.</td>
</tr>
<tr>
<td>engines</td>
<td>Makine çevirisi motorları</td>
</tr>
<tr>
<td>threshold</td>
<td>Skor eşiği</td>
</tr>
</tbody>
</table>

**Triggers** bileşen güncellenmesi, günlük

Makine çevirisini veya diğer bileşenleri kullanarak dizgileri kendiliğinden çevirir.

Ne zaman tetiklenir:

- Bileşende yeni dizgiler görüldüğünde.
- Once in a month for every component, this can be configured using `BACKGROUND_TASKS`.

Ayrıca bakınız:

*Kendiliğinden çeviri, Keeping translations same across components*

**JavaScript yerelleştirme CDN’i**

4.2 sürümünde geldi.

**Eklenti kimliği** weblate.cdn.cdnjs

**Configuration**
<table>
<thead>
<tr>
<th><strong>thres-hold</strong></th>
<th>Çeviri eşği</th>
<th>Çevirilerin dahl edilmesi için eşik.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>css_selector</strong></td>
<td>CSS seçici</td>
<td>Yerelleştirilebilir öğeleri almak için CSS seçici.</td>
</tr>
<tr>
<td><strong>co-okie_name</strong></td>
<td>Dil tanımlama bilgisi adı</td>
<td>Dil tercîhi saklayan tanımlama bilgisi adı.</td>
</tr>
<tr>
<td><strong>files</strong></td>
<td>HTML dosyalarından dizgileri çıkar</td>
<td>Çevirilebilir dizgiler için ayrıştıracak şu anki depodaki veya uzak URL’lerdeki dosya adlarının listesi.</td>
</tr>
</tbody>
</table>

**Triggers** daily, repository post-commit, repository post-update

JavaScript veya HTML yerelleştirmesinde kullanmak üzere içerik dağıtımına ağlarına çevirileri yayılacak.

Statik HTML sayfalarını yerelleştirmek veya JavaScript kodunda yerelleştirmeyi yüklemek için kullanılabilir.

Bileşenizin için, bunları yerelleştirmek için HTML sayfalara ekleyebileceğiniz benzersiz bir URL oluşturur. Daha fazla ayrıntı için bkz: weblate-cdn.

**Ayrıca bakınız:**

cdn-addon-config, weblate-cdn, cdn-addon-extract, cdn-addon-html

**Boş dizgileri kaldır**

4.4 sürümündede geldi.

**Eklenti kimliği** weblate.cleanup.blank

**Configuration** Bu eklentinin yapılandırılması yok.

**Triggers** repository post-commit, repository post-update

Çeviri dosyalarından çevirisi olmadan dizgileri kaldırır.

Çeviri dosyalarında boş dizgilerin görünmemesi için bunu kullanın (örneğin yerelleştirme kütüphanınız kaynak diz-giyi kullanmanız yerine bunları boş dizgiler olarak görüntüleyorsa).

**Ayrıca bakınız:**

*Does Weblate update translation files besides translations?*

**Çeviri dosyalarını temizle**

**Eklenti kimliği** weblate.cleanup.generic

**Configuration** Bu eklentinin yapılandırılması yok.

**Triggers** repository pre-commit, repository post-update

Tüm çeviri dosyalarını tek dilli taban dosyayla eşleşmesi için güncelle. Çoğu dosya biçimi için bu, taban dosyada artık bulunmayan eski çeviri anahtarlarını kaldırmak anlamına gelir.

**Ayrıca bakınız:**

*Does Weblate update translation files besides translations?*
Eksik dilleri ekle

Eklenti kimliği weblate.consistency.languages

**Configuration** Bu eklentinin yapılandırması yok.

**Triggers** daily, repository post-add

Proje içindeki tüm bileşenler için tutarlı bir dil kümesinin kullanılmasını sağlar.

Eksik diller her 24 saatte bir ve Weblate’e yeni diller eklenmesi de denetlenir.

Diğerlerinden farklı olarak, bu eklenti tüm projeyi etkiler.

İpucu: Yeni eklenen dizgileri Kendiliğinden çeviri ile otomatik olarak çevirin.

Bileşen keﬁşti

Eklenti kimliği weblate.discovery.discovery

**Configuration**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>match</td>
<td>Çeviri dosyalara karşı eşleştirmek için düzenli ifade</td>
</tr>
<tr>
<td>file_format</td>
<td>Dosya biçimi</td>
</tr>
<tr>
<td>name_template</td>
<td>Bileşen adına özelleştir</td>
</tr>
<tr>
<td>base_file_template</td>
<td>Tek dil taban dosya adımı tanımla</td>
</tr>
<tr>
<td>new_base_template</td>
<td>Yeni çeviriler için taban dosya tanımla</td>
</tr>
<tr>
<td>language_regex</td>
<td>Dil süzgeci</td>
</tr>
<tr>
<td>copy_addons</td>
<td>Ana bileşenden yeni oluşturulanlara ekleri çoğalt</td>
</tr>
<tr>
<td>remove</td>
<td>Varolmayan dosyalar için bileşenleri kaldı</td>
</tr>
<tr>
<td>confirm</td>
<td>Yukarıdaki eşleşme lerin doğru olduğunu onaylıyorum</td>
</tr>
</tbody>
</table>

**Triggers** depo güncellenmesinden sonra

Stirüm denetim sistemindeki dosya değişikliklerine göre proje bileşenlerini kendiliğinden ekler veya kaldırır.

Triggered each time the VCS is updated, and otherwise similar to the **import_project** management command. This way you can track multiple translation components within one VCS.

The matching is done using regular expressions enabling complex configuration, but some knowledge is required to do so. Some examples for common use cases can be found in the add-on help section.

Once you hit **Save**, a preview of matching components will be presented, from where you can check whether the configuration actually matches your needs:
2.14. Eklentiler

The Weblate Manual, Yayım 4.9
İpucu: Component discovery add-on uses Weblate internal URLs. It’s a convenient way to share VCS setup between multiple components. Linked components use the local repository of the main component set up by filling weblate://project/main-component into the Kaynak kod deposu field (in Manage ↓ Settings ↓ Version control system) of each respective component. This saves time with configuration and system resources too.

Ayrıca bakınız:

Template markup

Toplu düzenlene

3.11 sürümünde geldi.

Eklenti kimliği weblate.flags.bulk

Configuration

<table>
<thead>
<tr>
<th></th>
<th>Sorgu</th>
<th>Kullanılabilir seçenekler:</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>state</td>
<td>-1 – Değiştirmeyin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 – Düzenleme gerekli</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 – Çevrildi</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30 – Onaylandı</td>
</tr>
<tr>
<td>add_flags</td>
<td>Eklenecek çeviri işaretleri</td>
<td></td>
</tr>
<tr>
<td>remo-ve_flags</td>
<td>Kaldırılacak çeviri işaretleri</td>
<td></td>
</tr>
<tr>
<td>add_labels</td>
<td>Eklenecek etiketler</td>
<td></td>
</tr>
<tr>
<td>remo-ve_labels</td>
<td>Kaldırılacak etiketler</td>
<td></td>
</tr>
</tbody>
</table>

Triggers bileşen güncellemesi

Dizgilerin işaretlerini, etiketlerini veya durumlarını toplu düzenleyin.

NOT has:label arama sorgusuya başlayarak etiketlemeyi otomatikleştirelim ve tüm dizgiler gerekli tüm etiketlere sahip olana kadar etiketler ekleyin. Weblate üst verileri için diğer otomatik işlemler de yapılabilir.

Examples:

<table>
<thead>
<tr>
<th>Arama sorgusu</th>
<th>Eklenecek etiketler</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT has:label</td>
<td>en son</td>
</tr>
</tbody>
</table>

Tablo 4: Label new strings automatically

<table>
<thead>
<tr>
<th>Arama sorgusu</th>
<th>Eklenecek çevirisi işaretleri</th>
</tr>
</thead>
<tbody>
<tr>
<td>language:en AND key:changelogs/</td>
<td>read-only</td>
</tr>
</tbody>
</table>

Tablo 5: Marking all Uygulama mağazası üst veri dosyaları changelog entries read-only

Ayrıca bakınız:

Toplu düzenleme, Customizing behavior using flags, labels
Değiştirilmemiş çevirileri “Düzenleme gerekli” olarak işaretle

3.1 sürümünde geldi.

**Eklenti kimliği** weblate.flags.same_edit

**Configuration** Bu eklentinin yapılandırması yok.

**Triggers** unit post-create

VCS'den yeni bir çevrilebilir dizgi içe aktarıldığında ve bir kaynak dizgiyle eşleştiğinde, Weblate'te düzenleme gerekli olarak işaretlenir. Özellikle çevrilmemiş dizgiler için kaynak dizgileri içeren dosya biçimleri için kullanılmıştır.

**İpuçu:** You might also want to tighten the *Değiştirilmemiş çeviri* check by adding strict-same flag to Çeviri işaretleri.

**Ayrıca bakınız:**
Translation states

### Yeni kaynak dizgileri “Düzenleme gerekli” olarak işaretle

**Eklenti kimliği** weblate.flags.source_edit

**Configuration** Bu eklentinin yapılandırması yok.

**Triggers** unit post-create

Yeni bir kaynak dizgi VCS'den içe her aktarıldığında, Weblate'de düzenleme gerekli olarak işaretlenir. Bu yolla geliştiriciler tarafından yazılan kaynak dizgiler kolayca sızdırılabilir ve düzenleyebilirsiniz.

**Ayrıca bakınız:**
Translation states

### Yeni çevirileri “Düzenleme gerekli” olarak işaretle

**Eklenti kimliği** weblate.flags.target_edit

**Configuration** Bu eklentinin yapılandırması yok.

**Triggers** unit post-create

Yeni bir çevrilebilir dizgi, VCS'den içeri her aktarıldığında, Weblate'de düzenleme gerekli olarak işaretlenir. Bu yolla geliştiricilerin oluşturduğu çevirileri kolayca sızdır ve düzenleyebilirsiniz.

**Ayrıca bakınız:**
Translation states

### İstatistik oluşturuğu

**Eklenti kimliği** weblate.generate.generate

**Configuration**

| filename | Oluşturulan dosyanın adı |
| template | Oluşturulan dosyanın içeriği |

**Triggers** depo işlenmesinden önce

Çeviri durumu hakkında ayrıntılı bilgi içeren bir dosya oluşturur.

Hem dosya adında hem de içeriğe bir Django şablonu kullanabilirsiniz, ayrıntılı bir işaretleme açıklaması için Template markup bölümüne bakın.

Örneğin, her çeviri için bir özet dosyası oluşturma:

**Oluşturulan dosyanın adı** `locale/{{ language_code }}.json`

**İçerik**

```json
{
    "language": "{{ language_code }}",
    "strings": "{{ stats.all }}",
    "translated": "{{ stats.translated }}",
    "last_changed": "{{ stats.last_changed }}",
    "last_author": "{{ stats.last_author }}"
}
```

Ayrıca bakınız:

Template markup

**Sözde yerel oluşturma**

**Eklenti kimliği** `weblate.generate.pseudolocale`

**Configuration**

<table>
<thead>
<tr>
<th>source</th>
<th>Kaynak dizgiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>Hedef çeviri</td>
</tr>
<tr>
<td>prefix</td>
<td>Dizgi öncesi</td>
</tr>
<tr>
<td>suffix</td>
<td>Dizgi sonesi</td>
</tr>
</tbody>
</table>

**Triggers** bileşen güncellenmesi, günlük

Kaynak dizgileri otomatik olarak örnek ve sonkekleyerek bir çeviri oluşturur.

Pseudolocales are useful to find strings that are not prepared for localization. This is done by altering all translatable source strings to make it easy to spot unaltered strings when running the application in the pseudolocale language.

Finding strings whose localized counterparts might not fit the layout is also possible.

**İpucu:** You can use real languages for testing, but there are dedicated pseudolocales available in Weblate - en_XA and ar_XB.

**İpucu:** You can use this add-on to start translation to a new locale of an existing language or similar language. Once you add the translation to the component, follow to the add-on. Example: If you have fr and want to start fr_CA translation, simply set fr as the source, fr_CA as the target, and leave the prefix and suffix blank.

Uninstall the add-on once you have the new translation filled to prevent Weblate from changing the translations made after the copying.
Yorum içindeki katılımcılar

**Eklenti kimliği** weblate.gettext.authors

**Configuration** Bu eklentinin yapılandırılması yok.

**Triggers** depo işlenmesinden önce

PO dosyası başlığının yorum bölümünü, katılımcı adlarını ve katkı yıllarını içerecek şekilde günceller.

The PO file header will look like this:

```plaintext
# Pavel Borecki <pavel@example.com>, 2018, 2019.
# Filip Hron <filip@example.com>, 2018, 2019.
# anonymous <noreply@weblate.org>, 2019.
```

"configure" dosyasındaki ALL_LINGUAS değişkenini güncelle

**Eklenti kimliği** weblate.gettext.configure

**Configuration** Bu eklentinin yapılandırılması yok.

**Triggers** depo eklenmesinden sonra, günlük

Yeni bir çeviri eklediğinde ALL_LINGUAS değişkenini configure, configure.in veya herhangi bir configure.ac dosyasında günceller.

Gettext çıktısını özelleştir

**Eklenti kimliği** weblate.gettext.customize

**Configuration**

<table>
<thead>
<tr>
<th>width</th>
<th>Uzun satırları</th>
<th>Varsayılan olarak gettext satırları 77 karakter ve yeni satırlarda kaydırır, --no-wrap parametresi ile kaydırma yalnızca yeni satırlarda yapılır.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>kaydırma yok</td>
<td>-1 Satır kaydırma yok</td>
</tr>
<tr>
<td>65535</td>
<td>Satırları 77 karakter ve yeni satırlarda kaydır</td>
<td>65535 – Satırları yalnızca yeni satırlarda kaydır</td>
</tr>
<tr>
<td>77</td>
<td>Satırları, 77 karakter de yeni satırlarda kaydır</td>
<td>77 – Satırları, 77 karakterde ve yeni satırlarda kaydır</td>
</tr>
</tbody>
</table>

**Triggers** storage post-load

Gettext çıktısın davranışının净资产行われに許可を、例えば行コードが許可される。

Aşağıdaki seçenekleri sunar:

- Satırları, 77 karakterde ve yeni satırlarda kaydır
- Satırları yalnızca yeni satırlarda kaydır
- Satır kaydırma yok

**Not:** Varsayılan olarak gettext satırları 77 karakterde ve yeni satırlarda kaydırır. --no-wrap parametresi ile kaydırma yalnızca yeni satırlarda yapılır.
LINGUAS dosyasını güncelle

Eklenti kimliği: weblate.gettext.linguas
Configuration: Bu eklentinin yapılandırması yok.
Triggers: depo eklennmesinden sonra, günlük

Yeni bir çeviri eklendiğinde LINGUAS dosyası günceller.

MO dosyaları üret

Eklenti kimliği: weblate.gettext.mo
Configuration:

| path | Oluşturulan MO dosyasının yolu | Belirtilmemişse, PO dosyasının konumu kullanılmaktır. |

Triggers: depo işlenmesinden önce

Her değiştirilen PO dosyası için kendiliğinden bir MO dosyası üretir.

The location of the generated MO file can be customized and the field for it uses *Template markup.*

POT ile eşleşmesi için PO dosyalarını güncelle (msgmerge)

Eklenti kimliği: weblate.gettext.msgmerge
Configuration:

| previous | Çevrilen dizgilerin önceki msgid’lerini sakla |
| no_location | Çevrilen dizgilerin konumlarını kaldır |
| fuzzy | Belirsiz eşleşmeyi kullan |

Triggers: depo güncellenmesinden sonra

Tüm PO dosyalarını (*File mask* ile yapılandırıldıgı gibi) *msgmerge* komutunu kullanarak POT dosyasıyla eşleşecek şekilde (*Yeni çeviri için şablon* ile yapılandırıldıgı gibi) günceller.

Yukarı aksş deposundan yeni değişiklikler çekildiğinde tetiklenir. Eklenti yapılandırması aracılığıyla msgmerge komut satırı seçeneklerinin çoğu yapılandırılabilir.

Ayrıca bakınız:

*Does Weblate update translation files besides translations?*

**Git işlemlerini sıkıtır**

Eklenti kimliği: weblate.git.squash
Configuration:
**Triggers** depo işlenmesinden sonra

Değişiklikleri yollamadan önce Git işlemlerini sıkıştırın.

Git commits can be squashed prior to pushing changes in one of the following modes:

3.4 sürümünde geldi.

- Tüm işlemler bir arada
- Dil başına
- Dosya başına

3.5 sürümünde geldi.

- Hazırlayan başına

Orijinal işleme iletüleri tutulur, ancak *Hazırlayan başına* seçilmediği veya işleme iletisi eklemek üzere özellleştirilmiş sürece yazarsız kaybedilir.

4.1 sürümünde geldi.

The original commit messages can optionally be overridden with a custom commit message.

Trailers (commit lines like `Co-authored-by: ...`) can optionally be removed from the original commit messages and appended to the end of the squashed commit message. This also generates proper `Co-authored-by: credit` for every translator.

**JSON çıktısını özelleştir**

**Eklenti kimliği** `weblate.json.customize`

**Configuration**

<table>
<thead>
<tr>
<th>sort_keys</th>
<th>JSON anahtarlarını sırala</th>
</tr>
</thead>
<tbody>
<tr>
<td>indent</td>
<td>JSON girintisi</td>
</tr>
<tr>
<td>style</td>
<td>JSON girinti stili</td>
</tr>
<tr>
<td></td>
<td>Kullanılabilir seçenekler:</td>
</tr>
<tr>
<td></td>
<td>spaces – Spaces</td>
</tr>
<tr>
<td></td>
<td>tabs – Tabs</td>
</tr>
</tbody>
</table>

**Triggers** `storage post-load`

JSON çıktısını davranışını ayarlanmasına izin verir, örneğin girinti veya sıralama.
Java özellikleri dosyasını biçimlendirir

Eklenti kimliği weblate.properties.sort
Configuration Bu eklentinin yapılandırması yok.
Triggers depo işlenmesinden önce
Java özellikleri dosyasını sıralar.

Eski yorum kaldırma

3.7 sürümünde geldi.

Eklenti kimliği weblate.removal.comments
Configuration

<table>
<thead>
<tr>
<th>age</th>
<th>Saklanacak günler</th>
</tr>
</thead>
</table>

Triggers daily

Yorumların kaldırılması için bir zaman dilimi ayarla.

Bu, güncelliğini yırtmış olabilecek eski yorumları kaldırmanın faydaları olabilir. Yorumun eski olması önemlerini yitirdiği anlamına gelmediği için dikkatli kullanın.

Eski öneri kaldırma

3.7 sürümünde geldi.

Eklenti kimliği weblate.removal.suggestions
Configuration

<table>
<thead>
<tr>
<th>age</th>
<th>Saklanacak günler</th>
</tr>
</thead>
</table>

| votes | Oy verme eşği | Kaldırma eşği. Bu alanın oylamının kapatılması ile etkisi yoktur. |

Triggers daily

Önerilerin kaldırılması için bir zaman dilimi ayarlayın.

Belirli bir zaman diliminde yeterince olumlu oy almayın önerileri kaldırmanız için öneri oylamaya (bkz: Peer review) bağıntılı olarak çok faydalı olabilir.

RESX dosyalarını güncelle

3.9 sürümünde geldi.

Eklenti kimliği weblate.resx.update
Configuration Bu eklentinin yapılandırması yok.
Triggers depo güncellenmesinden sonra

Tek dilli yukarı akış tabanlı dosya ile eşleştirme için tüm çeviri dosyalarını güncelleyin. Kullanılamayan dizgiler kaldırılır ve kaynak dizгинin kopyaları olarak yenileri eklenir.

İpucu: Yalnızca eski çeviri anahtarlarını kaldırmak istiyorsanız Çeviri dosyalarını temizle kullanın.
Ayrıca bakınız:

Does Weblate update translation files besides translations?

**YAML çıktısını özelleştir**

3.10.2 sürümünde geldi.

**Ekleni kimliği** weblate.yaml.customize

**Configuration**

<table>
<thead>
<tr>
<th>indent</th>
<th>YAML girişili</th>
<th>width</th>
<th>Uzun satırları kaydırma seçenekleri:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>80</td>
<td>Satırları, 80 karakterde kaydır</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>Satırları, 100 karakterde kaydır</td>
</tr>
<tr>
<td></td>
<td></td>
<td>120</td>
<td>Satırları, 120 karakterde kaydır</td>
</tr>
<tr>
<td></td>
<td></td>
<td>180</td>
<td>Satırları, 180 karakterde kaydır</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65535</td>
<td>Satır kaydırma yok</td>
</tr>
</tbody>
</table>

**line_break** Satır sonları seçenekleri:

<table>
<thead>
<tr>
<th></th>
<th>dos – DOS (\r\n)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>unix – UNIX (\n)</td>
</tr>
<tr>
<td></td>
<td>mac – MAC (\r)</td>
</tr>
</tbody>
</table>

**Triggers** storage post-load

YAML çıktısı davranışını ayarlanmasına izin verir, örneğin satır uzunluğu veya yeni satırlar.

### 2.14.2 Eklenti listesini özelleştirme

The list of add-ons is configured by `WEBLATE_ADDONS`. To add another add-on, simply include the absolute class name in this setting.

### 2.14.3 Eklenti yazma

You can write your own add-ons too, create a subclass of `weblate.addons.base.BaseAddon` to define the addon metadata, and then implement a callback to do the processing.

Ayrıca bakınız:

**Eklenti geliştirme**

### 2.14.4 Eklentiden betikleri çalıştırma

Add-ons can also be used to execute external scripts. This used to be integrated in Weblate, but now you have to write some code to wrap your script with an add-on.

```bash
# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
# This file is part of Weblate <https://weblate.org/>
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
```

(sonraki sayfa devam)
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 description
    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"

For installation instructions see Custom quality checks, addons and auto-fixes.

The script is executed with the current directory set to the root of the VCS repository for any given component.

Additionally, the following environment variables are available:

**WL_VCS**
- Version control system used.

**WL_REPO**
- Upstream repository URL.

**WL_PATH**
- Absolute path to VCS repository.

**WL_BRANCH**
- 2.11 sürümünde geldi.
  - Repository branch configured in the current component.

**WL_FILEMASK**
- Filemask for current component.

**WL_TEMPLATE**
- Filenname of template for monolingual translations (can be empty).

**WL_NEW_BASE**
- 2.14 sürümünde geldi.
  - Filenname of the file used for creating new translations (can be empty).
**WL_FILE_FORMAT**
Geçerli bileşende kullanılan dosya biçimi.

**WL_LANGUAGE**
Language of currently processed translation (not available for component-level hooks).

**WL_PREVIOUS_HEAD**
Previous HEAD after update (only available after running the post-update hook).

**WL_COMPONENT_SLUG**
Component slug used to construct URL.

**WL_PROJECT_SLUG**
Project slug used to construct URL.

**WL_COMPONENT_NAME**
Component name.

**WL_PROJECT_NAME**
Project name.

**WL_COMPONENT_URL**
Component URL.

**WL_ENGAGE_URL**
Project engage URL.

**Ayrıca bakınız:**

*Component configuration*

**Güncelleme sonrası depo işleme**

Can be used to update translation files when the VCS upstream source changes. To achieve this, please remember Weblate only sees files committed to the VCS, so you need to commit changes as a part of the script.

For example with Gulp you can do it using following code:

```bash
#!/bin/sh
gulp --gulpfile gulp-i18n-extract.js
git commit -m 'Update source strings' src/languages/en.lang.json
```

**Depoya işlenmeden önce çevirilerin işlemden geçirilmesi**

Use the commit script to automatically change a translation before it is committed to the repository.

It is passed as a single parameter consisting of the filename of a current translation.
2.15 Çeviri Belleği

2.20 sürümünde geldi.

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, Otomatik öneriler view while translating.
- Automatically, by translating strings using Kendiliğinden çeviri, or Kendiliğinden çeviri addon.

For installation tips, see Weblate Translation Memory, which is turned on by default.

2.15.1 Translation memory scopes

3.2 sürümünde geldi: 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.

Paylaşılan çeviri belleği

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 sürümünde geldi.

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.

İpucu: Translation memory in JSON can be imported into Weblate, TMX is provided for interoperability with other tools.

Ayrıca bakınız:

Weblate Çeviri Belleği Şeması

Management interface

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 Configuration

All settings are stored in settings.py (as is usual for Django).

Not: After changing any of these settings, you need to restart Weblate - both WSGI and Celery processes.

In case it is run as mod_wsgi, you need to restart Apache to reload the configuration.

Ayrıca bakınız:

Please also check Django's documentation for parameters configuring Django itself.
2.16.1 AKISMET_API_KEY

Weblate can use Akismet to check incoming anonymous suggestions for spam. Visit akismet.com to purchase an API key and associate it with a site.

2.16.2 ANONYMOUS_USER_NAME

Username of users that are not signed in.

Ayrıca bakınız:
Erişim denetimi

2.16.3 AUDITLOG_EXPIRY

3.6 sürümünde geldi.
How many days Weblate should keep audit logs, which contain info about account activity.
Defaults to 180 days.

2.16.4 AUTH_LOCK_ATTEMPTS

2.14 sürümünde geldi.
Maximum number of failed authentication attempts before rate limiting is applied.
This is currently applied in the following locations:

- Logins. Deletes the account password, preventing the user from signing in without requesting a new password.
- Password resets. Prevents new e-mails from being sent, avoiding spamming users with too many password reset attempts.

Defaults to 10.

Ayrıca bakınız:
Rate limiting

2.16.5 AUTO_UPDATE

3.2 sürümünde geldi.
3.11 sürümünde değişti: The original on/off option was changed to differentiate which strings are accepted.
Updates all repositories on a daily basis.

İpucu: Useful if you are not using Bildirim kancaları to update Weblate repositories automatically.

Not: On/off options exist in addition to string selection for backward compatibility.

Options are:

"none"  No daily updates.
"remote" also False Only update remotes.
"full" also True Update remotes and merge working copy.
Not: This requires that *Background tasks using Celery* is working, and will take effect after it is restarted.

### 2.16.6 AVATAR_URL_PREFIX

Prefix for constructing avatar URLs as: \( \text{${AVATAR_URL_PREFIX}/avatar/${MAIL_HASH}?${PARAMS}} \). The following services are known to work:

- **Gravatar (default), as per** https://gravatar.com/
  
  \[
  \text{AVATAR_URL_PREFIX} = \text{'https://www.gravatar.com/'}
  \]

- **Libravatar, as per** https://www.libravatar.org/
  
  \[
  \text{AVATAR_URL_PREFIX} = \text{'https://www.libravatar.org/'}
  \]

Ayrıca bakınız:

*Avatar ön bellekleme, ENABLE_AVATARS, Avatars*

### 2.16.7 AUTH_TOKEN_VALID

2.14 sürümünde geldi.

How long the authentication token and temporary password from password reset e-mails is valid for. Set in number of seconds, defaulting to 172800 (2 days).

### 2.16.8 AUTH_PASSWORD_DAYS

2.15 sürümünde geldi.

How many days using the same password should be allowed.

Not: Password changes made prior to Weblate 2.15 will not be accounted for in this policy.

Defaults to 180 days.

### 2.16.9 AUTOFIX_LIST

List of automatic fixes to apply when saving a string.

Not: Provide a fully-qualified path to the Python class that implementing the autofixer interface.

Available fixes:

- **weblate.trans.autofixes.whitespace.SameBookendingWhitespace** Matches whitespace at the start and end of the string to the source.

- **weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis** Replaces trailing dots (...) if the source string has a corresponding ellipsis (...).

- **weblate.trans.autofixes.chars.RemoveZeroSpace** Removes zero-width space characters if the source does not contain any.

- **weblate.trans.autofixes.chars.RemoveControlChars** Removes control characters if the source does not contain any.

- **weblate.trans.autofixes.html.BleachHTML** Removes unsafe HTML markup from strings flagged as safe-html (see *Güvenli olmayan HTML*).
You can select which ones to use:

```python
AUTOFIX_LIST = {
    "weblate.trans.autofixes.whitespace.SameBookendingWhitespace",
    "weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis",
}
```

Ayrıca bakınız:

*Automatic fixups, Custom automatic fixups*

### 2.16.10 BACKGROUND_TASKS

4.5.2 sürümünde geldi.

Defines how often lengthy maintenance tasks should be triggered for a component.

Right now this controls:

- *Kendiliğinden çeviri* eklemesi
- *Checks and fixups* recalculation

Olası seçenekler:

- monthly (this is the default)
- weekly
- daily
- never

**Not:** Increasing the frequency is not recommended when Weblate contains thousands of components.

### 2.16.11 BASE_DIR

Base directory where Weblate sources are located. Used to derive several other paths by default:

- *DATA_DIR*

Default value: Top level directory of Weblate sources.

### 2.16.12 BASIC_LANGUAGES

4.4 sürümünde geldi.

List of languages to offer users for starting new translation. When not specified built-in list is used which includes all commonly used languages, but without country specific variants.

This only limits non privileged users to add unwanted languages. The project admins are still presented with full selection of languages defined in Weblate.

**Not:** This does not define new languages for Weblate, it only filters existing ones in the database.

**Example:**

```
BASIC_LANGUAGES = {"cs", "it", "ja", "en"}
```
Ayrıca bakınız:

Dil tanımları

2.16.13 CSP_SCRIPT_SRC, CSP_IMG_SRC, CSP_CONNECT_SRC, CSP_STYLE_SRC, CSP_FONT_SRC

Customize Content-Security-Policy header for Weblate. The header is automatically generated based on enabled integrations with third-party services (Matomo, Google Analytics, Sentry,…). All these default to empty list.

Example:

```plaintext
# Enable Cloudflare Javascript optimizations
CSP_SCRIPT_SRC = ["ajax.cloudflare.com"]
```

Ayrıca bakınız:

Content security policy, Content Security Policy (CSP)

2.16.14 CHECK_LIST

List of quality checks to perform on a translation.

Not: Provide a fully-qualified path to the Python class implementing the check interface.

Adjust the list of checks to include ones relevant to you.

All built-in Quality checks are turned on by default, from where you can change these settings. By default they are commented out in Sample configuration so that default values are used. New checks then carried out for each new Weblate version.

You can turn off all checks:

```plaintext
CHECK_LIST = ()
```

You can turn on only a few:

```plaintext
CHECK_LIST = (  
    "weblate.checks.chars.BeginNewlineCheck",  
    "weblate.checks.chars.EndNewlineCheck",  
    "weblate.checks.chars.MaxLengthCheck",  
)
```

Not: Changing this setting only affects newly changed translations, existing checks will still be stored in the database. To also apply changes to the stored translations, run updatechecks.

Ayrıca bakınız:

Quality checks, Customizing behavior using flags
2.16.15 COMMENT_CLEANUP_DAYS

3.6 sürümünde geldi.
Delete comments after a given number of days. Defaults to None, meaning no deletion at all.

2.16.16 COMMIT_PENDING_HOURS

2.10 sürümünde geldi.
Number of hours between committing pending changes by way of the background task.

Ayrıca bakınız:
Component configuration, İşlenecek değişikliklerin yaş, Running maintenance tasks, commit_pending

2.16.17 CONTACT_FORM

4.6 sürümünde geldi.
Configures how e-mail from the contact form is being sent. Choose a configuration that matches your mail server configuration.

"reply-to" The sender is used in as Reply-To, this is the default behaviour.
"from" The sender is used in as From. Your mail server needs to allow sending such e-mails.

2.16.18 DATA_DIR

The folder Weblate stores all data in. It contains links to VCS repositories, a fulltext index and various configuration files for external tools.
The following subdirectories usually exist:
home Home directory used for invoking scripts.
ssh SSH keys and configuration.
static Default location for static Django files, specified by STATIC_ROOT. See Serving static files.
The Docker container uses a separate volume for this, see Docker container volumes.
media Default location for Django media files, specified by MEDIA_ROOT. Contains uploaded screenshots, see Visual context for strings.
vcs Çeviriler için sürüm denetimi depoları.
backups Daily backup data, please check Dumped data for backups for details.
celery Celery scheduler data, see Background tasks using Celery.
fonts: User-uploaded fonts, see Managing fonts.

Not: This directory has to be writable by Weblate. Running it as uWSGI means the www-data user should have write access to it.
The easiest way to achieve this is to make the user the owner of the directory:

```
sudo chown www-data:www-data -R $DATA_DIR
```

Defaults to $BASE_DIR/data.
Ayrıca bakınız:

BASE_DIR, Filesystem permissions, Backing up and moving Weblate

2.16.19 DATABASE_BACKUP

3.1 sürümünde geldi.

Whether the database backups should be stored as plain text, compressed or skipped. The authorized values are:

- "plain"
- "compressed"
- "none"

Ayrıca bakınız:

Back up and moving Weblate

2.16.20 DEFAULT_ACCESS_CONTROL

3.3 sürümünde geldi.

The default access control setting for new projects:

0 Public
1 Protected
100 Private
200 Custom

Use Custom if you are managing ACL manually, which means not relying on the internal Weblate management.

Ayrıca bakınız:

Proje erişim kontrolü, Erişim denetimi

2.16.21 DEFAULT_AUTO_WATCH

4.5 sürümünde geldi.

Configures whether Automatically watch projects on contribution should be turned on for new users. Defaults to True.

Ayrıca bakınız:

Bildirimler

2.16.22 DEFAULT_RESTRICTED_COMPONENT

4.1 sürümünde geldi.

The default value for component restriction.

Ayrıca bakınız:

Restricted access, Grupların kapsamı
2.16.23 **DEFAULT_ADD_MESSAGE, DEFAULT_ADDON_MESSAGE, DEFAULT_COMMIT_MESSAGE, DEFAULT_DELETE_MESSAGE, DEFAULT_MERGE_MESSAGE**

Default commit messages for different operations, please check Component configuration for details.

**Ayrıca bakınız:**
Template markup, Component configuration, Commit, add, delete, merge and addon messages

### 2.16.24 **DEFAULT_ADDONS**

Default addons to install on every created component.

**Not:** This setting affects only newly created components.

Example:

```python
DEFAULT_ADDONS = {
    # Add-on with no parameters
    "weblate.flags.target_edit": {},
    # Add-on with parameters
    "weblate.autotranslate.autotranslate": {
        "mode": "suggest",
        "filter_type": "todo",
        "auto_source": "mt",
        "component": ",",
        "engines": ["weblate-translation-memory"],
        "threshold": "80",
    },
}
```

**Ayrıca bakınız:**
install_addon, Eklentiler, WEBLATE_ADDONS

### 2.16.25 **DEFAULT_COMMITER_EMAIL**

2.4 sürümünde geldi.
Commmitter e-mail address defaulting to noreply@weblate.org.

**Ayrıca bakınız:**
DEFAULT_COMMITER_NAME

### 2.16.26 **DEFAULT_COMMITER_NAME**

2.4 sürümünde geldi.
Commmitter name defaulting to Weblate.

**Ayrıca bakınız:**
DEFAULT_COMMITER_EMAIL
2.16.27 DEFAULT_LANGUAGE

4.3.2 sürümünde geldi.
Default source language to use for example in Kaynak dil.
Defaults to en. The matching language object needs to exist in the database.

Ayrıca bakınız:
Dil tanımları, Kaynak dil

2.16.28 DEFAULT_MERGE_STYLE

3.4 sürümünde geldi.
Merge style for any new components.
• rebase - default
• merge

Ayrıca bakınız:
Component configuration, Birleştirme stili

2.16.29 DEFAULT_SHARED_TM

3.2 sürümünde geldi.
Configures default value of Paylaşılan çeviri belleği kullan and Paylaşılan çeviri belleğine katkıda bulunun.

2.16.30 DEFAULT_TRANSLATION_PROPAGATION

2.5 sürümünde geldi.
Default setting for translation propagation, defaults to True.

Ayrıca bakınız:
Component configuration, Çeviri yaymaya izin ver

2.16.31 DEFAULT_PULL_MESSAGE

Title for new pull requests, defaulting to 'Update from Weblate'.

2.16.32 ENABLE_AVATARS

Whether to turn on Gravatar-based avatars for users. By default this is on.
Avatars are fetched and cached on the server, lowering the risk of leaking private info, speeding up the user experience.

Ayrıca bakınız:
Avatar önbellkleme, AVATAR_URL_PREFIX, Avatars
2.16.33 ENABLE_HOOKS

Whether to enable anonymous remote hooks.

Ayrıca bakınız:
Bildirim kancaları

2.16.34 ENABLE_HTTPS

Whether to send links to Weblate as HTTPS or HTTP. This setting affects sent e-mails and generated absolute URLs.

In the default configuration this is also used for several Django settings related to HTTPS - it enables secure cookies, toggles HSTS or enables redirection to HTTPS URL.

The HTTPS redirection might be problematic in some cases and you might hit issue with infinite redirection in case you are using a reverse proxy doing SSL termination which does not correctly pass protocol headers to Django. Please tweak your reverse proxy configuration to emit X-Forwarded-Proto or Forwarded headers or configure SECURE_PROXY_SSL_HEADER to let Django correctly detect the SSL status.

Ayrıca bakınız:
SESSION_COOKIE_SECURE, CSRF_COOKIE_SECURE, SECURE_SSL_REDIRECT, SECURE_PROXY_SSL_HEADER Set correct site domain

2.16.35 ENABLE_SHARING

Turn on/off the Share menu so users can share translation progress on social networks.

2.16.36 GET_HELP_URL

4.5.2 sürümünde geldi.

URL where support for your Weblate instance can be found.

2.16.37 GITLAB_CREDENTIALS

4.3 sürümünde geldi.

List for credentials for GitLab servers.

İpucu: Use this in case you want Weblate to interact with more of them, for single GitLab endpoint stick with GITLAB_USERNAME and GITLAB_TOKEN.

```python
GITLAB_CREDENTIALS = {
    "gitlab.com": {
        "username": "weblate",
        "token": "your-api-token",
    },
    "gitlab.example.com": {
        "username": "weblate",
        "token": "another-api-token",
    },
}
```
2.16.38 GITLAB_USERNAME

GitLab username used to send merge requests for translation updates.

Ayrıca bakınız:

GITLAB_CREDENTIALS, GitLab

2.16.39 GITLAB_TOKEN

4.3 sürümünde geldi.

Çeviri güncellemeleri için API çağrıları yapmak üzere kullanılan GitLab kişisel erişim belirteci.

Ayrıca bakınız:

GITLAB_CREDENTIALS, GitLab, GitLab: Personal access token

2.16.40 GITHUB_CREDENTIALS

4.3 sürümünde geldi.

List for credentials for GitHub servers.

**İpucu:** Use this in case you want Weblate to interact with more of them, for single GitHub endpoint stick with GITHUB_USERNAME and GITHUB_TOKEN.

```json
GITHUB_CREDENTIALS = {
    "api.github.com": {
        "username": "weblate",
        "token": "your-api-token",
    },
    "github.example.com": {
        "username": "weblate",
        "token": "another-api-token",
    },
}
```

2.16.41 GITHUB_USERNAME

GitHub username used to send pull requests for translation updates.

Ayrıca bakınız:

GITHUB_CREDENTIALS, GitHub

2.16.42 GITHUB_TOKEN

4.3 sürümünde geldi.

GitHub personal access token used to make API calls to send pull requests for translation updates.

Ayrıca bakınız:

GITHUB_CREDENTIALS, GitHub, Creating a GitHub personal access token
2.16.43 GOOGLE_ANALYTICS_ID

Google Analytics ID to turn on monitoring of Weblate using Google Analytics.

2.16.44 HIDE_REPO_CREDENTIALS

Hide repository credentials from the web interface. In case you have repository URL with user and password, Weblate will hide it when related info is shown to users.

For example instead of https://user:password@git.example.com/repo.git it will show just https://git.example.com/repo.git. It tries to clean up VCS error messages too in a similar manner.

**Not:** This is turned on by default.

2.16.45 HIDE_VERSION

4.3.1 sürümünde geldi.

Hides version information from unauthenticated users. This also makes all documentation links point to latest version instead of the documentation matching currently installed version.

Hiding version is recommended security practice in some corporations, but it doesn’t prevent attacker to figure out version by probing the behavior.

**Not:** Bu, varsayılan olarak kapalıdır.

2.16.46 IP_BEHIND_REVERSE_PROXY

2.14 sürümünde geldi.

Indicates whether Weblate is running behind a reverse proxy.

If set to True, Weblate gets IP address from a header defined by `IP_PROXY_HEADER`.

**Uyarı:** Ensure you are actually using a reverse proxy and that it sets this header, otherwise users will be able to fake the IP address.

**Not:** This is not on by default.

**Ayrıca bakınız:**

Running behind reverse proxy, Rate limiting, `IP_PROXY_HEADER`, `IP_PROXY_OFFSET`
2.16.47 IP_PROXY_HEADER

2.14 sürümünde geldi.
Indicates which header Weblate should obtain the IP address from when IP_BEHIND_REVERSE_PROXY is turned on.
Defaults to HTTP_X_FORWARDED_FOR.

Ayrıca bakınız:
Running behind reverse proxy, Rate limiting, SECURE_PROXY_SSL_HEADER, IP_BEHIND_REVERSE_PROXY, IP_PROXY_OFFSET

2.16.48 IP_PROXY_OFFSET

2.14 sürümünde geldi.
Indicates which part of IP_PROXY_HEADER is used as client IP address.
Depending on your setup, this header might consist of several IP addresses, (for example X-Forwarded-For: a, b, client-ip) and you can configure which address from the header is used as client IP address here.

Uyarı: Setting this affects the security of your installation, you should only configure it to use trusted proxies for determining IP address.

Defaults to 0.

Ayrıca bakınız:
Running behind reverse proxy, Rate limiting, SECURE_PROXY_SSL_HEADER, IP_BEHIND_REVERSE_PROXY, IP_PROXY_HEADER

2.16.49 LEGAL_URL

3.5 sürümünde geldi.
URL where your Weblate instance shows its legal documents.

İpucu: Useful if you host your legal documents outside Weblate for embedding them inside Weblate, please check Yasal for details.

Example:

LEGAL_URL = "https://weblate.org/terms/"

Ayrıca bakınız:
PRIVACY_URL
2.16.50 LICENSE_EXTRA

Additional licenses to include in the license choices.

**Not:** Each license definition should be tuple of its short name, a long name and an URL.

For example:

```python
LICENSE_EXTRA = [
    ("AGPL-3.0", "GNU Affero General Public License v3.0", "https://www.gnu.org/licenses/agpl-3.0-standalone.html"),
]
```

2.16.51 LICENSE_FILTER

4.3 sürümünde değişti: Setting this to blank value now disables license alert.

Filter list of licenses to show. This also disables the license alert when set to empty.

**Not:** This filter uses the short license names.

For example:

```python
LICENSE_FILTER = {"AGPL-3.0", "GPL-3.0-or-later"}
```

Following disables the license alert:

```python
LICENSE_FILTER = set()
```

**Ayrıca bakınız:**

alerts

2.16.52 LICENSE_REQUIRED

Defines whether the license attribute in Component configuration is required.

**Not:** This is off by default.

2.16.53 LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH

Whether the length of a given translation should be limited. The restriction is the length of the source string * 10 characters.

**İpucu:** Set this to False to allow longer translations (up to 10.000 characters) irrespective of source string length.

**Not:** Defaults to True.
2.16.54 LOCALIZE_CDN_URL and LOCALIZE_CDN_PATH

These settings configure the JavaScript yerelleştirme CDN'i addon. LOCALIZE_CDN_URL defines root URL where the localization CDN is available and LOCALIZE_CDN_PATH defines path where Weblate should store generated files which will be served at the LOCALIZE_CDN_URL.

İpucu: On Hosted Weblate, this uses https://weblate-cdn.com/.

Ayrıca bakınız:

JavaScript yerelleştirme CDN'i

2.16.55 LOGIN_REQUIRED_URLS

A list of URLs you want to require logging into. (Besides the standard rules built into Weblate).

İpucu: This allows you to password protect a whole installation using:

```
LOGIN_REQUIRED_URLS = (^/(.*)$),
REST_FRAMEWORK["DEFAULT_PERMISSION_CLASSES"] = [
    "rest_framework.permissions.IsAuthenticated"
]
```

İpucu: It is desirable to lock down API access as well, as shown in the above example.

Ayrıca bakınız:

REQUIRE_LOGIN

2.16.56 LOGIN_REQUIRED_URLS_EXCEPTIONS

List of exceptions for LOGIN_REQUIRED_URLS. If not specified, users are allowed to access the sign in page. Some of exceptions you might want to include:

```
LOGIN_REQUIRED_URLS_EXCEPTIONS = {
    r"/accounts/(.*)$",  # Required for sign in
    r"/static/(.*)$",  # Required for development mode
    r"/widgets/(.*)$",  # Allowing public access to widgets
    r"/data/(.*)$",  # Allowing public access to data exports
    r"/hooks/(.*)$",  # Allowing public access to notification hooks
    r"/api/(.*)$",  # Allowing access to API
    r"/js/i18n/$",  # JavaScript localization
}
```
2.16.57 MATOMO_SITE_ID

ID of a site in Matomo (formerly Piwik) you want to track.

Not: This integration does not support the Matomo Tag Manager.

Ayrıca bakınız: MATOMO_URL

2.16.58 MATOMO_URL

Full URL (including trailing slash) of a Matomo (formerly Piwik) installation you want to use to track Weblate use. Please check <https://matomo.org/> for more details.

İpucu: This integration does not support the Matomo Tag Manager.

For example:

```
MATOMO_SITE_ID = 1
MATOMO_URL = "https://example.matomo.cloud/
```

Ayrıca bakınız: MATOMO_SITE_ID

2.16.59 MT_SERVICES

3.0 sürümünde değişti: The setting was renamed from MACHINE_TRANSLATION_SERVICES to MT_SERVICES to be consistent with other machine translation settings.

List of enabled machine translation services to use.

Not: Many of the services need additional configuration like API keys, please check their documentation Makine çevirisi for more details.

İpucu: When using Docker container, this configuration is automatically generated based on provided API keys, see Machine translation settings.

```
MT_SERVICES = {
    "weblate.machinery.apertium.ApertiumAPYTranslation",
    "weblate.machinery.deepl.DeepLTranslation",
    "weblate.machinery.glosbe.GlosbeTranslation",
    "weblate.machinery.google.GoogleTranslation",
    "weblate.machinery.libretranslate.LibreTranslateTranslation",
    "weblate.machinery.microsoft.MicrosoftCognitiveTranslation",
    "weblate.machinery.microsoftterminology.MicrosoftTerminologyService",
    "weblate.machinery.mymemory.MyMemoryTranslation",
    "weblate.machinery.tmserver.AmagamaTranslation",
    "weblate.machinery.tmserver.TMServerTranslation",
    "weblate.machinery.yandex.YandexTranslation",
    "weblate.machinery.weblatetm.WeblateTranslation",
    "weblate.machinery.saptranslationhub.SAPTranslationHub",
}
```

(sonraki sayfaya devam)
Ayrıca bakınız:

Makine çevirisı, Otomatik öneriler

2.16.60 MT_APERTIUM_APY


Ayrıca bakınız:

Apertium, Makine çevirisı, Otomatik öneriler

2.16.61 MT_AWS_ACCESS_KEY_ID

Access key ID for Amazon Translate.

Ayrıca bakınız:

AWS, Makine çevirisı, Otomatik öneriler

2.16.62 MT_AWS_SECRET_ACCESS_KEY

API secret key for Amazon Translate.

Ayrıca bakınız:

AWS, Makine çevirisı, Otomatik öneriler

2.16.63 MT_AWS_REGION

Region name to use for Amazon Translate.

Ayrıca bakınız:

AWS, Makine çevirisı, Otomatik öneriler

2.16.64 MT_BAIDU_ID

Client ID for the Baidu Zhiyun API, you can register at https://api.fanyi.baidu.com/api/trans/product/index

Ayrıca bakınız:

Baidu API machine translation, Makine çevirisı, Otomatik öneriler
2.16.65 MT_BAIDU_SECRET

Client secret for the Baidu Zhiyun API, you can register at https://api.fanyi.baidu.com/api/trans/product/index

Ayrıca bakınız:
Baidu API machine translation, Makine çevirisı, Otomatik öneriler

2.16.66 MT_DEEPL_API_URL

4.7 sürümünde değişti: The full API URL is now configured to allow using the free plan. Before, it was only possible to configure the API version using MT_DEEPL_API_VERSION.

API URL to use with the DeepL service. At the time of writing, there is the v1 API as well as a free and a paid version of the v2 API.

https://api.deepl.com/v2/ (default in Weblate) Is meant for API usage on the paid plan, and the subscription is usage-based.

https://api-free.deepl.com/v2/ Is meant for API usage on the free plan, and the subscription is usage-based.

https://api.deepl.com/v1/ Is meant for CAT tools and is usable with a per-user subscription.

Previously Weblate was classified as a CAT tool by DeepL, so it was supposed to use the v1 API, but now is supposed to use the v2 API. Therefore it defaults to v2, and you can change it to v1 in case you have an existing CAT subscription and want Weblate to use that.

The easiest way to find out which one to use is to open an URL like the following in your browser:


Replace the XXX with your auth_key. If you receive a JSON object which contains “Bonjour”, you have the correct URL; if not, try the other three.

Ayrıca bakınız:
DeepL, Makine çevirisı, Otomatik öneriler

2.16.67 MT_DEEPL_KEY

API key for the DeepL API, you can register at https://www.deepl.com/pro.html

Ayrıca bakınız:
DeepL, Makine çevirisı, Otomatik öneriler

2.16.68 MT_LIBRETRANSLATE_API_URL

4.7.1 sürümünde geldi.

API URL for the LibreTranslate instance to use.

https://libretranslate.com/ (official public instance) Requires an API key to use outside of the website.

Mirrors are documented on the LibreTranslate GitHub repository, some of which can be used without authentication:
https://github.com/LibreTranslate/LibreTranslate#user-content-mirrors

Ayrıca bakınız:
LibreTranslate, Makine çevirisı, Otomatik öneriler
2.16.69 MT_LIBRETRANSLATE_KEY

4.7.1 sürümünde geldi.
API key for the LibreTranslate instance specified in MT_LIBRETRANSLATE_API_URL.

Ayrıca bakınız:
LibreTranslate, Makine çeviri, Otomatik öneriler

2.16.70 MT_GOOGLE_KEY

API key for Google Translate API v2, you can register at https://cloud.google.com/translate/docs

Ayrıca bakınız:
Google Translate, Makine çevirisi, Otomatik öneriler

2.16.71 MT_GOOGLE_CREDENTIALS

API v3 JSON credentials file obtained in the Google cloud console. Please provide a full OS path. Credentials are per service-account affiliated with certain project. Please check https://cloud.google.com/docs/authentication/getting-started for more details.

2.16.72 MT_GOOGLE_PROJECT

Google Cloud API v3 project id with activated translation service and billing activated. Please check https://cloud.google.com/appengine/docs/standard/nodejs/building-app/creating-project for more details

2.16.73 MT_GOOGLE_LOCATION

API v3 Google Cloud App Engine may be specific to a location. Change accordingly if the default global fallback does not work for you.

Please check https://cloud.google.com/appengine/docs/locations for more details

Ayrıca bakınız:
Google Translate API V3 (Advanced)

2.16.74 MT_MICROSOFT_BASE_URL

Region base URL domain as defined in the “Base URLs” section.
Defaults to api.cognitive.microsofttranslator.com for Azure Global.
For Azure China, please use api.translator.azure.cn.
2.16.75 MT_MICROSOFT_COGNITIVE_KEY

Client key for the Microsoft Cognitive Services Translator API.

Ayrıca bakınız:

Microsoft Cognitive Services Translator, Makine çeviri, Otomatik öneriler, Cognitive Services - Text Translation API, Microsoft Azure Portal

2.16.76 MT_MICROSOFT_REGION

Region prefix as defined in the “Authenticating with a Multi-service resource” section.

2.16.77 MT_MICROSOFT_ENDPOINT_URL

Region endpoint URL domain for access token as defined in the “Authenticating with an access token” section. Defaults to api.cognitive.microsoft.com for Azure Global. For Azure China, please use your endpoint from the Azure Portal.

2.16.78 MT_MODERNMT_KEY

ModernMT makine çeviri motoru için API anahtarı.

Ayrıca bakınız:

ModernMT MT_MODERNMT_URL

2.16.79 MT_MODERNMT_URL

URL of ModernMT. It defaults to https://api.modernmt.com/ for the cloud service.

Ayrıca bakınız:

ModernMT MT_MODERNMT_KEY

2.16.80 MT_MYMEMORY_EMAIL

MyMemory identification e-mail address. It permits 1000 requests per day.

Ayrıca bakınız:

MyMemory, Makine çeviri, Otomatik öneriler, MyMemory: API technical specifications

2.16.81 MT_MYMEMORY_KEY

MyMemory access key for private translation memory, use it with MT_MYMEMORY_USER.

Ayrıca bakınız:

MyMemory, Makine çeviri, Otomatik öneriler, MyMemory: API key generator
2.16.82 MT_MYMEMORY_USER

MyMemory user ID for private translation memory, use it with MT_MYMEMORY_KEY.

 Ayrıca bakınız:
MyMemory, Makine çevirisı, Otomatik öneriler, MyMemory: API key generator

2.16.83 MT_NETEASE_KEY

App key for NetEase Sight API, you can register at https://sight.youdao.com/

 Ayrıca bakınız:
NetEase Sight API machine translation, Makine çevirisı, Otomatik öneriler

2.16.84 MT_NETEASE_SECRET

App secret for the NetEase Sight API, you can register at https://sight.youdao.com/

 Ayrıca bakınız:
NetEase Sight API machine translation, Makine çevirisı, Otomatik öneriler

2.16.85 MT_TMSERVER

URL where tmserver is running.

 Ayrıca bakınız:
tmserver, Makine çevirisı, Otomatik öneriler, tmserver

2.16.86 MT_YANDEX_KEY

API key for the Yandex Translate API, you can register at https://yandex.com/dev/translate/

 Ayrıca bakınız:
Yandex Translate, Makine çevirisı, Otomatik öneriler

2.16.87 MT_YOUDAO_ID


 Ayrıca bakınız:
Youdao Zhiyun API machine translation, Makine çevirisı, Otomatik öneriler

2.16.88 MT_YOUDAO_SECRET


 Ayrıca bakınız:
Youdao Zhiyun API machine translation, Makine çevirisı, Otomatik öneriler
2.16.89 MT_SAP_BASE_URL

API URL to the SAP Translation Hub service.

Ayrıca bakınız:
*SAP Translation Hub, Makine çevirisi, Otomatik öneriler*

2.16.90 MT_SAP_SANDBOX_APIKEY

API key for sandbox API usage

Ayrıca bakınız:
*SAP Translation Hub, Makine çevirisi, Otomatik öneriler*

2.16.91 MT_SAP_USERNAME

Your SAP username

Ayrıca bakınız:
*SAP Translation Hub, Makine çevirisi, Otomatik öneriler*

2.16.92 MT_SAP_PASSWORD

Your SAP password

Ayrıca bakınız:
*SAP Translation Hub, Makine çevirisi, Otomatik öneriler*

2.16.93 MT_SAP_USE_MT

Whether to also use machine translation services, in addition to the term database. Possible values: True or False

Ayrıca bakınız:
*SAP Translation Hub, Makine çevirisi, Otomatik öneriler*

2.16.94 NEARBY_MESSAGES

How many strings to show around the currently translated string. This is just a default value, users can adjust this in Kullanıcı profili.

2.16.95 DEFAULT_PAGE_LIMIT

4.7 sürümünde geldi.

Default number of elements to display when pagination is active.
2.16.96 PAGURE_CREDENTIALS

4.3.2 sürümünde geldi.
List for credentials for Pagure servers.

İpucu: Use this in case you want Weblate to interact with more of them, for single Pagure endpoint stick with PAGURE_USERNAME and PAGURE_TOKEN.

```json
PAGURE_CREDENTIALS = {
    "pagure.io": {
        "username": "weblate",
        "token": "your-api-token",
    },
    "pagure.example.com": {
        "username": "weblate",
        "token": "another-api-token",
    }
}
```

2.16.97 PAGURE_USERNAME

4.3.2 sürümünde geldi.
Çeviri güncellemeleri için birleştirme istekleri gönderirken kullanılarak Pagure kullanıcı adı.

Ayrıca bakınız: PAGURE_CREDENTIALS, Pagure

2.16.98 PAGURE_TOKEN

4.3.2 sürümünde geldi.
Çeviri güncellemeleri için API çağrıları yaparken kullanılarak Pagure kişisel erişim belirteci.

Ayrıca bakınız: PAGURE_CREDENTIALS, Pagure, Pagure API

2.16.99 PRIVACY_URL

4.8.1 sürümünde geldi.
URL where your Weblate instance shows its privacy policy.

İpucu: Useful if you host your legal documents outside Weblate for embedding them inside Weblate, please check Yasal for details.

Example:

```python
PRIVACY_URL = "https://weblate.org/terms/"
```

Ayrıca bakınız:

LEGAL_URL
2.16.100 RATELIMIT_ATTEMPTS

3.2 sürümünde geldi.
Maximum number of authentication attempts before rate limiting is applied.
Defaults to 5.

Ayrıca bakınız:
Rate limiting, RATELIMIT_WINDOW, RATELIMIT_LOCKOUT

2.16.101 RATELIMIT_WINDOW

3.2 sürümünde geldi.
How long authentication is accepted after rate limiting applies.
An amount of seconds defaulting to 300 (5 minutes).

Ayrıca bakınız:
Rate limiting, RATELIMIT_ATTEMPTS, RATELIMIT_LOCKOUT

2.16.102 RATELIMIT_LOCKOUT

3.2 sürümünde geldi.
How long authentication is locked after rate limiting applies.
An amount of seconds defaulting to 600 (10 minutes).

Ayrıca bakınız:
Rate limiting, RATELIMIT_ATTEMPTS, RATELIMIT_WINDOW

2.16.103 REGISTRATION_ALLOW_BACKENDS

4.1 sürümünde geldi.
List of authentication backends to allow registration from. This only limits new registrations, users can still authen-
ticate and add authentication using all configured authentication backends.

It is recommended to keep REGISTRATION_OPEN enabled while limiting registration backends, otherwise users
will be able to register, but Weblate will not show links to register in the user interface.

Example:

```
REGISTRATION_ALLOW_BACKENDS = ["azuread-oauth2", "azuread-tenant-oauth2"]
```

İpucu: The backend names match names used in URL for authentication.

Ayrıca bakınız:
REGISTRATION_OPEN, Kimlik doğrulama
2.16.104 REGISTRATION_CAPTCHA

A value of either `True` or `False` indicating whether registration of new accounts is protected by CAPTCHA. This setting is optional, and a default of `True` will be assumed if it is not supplied.

If turned on, a CAPTCHA is added to all pages where a users enters their e-mail address:

- New account registration.
- Password recovery.
- Adding e-mail to an account.
- Contact form for users that are not signed in.

2.16.105 REGISTRATION_EMAIL_MATCH

2.17 sürümünde geldi.

Allows you to filter which e-mail addresses can register.

Defaults to `.*`, which allows any e-mail address to be registered.

You can use it to restrict registration to a single e-mail domain:

```
REGISTRATION_EMAIL_MATCH = r'^.*@weblate\.org$'
```

2.16.106 REGISTRATION_OPEN

Whether registration of new accounts is currently permitted. This optional setting can remain the default `True`, or changed to `False`.

This setting affects built-in authentication by e-mail address or through the Python Social Auth (you can whitelist certain back-ends using `REGISTRATION_ALLOW_BACKENDS`).

**Not:** If using third-party authentication methods such as LDAP authentication, it just hides the registration form, but new users might still be able to sign in and create accounts.

Ayrıca bakınız:

`REGISTRATION_ALLOW_BACKENDS, REGISTRATION_EMAIL_MATCH, Kimlik doğrulama`

2.16.107 REPOSITORY_ALERT_THRESHOLD

4.0.2 sürümünde geldi.

Threshold for triggering an alert for outdated repositories, or ones that contain too many changes. Defaults to 25.

Ayrıca bakınız:

alerts
2.16.108 REQUIRE_LOGIN

This enables LOGIN_REQUIRED_URLS and configures REST framework to require authentication for all API endpoints.

**Not:** This is implemented in the Sample configuration. For Docker, use WEBLATE_REQUIRE_LOGIN.

2.16.109 SENTRY_DSN

Sentry DSN to use for Collecting error reports.

**Ayrıca bakınız:**

Django integration for Sentry

2.16.110 SESSION_COOKIE_AGE_AUTHENTICATED

Set session expiry for authenticated users. This complements SESSION_COOKIE_AGE which is used for unauthenticated users.

**Ayrıca bakınız:**

SESSION_COOKIE_AGE

2.16.111 SIMPLIFY_LANGUAGES

Use simple language codes for default language/country combinations. For example an fr_FR translation will use the fr language code. This is usually the desired behavior, as it simplifies listing languages for these default combinations.

Turn this off if you want to different translations for each variant.

2.16.112 SITE_DOMAIN

Configures site domain. This is necessary to produce correct absolute links in many scopes (for example activation e-mails, notifications or RSS feeds).

In case Weblate is running on non-standard port, include it here as well.

**Examples:**

```
# Production site with domain name
SITE_DOMAIN = "weblate.example.com"

# Local development with IP address and port
SITE_DOMAIN = "127.0.0.1:8000"
```

**Not:** This setting should only contain the domain name. For configuring protocol, (enabling and enforcing HTTPS) use ENABLE_HTTPS and for changing URL, use URL_PREFIX.
İpucu: On a Docker container, the site domain is configured through `WEBLATE_ALLOWED_HOSTS`.

Ayrıca bakınız:
Set correct site domain, Allowed hosts setup, Correctly configure HTTPS `WEBLATE_SITE_DOMAIN`, `ENABLE_HTTPS`.

### 2.16.113 SITE_TITLE
Site title to be used for the website and sent e-mails.

### 2.16.114 SPECIAL_CHARS
Additional characters to include in the visual keyboard, *Visual keyboard*.
The default value is:

```plaintext
SPECIAL_CHARS = ("\t", "\n", "\u00a0", "_")
```

### 2.16.115 SINGLE_PROJECT
3.8 sürümünde geldi.
Redirects users directly to a project or component instead of showing the dashboard. You can either set it to `True` and in this case it only works in case there is actually only single project in Weblate. Alternatively set the project slug, and it will redirect unconditionally to this project.
3.11 sürümünde değişti: The setting now also accepts a project slug, to force displaying that single project.
Example:

```plaintext
SINGLE_PROJECT = "test"
```

### 2.16.116 STATUS_URL
The URL where your Weblate instance reports its status.

### 2.16.117 SUGGESTION_CLEANUP_DAYS
3.2.1 sürümünde geldi.
Automatically deletes suggestions after a given number of days. Defaults to `None`, meaning no deletions.

### 2.16.118 UPDATE_LANGUAGES
4.3.2 sürümünde geldi.
Controls whether languages database should be updated when running database migration and is enabled by default. This setting has no effect on invocation of `setuplang`.

Ayrıca bakınız:
*Yerleşik dil tanımları*
### 2.16.119 URL_PREFIX

This setting allows you to run Weblate under some path (otherwise it relies on being run from the webserver root).

**Not:** To use this setting, you also need to configure your server to strip this prefix. For example with WSGI, this can be achieved by setting `WSGIScriptAlias`.

**İpucu:** The prefix should start with a `/`.

Example:

```
URL_PREFIX = "/translations"
```

**Not:** This setting does not work with Django's built-in server, you would have to adjust `urls.py` to contain this prefix.

### 2.16.120 VCS_BACKENDS

Configuration of available VCS backends.

**Not:** Weblate tries to use all supported back-ends you have the tools for.

**İpucu:** You can limit choices or add custom VCS back-ends by using this.

```
VCS_BACKENDS = ("weblate.vcs.git.GitRepository",)
```

**Ayrıca bakınız:**

*Sürüm denetimi tümleşimi*

### 2.16.121 VCS_CLONE_DEPTH

3.10.2 sürümünde geldi.

Configures how deep cloning of repositories Weblate should do.

**Not:** Currently this is only supported in `Git`. By default Weblate does shallow clones of the repositories to make cloning faster and save disk space. Depending on your usage (for example when using custom *Ekleniler*), you might want to increase the depth or turn off shallow clones completely by setting this to 0.

**İpucu:** In case you get `fatal: protocol error: expected old/new/ref, got 'shallow <commit hash>'` error when pushing from Weblate, turn off shallow clones completely by setting:

```
VCS_CLONE_DEPTH = 0
```
2.16.122 WEBLATE_ADDONS

List of addons available for use. To use them, they have to be enabled for a given translation component. By default this includes all built-in addons, when extending the list you will probably want to keep existing ones enabled, for example:

```python
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.LanguageConsistencyAddon",
    "weblate.addons.discovery.DiscoveryAddon",
    "weblate.addons.flags.SourceEditAddon",
    "weblate.addons.flags.TargetEditAddon",
    "weblate.addons.flags.SameEditAddon",
    "weblate.addons.flags.BulkEditAddon",
    "weblate.addons.generate.GenerateFileAddon",
    "weblate.addons.json.JSONCustomizeAddon",
    "weblate.addons.properties.PropertiesSortAddon",
    "weblate.addons.git.GitSquashAddon",
    "weblate.addons.removal.RemoveComments",
    "weblate.addons.removal.RemoveSuggestions",
    "weblate.addons.resx.ResxUpdateAddon",
    "weblate.addons.autotranslate.AutoTranslateAddon",
    "weblate.addons.yaml.YAMLCustomizeAddon",
    "weblate.addons.cdn.CDNJSAddon",
    # Add-on you want to include
    "weblate.addons.example.ExampleAddon",
}
```

**Not:** Removing the addon from the list does not uninstall it from the components. Weblate will crash in that case. Please uninstall addon from all components prior to removing it from this list.

Ayrıca bakınız:

Eklentiler, DEFAULT_ADDONS

2.16.123 WEBLATE_EXPORTERS

4.2 sürümünde geldi.

List of available exporters offering downloading translations or glossaries in various file formats.

Ayrıca bakınız:

Supported file formats
2.16.124 WEBLATE_FORMATS

3.0 sürümünde geldi.

List of file formats available for use.

**Not:** The default list already has the common formats.

**Ayrıca bakınız:**

Supported file formats

2.16.125 WEBLATE_GPG_IDENTITY

3.1 sürümünde geldi.

Identity used by Weblate to sign Git commits, for example:

```
WEBLATE_GPG_IDENTITY = "Weblate <weblate@example.com>"
```

The Weblate GPG keyring is searched for a matching key (home/.gnupg under DATA_DIR). If not found, a key is generated, please check Signing Git commits with GnuPG for more details.

**Ayrıca bakınız:**

Signing Git commits with GnuPG

2.16.126 WEBSITE_REQUIRED

Defines whether Proje web sitesi has to be specified when creating a project. Turned on by default as that suits public server setups.

2.17 Sample configuration

The following example is shipped as weblate/settings_example.py with Weblate:

```python
import os
```

(sonraki sayfaya devam)
import platform
from logging.handlers import SysLogHandler

# Title of site to use
SITE_TITLE = "Weblate"

# Site domain
SITE_DOMAIN = ""

# Whether site uses https
ENABLE_HTTPS = False

# Django settings for Weblate project.

DEBUG = True
ADMINS = {
    # ("Your Name", "your_email@example.com"),
}
MANAGERS = ADMINS

DATABASES = {
    "default": {
        # Use "postgresql" or "mysql".
        "ENGINE": "django.db.backends.postgresql",
        # Database name.
        "NAME": "weblate",
        # Database user.
        "USER": "weblate",
        # Name of role to alter to set parameters in PostgreSQL,
        # use in case role name is different than user used for authentication.
        "ALTER_ROLE": "weblate",
        # Database password.
        "PASSWORD": "",
        # Set to empty string for localhost.
        "HOST": "127.0.0.1",
        # Set to empty string for default.
        "PORT": "",
        # Customizations for databases.
        "OPTIONS": {
            # In case of using an older MySQL server,
            # which has MyISAM as a default storage
            "init_command": "SET storage_engine =INNODB",
            # Uncomment for MySQL older than 5.7:
            # "init_command": "SET sql_mode ='STRICT_TRANS_TABLES'",
            # Set emoji capable charset for MySQL:
            "charset": "utf8mb4",
            # Change connection timeout in case you get MySQL gone away error:
            # "connect_timeout": 28800,
        },
        # Persistent connections
        "CONN_MAX_AGE": 0,
    }
}

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Data directory
DATA_DIR = os.path.join(BASE_DIR, "data")

# Local time zone for this installation. Choices can be found here:
# http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
# although not all choices may be available on all operating systems.
# In a Windows environment this must be set to your system time zone.
TIME_ZONE = "UTC"

# Language code for this installation. All choices can be found here:
# http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = "en-us"

LANGUAGES = (
    ("ar", "العربية"),
    ("az", "Azerbaijani"),
    ("be", "Belarusian"),
    ("bi", "Bislama"),
    ("bg", "Bulgarian"),
    ("br", "Breton"),
    ("ca", "Catalan"),
    ("cs", "Czech"),
    ("da", "Danish"),
    ("de", "German"),
    ("en", "English"),
    ("el", "Greek"),
    ("en-gb", "English (United Kingdom)"),
    ("es", "Spanish"),
    ("fi", "Finnish"),
    ("fr", "French"),
    ("gl", "Galician"),
    ("he", "Hebrew"),
    ("hu", "Hungarian"),
    ("hr", "Croatian"),
    ("id", "Indonesian"),
    ("is", "Icelandic"),
    ("it", "Italian"),
    ("ja", "Japanese"),
    ("ka", "Georgian"),
    ("kk", "Kazakh"),
    ("ko", "Korean"),
    ("nb", "Norwegian"),
    ("nl", "Dutch"),
    ("pl", "Polish"),
    ("pt", "Portuguese"),
    ("pt-br", "Brazilian Portuguese"),
    ("ro", "Romanian"),
    ("ru", "Russian"),
    ("sk", "Slovak"),
    ("sl", "Slovenian"),
    ("sq", "Albanian"),
    ("sr", "Serbian"),
    ("sr-latn", "Serbian"),
    ("sv", "Swedish"),
    ("th", "Thai"),
    ("tr", "Turkish"),
    ("uk", "Ukrainian"),
    ("zh-hans", "Chinese (Simplified)"),
    ("zh-hant", "Chinese (Traditional)"),
)

SITE_ID = 1
# If you set this to False, Django will make some optimizations so as not to load the internationalization machinery.
USE_I18N = True

# If you set this to False, Django will not format dates, numbers and calendars according to the current locale.
USE_L10N = True

# If you set this to False, Django will not use timezone-aware datetimes.
USE_TZ = True

# Type of automatic primary key, introduced in Django 3.2
DEFAULT_AUTO_FIELD = "django.db.models.AutoField"

# URL prefix to use, please see documentation for more details
URL_PREFIX = ""

# Absolute filesystem path to the directory that will hold user-uploaded files.
MEDIA_ROOT = os.path.join(DATA_DIR, "media")

# URL that handles the media served from MEDIA_ROOT. Make sure to use a trailing slash.
MEDIA_URL = f"{URL_PREFIX}/media/

# Absolute path to the directory static files should be collected to.
# Don't put anything in this directory yourself; store your static files in apps' "static/" subdirectories and in STATICFILES_DIRS.
STATIC_ROOT = os.path.join(DATA_DIR, "static")

# URL prefix for static files.
STATIC_URL = f"{URL_PREFIX}/static/

# Additional locations of static files
STATICFILES_DIRS = (  
    # Put strings here, like "/home/html/static" or "C:/www/django/static".  
    # Always use forward slashes, even on Windows.  
    # 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.request",
        "django.template.context_processors.csrf",
        "django.contrib.messages.context_processors.messages",
        "weblate.trans.context_processors.weblate_context",
    ],
    "loaders": _TEMPLATE_LOADERS,
},

# GitHub username and token for sending pull requests.
# Please see the documentation for more details.
GITHUB_USERNAME = None
GITHUB_TOKEN = None

# GitLab username and token for sending merge requests.
# Please see the documentation for more details.
GITLAB_USERNAME = None
GITLAB_TOKEN = None

# Authentication configuration
AUTHENTICATION_BACKENDS = {
    "social_core.backends.email.EmailAuth",
    "social_core.backends.google.GoogleOAuth2",
    "social_core.backends.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_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.remove_account",
    "social_core.pipeline.social_auth.associate_by_email",
    "weblate.accounts.pipeline.reauthenticate",
    "weblate.accounts.pipeline.verify_username",
    "social_core.pipeline.user.create_user",
    "social_core.pipeline.social_auth.associate_user",
    "social_core.pipeline.social_auth.load_extra_data",
    "weblate.accounts.pipeline.cleanup_next",
    "weblate.accounts.pipeline.user_full_name",
    "weblate.accounts.pipeline.store_email",
    "weblate.accounts.pipeline.notify_connect",
    "weblate.accounts.pipeline.password_reset",
)
SOCIAL_AUTH_DISCONNECT_PIPELINE = (
    "social_core.pipeline.disconnect.allowed_to_disconnect",
    "social_core.pipeline.disconnect.get_entries",
    "social_core.pipeline.disconnect.revoke_tokens",
    "weblate.accounts.pipeline.cycle_session",
    "weblate.accounts.pipeline.adjust_primary_mail",
    "weblate.accounts.pipeline.notify_disconnect",
    "social_core.pipeline.disconnect.disconnect",
    "weblate.accounts.pipeline.cleanup_next",
)

# Custom authentication strategy
SOCIAL_AUTH_STRATEGY = "weblate.accounts.strategy.WeblateStrategy"
# Raise exceptions so that we can handle them later
SOCIAL_AUTH_RAISE_EXCEPTIONS = True
SOCIAL_AUTH_EMAIL_VALIDATION_FUNCTION = "weblate.accounts.pipeline.send_validation"
SOCIAL_AUTH_EMAIL_VALIDATION_URL = f"{URL_PREFIX}/accounts/email-sent/
SOCIAL_AUTH_LOGIN_ERROR_URL = f"{URL_PREFIX}/accounts/login/
SOCIAL_AUTH_EMAIL_FORM_URL = f"{URL_PREFIX}/accounts/email/
SOCIAL_AUTH_NEW_ASSOCIATION_REDIRECT_URL = f"{URL_PREFIX}/accounts/profile/#account_

SOCIAL_AUTH_PROTECTED_USER_FIELDS = ("email",)
SOCIAL_AUTH_SLUGIFY_USERNAMES = True
SOCIAL_AUTH_SLUGIFY_FUNCTION = "weblate.accounts.pipeline.slugify_username"

# Password validation configuration
AUTH_PASSWORD_VALIDATORS = [
    {"NAME": "django.contrib.auth.password_validation.
UserAttributeSimilarityValidator" # noqa: E501, pylint: disable =line-too-long
    },
    {"NAME": "django.contrib.auth.password_validation.
MinimumLengthValidator", "OPTIONS": {"min_length": 10},
}
Password hashing (prefer Argon)
```
PASSWORD_HASHERS = [
    "django.contrib.auth.hashers.Argon2PasswordHasher",
    "django.contrib.auth.hashers.PBKDF2PasswordHasher",
    "django.contrib.auth.hashers.PBKDF2SHA1PasswordHasher",
    "django.contrib.auth.hashers.BCryptSHA256PasswordHasher",
]
```

# Allow new user registrations
REGISTRATION_OPEN = True

# Shortcut for login required setting
REQUIRE_LOGIN = False

# Middleware
MIDDLEWARE = [
    "weblate.middleware.RedirectMiddleware",
    "weblate.middleware.ProxyMiddleware",
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "weblate.accounts.middleware.AuthenticationMiddleware",
    "weblate.middleware.SessionMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
    "social_django.middleware.SocialAuthExceptionMiddleware",
    "weblate.accounts.middleware.RequireLoginMiddleware",
    "weblate.api.middleware.ThrottlingMiddleware",
    "weblate.middleware.SecurityMiddleware",
    "weblate.wladmin.middleware.ManageMiddleware",
]

ROOT_URLCONF = "weblate.urls"

# Django and Weblate apps
INSTALLED_APPS = [
    # Weblate apps on top to override Django locales and templates
    "weblate.addons",
    "weblate.auth",
    "weblate.checks",
    "weblate.forms",
    "weblate.glossary",
    "weblate.machinery",
    "weblate.trans",
    "weblate.lang",
    "weblate_language_data",
    "weblate.memory",
]

# Optional password strength validation by django-zxcvbn-password
```
# {
#     "NAME": "zxcvbn_password.ZXCVBNValidator",
#     "OPTIONS": {
#         "min_score": 3,
#         "user_attributes": ["username", "email", "full_name"]
#     }
# }
```

Password hashing (prefer Argon)
"weblate.screenshots",
"weblate.fonts",
"weblate.accounts",
"weblate.configuration",
"weblate.utils",
"weblate.vcs",
"weblate.wladmin",
"weblate.metrics",
"weblate",
# Optional: Git exporter
"weblate.gitexport",
# Standard Django modules
"django.contrib.auth",
"django.contrib.contenttypes",
"django.contrib.sessions",
"django.contrib.messages",
"django.contrib.staticfiles",
"django.contrib.admin.apps.SimpleAdminConfig",
"django.contrib.admindocs",
"django.contrib.sitemaps",
"django.contrib.humanize",
# Third party Django modules
"social_django",
"crispy_forms",
"compressor",
"rest_framework",
"rest_framework.authtoken",
"django_filters",

# Custom exception reporter to include some details
DEFAULT_EXCEPTION_REPORTER_FILTER = "weblate.trans.debug.
!WeblateExceptionReporterFilter"

# Default logging of Weblate messages
# - to syslog in production (if available)
# - otherwise to console
# - you can also choose "logfile" to log into separate file
# after configuring it below

# Detect if we can connect to syslog
HAVE_SYSLOG = False
if platform.system() != "Windows":
    try:
        handler = SysLogHandler(address="/dev/log", facility=sysLogHandler.LOG_~LOCAL2)
        handler.close()
        HAVE_SYSLOG = True
    except OSError:
        HAVE_SYSLOG = False

if DEBUG or not HAVE_SYSLOG:
    DEFAULT_LOG = "console"
else:
    DEFAULT_LOG = "syslog"
DEFAULT_LOGLEVEL = "DEBUG" if DEBUG else "INFO"

# A sample logging configuration. The only tangible logging
# performed by this configuration is to send an email to
# the site admins on every HTTP 500 error when DEBUG =False.
# See http://docs.djangoproject.com/en/stable/topics/logging for

(sonraki sayfaya devam)
# more details on how to customize your logging configuration.

```json
LOGGING = {
    "version": 1,
    "disable_existing_loggers": True,
    "filters": {
        "require_debug_false": {
            "()": "django.utils.log.RequireDebugFalse"
        }
    },
    "formatters": {
        "syslog": {
            "format": "weblate[%(process)d]: %(levelname)s %(message)s",
            "simple": {
                "format": "%(asctime)s %(levelname)s %(message)s"
            }
        },
        "logfile": {
            "format": "%(asctime)s %(levelname)s %(message)s"
        },
        "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",
        },
    },
    "handlers": {
        "mail_admins": {
            "level": "ERROR",
            "filters": ["require_debug_false"],
            "class": "django.utils.log.AdminEmailHandler",
            "include_html": True,
        },
        "console": {
            "level": "DEBUG",
            "class": "logging.StreamHandler",
            "formatter": "simple",
        },
        "django.server": {
            "level": "INFO",
            "class": "logging.StreamHandler",
            "formatter": "django.server",
        },
    },
    "loggers": {
        "django.request": {
            "handlers": ["mail_admins", DEFAULT_LOG],
            "level": "ERROR",
            "propagate": True,
        },
        "django.server": {
            "handlers": ["django.server"],
            "level": "INFO",
            "propagate": False,
        },
    }
}
```

(sonraki sayfaya devam)
# Logging database queries
"django.db.backends": {
    "handlers": [DEFAULT_LOG],
    "level": "DEBUG",
},
"weblate": {
    "handlers": [DEFAULT_LOG],
    "level": DEFAULT_LOGLEVEL,
},

# Logging VCS operations
"weblate.vcs": {
    "handlers": [DEFAULT_LOG],
    "level": DEFAULT_LOGLEVEL,
},

# Python Social Auth
"social": {
    "handlers": [DEFAULT_LOG],
    "level": DEFAULT_LOGLEVEL,
},

# Django Authentication Using LDAP
"django_auth_ldap": {
    "handlers": [DEFAULT_LOG],
    "level": DEFAULT_LOGLEVEL,
},

# SAML IdP
"djangosaml2idp": {
    "handlers": [DEFAULT_LOG],
    "level": DEFAULT_LOGLEVEL,
},

# Remove syslog setup if it's not present
if not HAVE_SYSLOG:
    del LOGGING["handlers"]['syslog']

# List of machine translations
MT_SERVICES = {
    "weblate.machinery.apertium.ApertiumAPYTranslation",
    "weblate.machinery.baidu.BaiduTranslation",
    "weblate.machinery.deepl.DeepLTranslation",
    "weblate.machinery.glosbe.GlosbeTranslation",
    "weblate.machinery.google.GoogleTranslation",
    "weblate.machinery.googlev3.GoogleV3Translation",
    "weblate.machinery.libretranslate.LibreTranslateTranslation",
    "weblate.machinery.microsoft.MicrosoftCognitiveTranslation",
    "weblate.machinery.microsoftterminology.MicrosoftTerminologyService",
    "weblate.machinery.modernmt.ModernMTTranslation",
    "weblate.machinery.mymemory.MyMemoryTranslation",
    "weblate.machinery.netease.NeteaseSightTranslation",
    "weblate.machinery.tmserver.AmagamaTranslation",
    "weblate.machinery.tmservserver.TMServerTranslation",
    "weblate.machinery.yandex.YandexTranslation",
    "weblate.machinery.saptranslationalhub.SAPTranslationHub",
    "weblate.machinery.youdao.YoudaoTranslation",
    "weblate.machinery.weblatetm.WeblateTranslation",
    "weblate.memory.machine.WeblateMemory",
}

# Machine translation API keys

# URL of the Apertium APy server
MT_APERTIUM_APY = None

# DeepL API key
MT_DEEPL_KEY = None

# LibreTranslate
MT_LIBRETRANSLATE_API_URL = None
MT_LIBRETRANSLATE_KEY = None

# Microsoft Cognitive Services Translator API, register at
# https://portal.azure.com/
MT_MICROSOFT_COGNITIVE_KEY = None
MT_MICROSOFT_REGION = None
# ModernMT
MT_MODERNMT_KEY = None

# MyMemory identification email, see
# https://mymemory.translated.net/doc/spec.php
MT_MYMEMORY_EMAIL = None

# Optional MyMemory credentials to access private translation memory
MT_MYMEMORY_USER = None
MT_MYMEMORY_KEY = None

# Google API key for Google Translate API v2
MT_GOOGLE_KEY = None

# Google Translate API3 credentials and project id
MT_GOOGLE_CREDENTIALS = None
MT_GOOGLE_PROJECT = None

# Baidu app key and secret
MT_BAIDU_ID = None
MT_BAIDU_SECRET = None

# Youdao Zhiyun app key and secret
MT_YOUDAO_ID = None
MT_YOUDAO_SECRET = None

# NetEase Sight (Jianwai) app key and secret
MT_NETEASE_KEY = None
MT_NETEASE_SECRET = None

# API key for Yandex Translate API
MT_YANDEX_KEY = None

# tmserver URL
MT_TMSERVER = None

# SAP Translation Hub
MT_SAP_BASE_URL = None
MT_SAP_SANDBOX_APIKEY = None
MT_SAP_USERNAME = None
MT_SAP_PASSWORD = None
MT_SAP_USE_MT = True

# Use HTTPS when creating redirect URLs for social authentication, see
# documentation for more details:
# --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"

# SSL redirect
SECURE_SSL_REDIRECT = True

SESSION_COOKIE_SECURE = ENABLE_HTTPS
SESSION_COOKIE_HTTPONLY = True

# SSL redirect
SECURE_SSL_REDIRECT = ENABLE_HTTPS
# Sent referer only for same origin links
SECURE_REFERRER_POLICY = "same-origin"

# SSL redirect URL exemption list
SECURE_REDIRECT_EXEMPT = (r"healthz/$",) # Allowing HTTP access to health check

# Session cookie age (in seconds)
SESSION_COOKIE_AGE = 1000
SESSION_COOKIE_AGE_AUTHENTICATED = 1209600
SESSION_COOKIE_SAMESITE = "Lax"

# Increase allowed upload size
DATA_UPLOAD_MAX_MEMORY_SIZE = 50000000

# Apply session cookie settings to language cookie as well
LANGUAGE_COOKIE_SECURE = SESSION_COOKIE_SECURE
LANGUAGE_COOKIE_HTTPONLY = SESSION_COOKIE_HTTPONLY
LANGUAGE_COOKIE_AGE = SESSION_COOKIE_AGE_AUTHENTICATED * 10
LANGUAGE_COOKIE_SAMESITE = SESSION_COOKIE_SAMESITE

# Some security headers
SECURE_BROWSER_XSS_FILTER = True
X_FRAME_OPTIONS = "DENY"
SECURE_CONTENT_TYPE_NOSNIFF = True

# Optionally enable HSTS
SECURE_HSTS_SECONDS = 31536000 if ENABLE_HTTPS else 0
SECURE_HSTS_PRELOAD = ENABLE_HTTPS
SECURE_HSTS_INCLUDE_SUBDOMAINS = ENABLE_HTTPS

# HTTPS detection behind reverse proxy
SECURE_PROXY_SSL_HEADER = None

# URL of login
LOGIN_URL = f"{URL_PREFIX}/accounts/login/"

# URL of logout
LOGOUT_URL = f"{URL_PREFIX}/accounts/logout/"

# Default location for login
LOGIN_REDIRECT_URL = f"{URL_PREFIX}/"

# Anonymous user name
ANONYMOUS_USER_NAME = "anonymous"

# Reverse proxy settings
IP_PROXY_HEADER = "HTTP_X_FORWARDED_FOR"
IP_BEHIND_REVERSE_PROXY = False
IP_PROXY_OFFSET = 0

# Sending HTML in mails
EMAIL_SEND_HTML = True

# Subject of emails includes site title
EMAIL_SUBJECT_PREFIX = f"[{SITE_TITLE}]"

# Enable remote hooks
ENABLE_HOOKS = True

# By default the length of a given translation is limited to the length of
# the source string * 10 characters. Set this option to False to allow longer
# translations (up to 10,000 characters)
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH = True
# Use simple language codes for default language/country combinations
SIMPLIFY_LANGUAGES = True

# Render forms using bootstrap
CRISPY_TEMPLATE_PACK = "bootstrap3"

# List of quality checks
# CHECK_LIST = (  #
#    "weblate.checks.same.SameCheck",
#    "weblate.checks.chars.BeginNewlineCheck",
#    "weblate.checks.chars.EndNewlineCheck",
#    "weblate.checks.chars.BeginSpaceCheck",
#    "weblate.checks.chars.EndSpaceCheck",
#    "weblate.checks.chars.DoubleSpaceCheck",
#    "weblate.checks.chars.EndStopCheck",
#    "weblate.checks.chars.EndColonCheck",
#    "weblate.checks.chars.EndQuestionCheck",
#    "weblate.checks.chars.EndExclamationCheck",
#    "weblate.checks.chars.EndEllipsisCheck",
#    "weblate.checks.chars.EndSemicolonCheck",
#    "weblate.checks.chars.MaxLengthCheck",
#    "weblate.checks.chars.KashidaCheck",
#    "weblate.checks.chars.PunctuationSpacingCheck",
#    "weblate.checks.format.PythonFormatCheck",
#    "weblate.checks.format.PythonBraceFormatCheck",
#    "weblate.checks.format.PHPFormatCheck",
#    "weblate.checks.format.CFormatCheck",
#    "weblate.checks.format.PerlFormatCheck",
#    "weblate.checks.format.JavaScriptFormatCheck",
#    "weblate.checks.format.ObjectPascalFormatCheck",
#    "weblate.checks.format.SchemeFormatCheck",
#    "weblate.checks.format.CSharpFormatCheck",
#    "weblate.checks.format.JavaFormatCheck",
#    "weblate.checks.format.JavaMessageFormatCheck",
#    "weblate.checks.format.PercentPlaceholdersCheck",
#    "weblate.checks.format.VueFormattingCheck",
#    "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.markup.ZeroWidthSpaceCheck",
#    "weblate.checks.markup.MaxLengthCheck",
#    "weblate.checks.markup.XMLValidityCheck",
#    "weblate.checks.markup.XMLTagsCheck",
#    "weblate.checks.markup.MarkdownRefLinkCheck",
#    "weblate.checks.markup.MarkdownSyntaxCheck",
#    "weblate.checks.markup.URLCheck",
#    "weblate.checks.markup.SafeHTMLCheck",
#    "weblate.checks.placeholders.PlaceholderCheck",
#    "weblate.checks.placeholders.RegexCheck",  #
# List of automatic fixups
# AUTOFIX_LIST = (  
#     "weblate.trans.autofixes.whitespace.SameBookendingWhitespace",  
#     "weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis",  
#     "weblate.trans.autofixes.chars.RemoveZeroSpace",  
#     "weblate.trans.autofixes.chars.RemoveControlChars",  
# )

# List of enabled addons
# WEBLATE_ADDONS = (  
#     "weblate.addons.gettext.GenerateMoAddon",  
#     "weblate.addons.gettext.UpdateLinguasAddon",  
#     "weblate.addons.gettext.UpdateConfigureAddon",  
#     "weblate.addons.gettext.MsgmergeAddon",  
#     "weblate.addons.gettext.GettextCustomizeAddon",  
#     "weblate.addons.gettext.GettextAuthorComments",  
#     "weblate.addons.cleanup.CleanupAddon",  
#     "weblate.addons.cleanup.RemoveBlankAddon",  
#     "weblate.addons.consistency.LanguageConsistencyAddon",  
#     "weblate.addons.discovery.DiscoveryAddon",  
#     "weblate.addons.autotranslate.AutoTranslateAddon",  
#     "weblate.addons.flags.SourceEditAddon",  
#     "weblate.addons.flags.TargetEditAddon",  
#     "weblate.addons.flags.SameEditAddon",  
#     "weblate.addons.flags.BulkEditAddon",  
#     "weblate.addons.generate.GenerateFileAddon",  
#     "weblate.addons.generate.PseudolocaleAddon",  
#     "weblate.addons.json.JSONCustomizeAddon",  
#     "weblate.addons.properties.PropertiesSortAddon",  
#     "weblate.addons.git.GitSquashAddon",  
#     "weblate.addons.removal.RemoveComments",  
#     "weblate.addons.removal.RemoveSuggestions",  
#     "weblate.addons.resx.ResxUpdateAddon",  
#     "weblate.addons.yaml.YAMLCustomizeAddon",  
#     "weblate.addons.cdn.CDNJSAddon",  
# )

# E-mail address that error messages come from.
SERVER_EMAIL = "noreply@example.com"

# Default email address to use for various automated correspondence from
# the site managers. Used for registration emails.
DEFAULT_FROM_EMAIL = "noreply@example.com"

# List of URLs your site is supposed to serve
ALLOWED_HOSTS = ["*"]

# Configuration for caching
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/1",
    }
}
# If redis is running on same host as Weblate, you might want to use unix sockets instead:

```python
"LOCATION": "unix:///var/run/redis/redis.sock?db =1",
"OPTIONS": {
    "CLIENT_CLASS": "django_redis.client.DefaultClient",
    "PARSER_CLASS": "redis.connection.HiredisParser",
    "KEY_PREFIX": "weblate",
    "avator": {
        "BACKEND": "django.core.cache.backends.filebased.FileBasedCache",
        "LOCATION": os.path.join(DATA_DIR, "avatar-cache"),
        "TIMEOUT": 86400,
        "OPTIONS": {"MAX_ENTRIES": 1000},
    },
    "USER": "weblate",
    "PASSWORD": None,
    "CONNECTION_POOL_KWARGS": {},
}
```

# Store sessions in cache

```python
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
```

# Store messages in session

```python
MESSAGE_STORAGE = "django.contrib.messages.storage.session.SessionStorage"
```

# REST framework settings for API

```python
REST_FRAMEWORK = {
    "DEFAULT_PERMISSION_CLASSES": [
        rest_framework.permissions.IsAuthenticated
        if REQUIRE_LOGIN
        else rest_framework.permissions.IsAuthenticatedOrReadOnly
    ],
    "DEFAULT_AUTHENTICATION_CLASSES": (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

```python
FONTS_CDN_URL = None
```

# Django compressor offline mode

```python
COMPRESS_OFFLINE = False
COMPRESS_OFFLINE_CONTEXT = [
    {"fonts_cdn_url": FONTS_CDN_URL, "STATIC_URL": STATIC_URL, "LANGUAGE_BIDI": ...
    }]
```

(sonraki sayfaya devam)
# Require login for all URLs

```python
if REQUIRE_LOGIN:
    LOGIN_REQUIRED_URLS = (r"/(.*)$",)
```

# In such case you will want to include some of the exceptions

```python
# 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

```python
SILENCED_SYSTEM_CHECKS = [
    # We have modified django.contrib.auth.middleware.AuthenticationMiddleware
    # as weblate.accounts.middleware.AuthenticationMiddleware
    "admin.E408"
]
```

# Celery worker configuration for testing

```python
# 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
```

# Celery settings, it is not recommended to change these

```python
CELERY_WORKER_MAX_MEMORY_PER_CHILD = 200000
CELERY_BEAT_SCHEDULE_FILENAME = os.path.join(DATA_DIR, "celery", "beat-schedule")
CELERY_TASK_ROUTES = {
    "weblate.trans.tasks.auto_translate": {"queue": "translate"},
    "weblate.accounts.tasks.notify_*": {"queue": "notify"},
    "weblate.accounts.tasks.send_mails": {"queue": "notify"},
    "weblate.utils.tasks.settings_backup": {"queue": "backup"},
    "weblate.utils.tasks.database_backup": {"queue": "backup"},
    "weblate.wladmin.tasks.backup": {"queue": "backup"},
    "weblate.wladmin.tasks.backup_service": {"queue": "backup"},
    "weblate.memory.tasks.**": {"queue": "memory"},
}
```

# Enable plain database backups

```python
DATABASE_BACKUP = "plain"
```

# Enable auto updating

```python
AUTO_UPDATE = False
```

# PGP commits signing

```python
WEBLATE_GPG_IDENTITY = None
```

# Third party services integration

```python
(sonraki sayfaya devam)
```
2.18 Management commands

Not: 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 /tmp/users.json
```

Ayrıca bakınız:

Installing using Docker, Installing on Debian and Ubuntu, Installing on SUSE and openSUSE, Installing on RedHat, Fedora and CentOS, Installing from sources
2.18.2 add_suggestions

`weblate add_suggestions <project> <component> <language> <file>`

2.5 sürümünde geldi.

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).

Example:

```
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 sürümünde geldi.

4.6 sürümünde değişti: Çeviri modu için parametre eklendi.

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 `Tutarsız`).

`--add`

Automatically add language if a given translation does not exist.

`--mt MT`

Use machine translation instead of other components as machine translations.

`--threshold THRESHOLD`

Similarity threshold for machine translation, defaults to 80.

`--mode MODE`

Specify translation mode, default is translate but fuzzy or suggest can be used.

Example:

```
weblate auto_translate --user nijel --inconsistent --source weblate/application... --weblate website cs
```

Ayrıca bakınız:

`Kendiliğinden çeviri`
2.18.4 celery_queues

`weblate celery_queues`

3.7 sürümünde geldi.
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.

**Not:** 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.

Ayrıca bakınız:

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.

Ayrıca bakınız:

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).

2.9 sürümünde değişti: Added parameters `--username, --email, --name and --update`.

2.18.10 dump_memory

`weblate dump_memory`

2.20 sürümünde geldi.

Export a JSON file containing Weblate Translation Memory content.

Ayrıca bakınız:
Çeviri Belleği, Weblate Çeviri Belleği Şeması

2.18.11 dumpuserdata

`weblate dumpuserdata <file.json>`

Dumps userdata to a file for later use by `importuserdata`

İpucu: This comes in handy when migrating or merging Weblate instances.

2.18.12 import_demo

`weblate import_demo`

4.1 sürümünde geldi.

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 sürümünde geldi.

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.

2.9 sürümünde değişti: The parameters --ignore and --update are there to deal with already imported components.

Example of JSON file:

```json
[
  {
    "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"
  }
]
```

Ayrıca bakınız:
import_memory

2.18.14 import_memory

weblate import_memory <file>

2.20 sürümünde geldi.

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.
Ayrıca bakınız:
Çeviri Belleği, Weblate Çeviri Belleği Şeması

2.18.15 import_project

```
weblate import_project <project> <gitrepo> <branch> <filemask>
```

3.0 sürümünde değişti: The import_project command is now based on the Bileşen keşfi add-on, leading to some changes in behavior and what parameters are accepted.

Batch imports components into project based on filemask.

```
<project> names an existing project, into which the components are to be imported.
```

The `<gitrepo>` defines the Git repository URL to use, and `<branch>` signifies the Git branch. To import additional translation components from an existing Weblate component, use a `weblate://<project>/<component>` URL for the `<gitrepo>`.

The `<filemask>` defines file discovery for the repository. It can be either be made simple using wildcards, or it can use the full power of regular expressions.

The simple matching uses ** for component name and * for language, for example: **/*.po

The regular expression has to contain groups named component and language. For example: (?P<language>[^/\]*)/(?P<component>[^\-/\]*).po

The import matches existing components based on files and adds the ones that do not exist. It does not change already existing ones.

```
--name-template TEMPLATE
Customize the name of a component using Django template syntax.
For example: Documentation: {{ component }}
```

```
--base-file-template TEMPLATE
Customize the base file for monolingual translations.
For example: {{ component }}/res/values/string.xml
```

```
--new-base-template TEMPLATE
Customize the base file for addition of new translations.
For example: {{ component }}/ts/en.ts
```

```
--file-format FORMAT
You can also specify the file format to use (see Supported file formats), the default is auto-detection.
```

```
--language-regex REGEX
You can specify language filtering (see Component configuration) with this parameter. It has to be a valid regular expression.
```

```
--main-component
You can specify which component will be chosen as the main one—the one actually containing the VCS repository.
```

```
--license NAME
Specify the overall, project or component translation license.
```

```
--license-url URL
Specify the URL where the translation license is to be found.
```

```
--vcs NAME
In case you need to specify which version control system to use, you can do it here. The default version control is Git.
```

2.18. Management commands
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:

```bash
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:

```bash
weblate import_project \\ 
--file-format-properties \\ 
--base-file-template=web-app/tgol-web-app/src/main/resources/i18n/%s-I18N.properties \\ 
--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:

```bash
weblate import_project \\ 
tails \\
git://git.tails.boum.org/tails master \\
'wiki/src/security/(?P<component>.*)\.(?P<language>\[^.]*)(?P<language>[^\.*])\.(?P<language>[^\.*])\.po$'
```

Filtering only translations in a chosen language:

```bash
./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:

```bash
$ 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:

```bash
$ 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'
```

Ayrıca bakınız:

More detailed examples can be found in the starting chapter, alternatively you might want to use `import_json`.

352 Bölüm 2. Administrator docs
### 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 sürümünde geldi.

```
weblate install_addon --addon ADDON <project|project/component>
```

Installs an add-on to a set of components.

---addon ADDON

Name of the add-on to install. For example `weblate.gettext.customize`.

---configuration CONFIG

Bir ekleninın JSON biçimindeki yapılandırması.

---update

Mevcut eklenin yapılandırmasını güncelleye.

You can either define which project or component to install the add-on in (for example `weblate/application`), or use `--all` to include all existing components.

To install `Gettext çıktısi özelleştir` for all components:

```
weblate install_addon --addon weblate.gettext.customize --config '{"width": -1}' --update --all
```

Ayrıca bakınız:

`Eklentiler`

### 2.18.19 list_languages

```
weblate list_languages <locale>
```

Lists supported languages in MediaWiki markup - language codes, English names and localized names.

This is used to generate `<https://wiki.l10n.cz/Slovn%C3%ADk_s_n%C5%A1zvy_jazyk%C5%A1%AF>`.

---

2.18. Management commands 353
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.

Not: 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.

İpucu: 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.

Ayrıca bakınız:

```
unlock_translation
```
2.18.24 move_language

**weblate move_language source target**

3.0 sürümünde geldi.

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.

Example:

```bash
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.

**Not:** Weblate pushes changes automatically if İşleneye yolla 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.

**İpucu:** 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.

**Ayrıca bakınız:**

**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 *Proje erişim kontrolü*.

**Ayrıca bakınız:**

Yetkiler listesi

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.

**İpucu:** 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.

**Not:** Usually it is better to configure hooks in the repository to trigger *Bildirim kancaları*, instead of regular polling by `updategit`. 
2.19 Duyurular

4.0 sürümünde değişti: In prior releases this feature was called whiteboard messages.

Provide info to your translators by posting announcements, site-wide, per project, component, or language.

Announce the purpose, deadlines, status, or specify targets for translation.

The users will receive notification on the announcements for watched projects (unless they opt out).

This can be useful for various things from announcing the purpose of the website to specifying targets for translations. The announcements can posted on each level in the "Manage" menu, using "Post announcement":

It can be also added using the admin interface:
The announcements are then shown based on their specified context:

No context specified

  Shown on dashboard (landing page).

Project specified

  Shown within the project, including all its components and translations.

Component specified

  Shown for a given component and all its translations.

Language specified

  Shown on the language overview and all translations in that language.

This is how it looks on the language overview page:
2.20 Component Lists

Specify multiple lists of components to appear as options on the user dashboard, from which users can pick one as their default view. See Panel to learn more.

2.20 sürümünde değişti: A status will be presented for each component list presented on the dashboard.

The names and content of component lists can be specified in the admin interface, in Component lists section. Each component list must have a name that is displayed to the user, and a slug representing it in the URL.

2.13 sürümünde değişti: Change dashboard settings for anonymous users from the admin interface, altering what dashboard is presented to unauthenticated users.

2.20.1 Automatic component lists

2.13 sürümünde geldi.

Add components to the list automatically based on their slug by creating Automatic component list assignment rules.

- Useful for maintaining component lists for large installations, or in case you want to have one component list with all components on your Weblate installation.

İpucu: Make a component list containing all the components of your Weblate installation.

1. Define Automatic component list assignment with ^.*$ as regular expression in both the project and the component fields, as shown on this image:
Optional Weblate modules

Several optional modules are available for your setup.
2.21.1 Git exporter

2.10 sürümünde geldi.
Provides you read-only access to the underlying Git repository using HTTP(S).

**Installation**

1. Add `weblate.gitexport` to installed apps in `settings.py`:

```
INSTALLED_APPS += ("weblate.gitexport",)
```

2. Export existing repositories by migrating your database after installation:

```
weblate migrate
```

**Usage**

The module automatically hooks into Weblate and sets the exported repository URL in the `Component configuration`. The repositories are accessible under the `/git/` part of the Weblate URL, for example `https://example.org/git/weblate/main/`.

Repositories for publicly available projects can be cloned without authentication:

```
git clone 'https://example.org/git/weblate/main/
```

Access to browse the repositories with restricted access (with Private access control or when `REQUIRE_LOGIN` is enabled) requires an API token which can be obtained in your user profile:

```
git clone 'https://user:KEY@example.org/git/weblate/main/
```

**İpucu:** By default members or `Users` group and anonymous user have access to the repositories for public projects via Access repository and Power user roles.

2.21.2 Faturalandırma

2.4 sürümünde geldi.
This is used on Hosted Weblate to define billing plans, track invoices and usage limits.

**Installation**

1. Add `weblate.billing` to installed apps in `settings.py`:

```
INSTALLED_APPS += ("weblate.billing",)
```

2. Run the database migration to optionally install additional database structures for the module:

```
weblate migrate
```
Usage

After installation you can control billing in the admin interface. Users with billing enabled will get new Billing tab in their Kullanıcı profili.

The billing module additionally allows project admins to create new projects and components without being superusers (see Adding translation projects and components). This is possible when following conditions are met:

- The billing is in its configured limits (any overusage results in blocking of project/component creation) and paid (if its price is non zero)
- The user is admin of existing project with billing or user is owner of billing (the latter is necessary when creating new billing for users to be able to import new projects).

Upon project creation user is able to choose which billing should be charged for the project in case he has access to more of them.

2.21.3 Yasal

2.15 sürümünde geldi.

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


Most likely these will not be directly usable to you, but might come in handy as a starting point if adjusted to meet your needs.

Installation

1. Add weblate.legal to installed apps in settings.py:

```python
INSTALLED_APPS += ("weblate.legal",)

# Optional:
# Social auth pipeline to confirm TOS upon registration/subsequent sign in
SOCIAL_AUTH_PIPELINE += ("weblate.legal.pipeline.tos_confirm",)

# Middleware to enforce TOS confirmation of signed in users
MIDDLEWARE += [  
    "weblate.legal.middleware.RequireTOSMiddleware",  
]
```

2. Run the database migration to optionally install additional database structures for the module:

```bash
weblate migrate
```

3. Edit the legal documents in the weblate/legal/templates/legal/ folder to match your service.
Usage

After installation and editing, the legal documents are shown in the Weblate UI.

2.21.4 Avatars

Avatars are downloaded and cached server-side to reduce information leaks to the sites serving them by default. The built-in support for fetching avatars from e-mails addresses configured for it can be turned off using `ENABLE_AVATARS`.

Weblate currently supports:

- Gravatar
- Libravatar

**Ayrıca bakınız:**

*Avatar ön bellekleme, AVATAR_URL_PREFIX, ENABLE_AVATARS*

2.21.5 Spam protection

You can protect against spamming by users by using the Akismet service.

1. Install the `akismet` Python module (this is already included in the official Docker image).
2. Obtain the Akismet API key.
3. Store it as `AKISMET_API_KEY` or `WEBLATE_AKISMET_API_KEY` in Docker.

Following content is sent to Akismet for checking:

- Kimliği doğrulanmamış kullanıcılardan öneriler
- Proje ve bileşen açıklamaları ve bağlantılar

**Not:** This (among other things) relies on IP address of the client, please see `Running behind reverse proxy` for properly configuring that.

**Ayrıca bakınız:**

*Running behind reverse proxy, AKISMET_API_KEY, WEBLATE_AKISMET_API_KEY*

2.21.6 Signing Git commits with GnuPG

3.1 sürümünde geldi.

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. Alternativ olarak, şifrë kodlarını Webat ile kullanabilirsiniz. HOME = $DATA_DIR/home varsını kullanarak gpg kalıcı olarak kullanabilirsiniz.

Ayrıca bakınız:

WEBLATE_GPG_IDENTITY

2.21.7 Rate limiting

3.2 sürümünde değişti: The rate limiting now accepts more fine-grained configuration.

4.6 sürümünde değişti: The rate limiting no longer applies to superusers.

Several operations in Weblate are rate limited. At most RATELIMIT_ATTEMPTS attempts are allowed within RATELIMIT_WINDOW seconds. The user is then blocked for RATELIMIT_LOCKOUT. There are also settings specific to scopes, for example RATELIMIT_CONTACT_ATTEMPTS or RATELIMIT_TRANSLATE_ATTEMPTS. The table below is a full list of available scopes.

The following operations are subject to rate limiting:

<table>
<thead>
<tr>
<th>Adı</th>
<th>Kapsam</th>
<th>Allowed attempts</th>
<th>Rate limit window</th>
<th>Lockout period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kayıt</td>
<td>REGISTRATION</td>
<td>5</td>
<td>300</td>
<td>600</td>
</tr>
<tr>
<td>Sending message to admins</td>
<td>MESSAGE</td>
<td>5</td>
<td>300</td>
<td>600</td>
</tr>
<tr>
<td>Password authentication on sign in</td>
<td>LOGIN</td>
<td>5</td>
<td>300</td>
<td>600</td>
</tr>
<tr>
<td>Sitewide search</td>
<td>SEARCH</td>
<td>6</td>
<td>60</td>
<td>600</td>
</tr>
<tr>
<td>Translating</td>
<td>TRANSLATE</td>
<td>30</td>
<td>60</td>
<td>600</td>
</tr>
<tr>
<td>Adding to glossary</td>
<td>GLOSSARY</td>
<td>30</td>
<td>60</td>
<td>600</td>
</tr>
<tr>
<td>Yeni bir dille çeviri başlatma</td>
<td>LANGUAGE</td>
<td>2</td>
<td>300</td>
<td>600</td>
</tr>
<tr>
<td>Yeni proje oluşturma</td>
<td>PROJECT</td>
<td>5</td>
<td>600</td>
<td>600</td>
</tr>
</tbody>
</table>
If a user fails to log in `AUTH_LOCK_ATTEMPTS` times, password authentication will be turned off on the account until having gone through the process of having its password reset.

The settings can be also applied in the Docker container by adding `WEBLATE_` prefix to the setting name, for example `RATELIMIT_ATTEMPTS` becomes `WEBLATE_RATELIMIT_ATTEMPTS`.

The API has separate rate limiting settings, see `API rate limiting`.

**Ayrıca bakınız:**

*Rate limiting, Running behind reverse proxy, API rate limiting*

### 2.21.8 Fedora Mesajlaşma tümleşimi

Fedora Mesajlaşma, Weblate’te gerçekleşen tüm değişiklikler için AMQP tabanlı bir yayıncıdır. Bunu kullanarak Weblate’te meydana gelen değişikliklere ek hizmetler bağlayabilirsiniz.


### 2.22 Customizing Weblate

Extend and customize using Django and Python. Contribute your changes upstream so that everybody can benefit. This reduces your maintenance costs; code in Weblate is taken care of when changing internal interfaces or refactoring the code.

**Uyarı:** Neither internal interfaces nor templates are considered a stable API. Please review your own customizations for every upgrade, the interfaces or their semantics might change without notice.

**Ayrıca bakınız:**

*Weblate’e Katkıda Bulunma*

#### 2.22.1 Creating a Python module

If you are not familiar with Python, you might want to look into Python For Beginners, explaining the basics and pointing to further tutorials.

To write some custom Python code (called a module), a place to store it is needed, either in the system path (usually something like `/usr/lib/python3.7/site-packages/`) or in the Weblate directory, which is also added to the interpreter search path.

Better yet, turn your customization into a proper Python package:

1. Create a folder for your package (we will use `weblate_customization`).
2. Within it, create a `setup.py` file to describe the package:

```python
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",
)```

(sonraki sayfada devam)
3. Create a folder for the Python module (also called `weblate_customization`) for the customization code.

4. Within it, create a `__init__.py` file to make sure Python can import the module.

5. This package can now be installed using `pip install -e`. More info to be found in “Editable” Installs.

6. Once installed, the module can be used in the Weblate configuration (for example `weblate_customization.checks.FooCheck`).

Your module structure should look like this:

```bash
weblate_customization
├── setup.py
└── weblate_customization
    ├── __init__.py
    ├── addons.py
    └── checks.py
```

You can find an example of customizing Weblate at <https://github.com/WeblateOrg/customize-example>, it covers all the topics described below.

### 2.22.2 Changing the logo

1. Create a simple Django app containing the static files you want to overwrite (see Creating a Python module).

   Branding appears in the following files:
   - `icons/weblate.svg` Logo shown in the navigation bar.
   - `logo-*.png` Web icons depending on screen resolution and web-browser.
   - `favicon.ico` Web icon used by legacy browsers.
   - `weblate-*.png` Avatars for bots or anonymous users. Some web-browsers use these as shortcut icons.
   - `email-logo.png` Used in notifications e-mails.

2. Add it to `INSTALLED_APPS`:

   ```python
   INSTALLED_APPS = (  
     # Add your customization as first  
     "weblate_customization",  
     # Weblate apps are here...  
   )
   ```

3. Run `weblate collectstatic --noinput`, to collect static files served to clients.

Ayrıca bakınız:

Managing static files (e.g. images, JavaScript, CSS), Serving static files
2.22.3 Custom quality checks, addons and auto-fixes

To install your code for Custom automatic fixups, Writing own checks or Eklenti yazma in Weblate:

1. Place the files into your Python module containing the Weblate customization (see Creating a Python module).
2. Add its fully-qualified path to the Python class in the dedicated settings (WEBLATE_ADDONS, CHECK_LIST or AUTOFIX_LIST):

```
# Checks
CHECK_LIST += ("weblate_customization.checks.FooCheck",)
# Autofixes
AUTOFIX_LIST += ("weblate_customization.autofix.FooFixer",)
# Add-ons
WEBLATE_ADDONS += ("weblate_customization.addons.ExamplePreAddon",)
```

Ayrıca bakınız:
Custom automatic fixups, Writing own checks, Eklenti yazma, Eklentiden betikleri çalıştırma

2.23 Management interface

The management interface offer administration settings under the /manage/ URL. It is available for users signed in with admin privileges, accessible by using the wrench icon top right:

![Management interface](image)

It includes basic overview of your Weblate:

- Support status, see Getting support for Weblate
- Backups, see Backing up and moving Weblate
- Paylaşılan çeviri belleği, bkz: Çeviri Bellegi
- Performance report to review Weblate health and length of Celery queues

2.23 Management interface 367
• SSH anahtar yönetimi, bkz: SSH repositories
• Alerts overview for all components, see alerts

2.23.1 The Django admin interface

Uyarı: Will be removed in the future, as its use is discouraged—most features can be managed directly in Weblate.

Here you can manage objects stored in the database, such as users, translations and other settings:
## Webtranslate Administration

### Site Administration

<table>
<thead>
<tr>
<th>Section</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repositories</td>
<td></td>
</tr>
<tr>
<td>Languages</td>
<td></td>
</tr>
<tr>
<td>Projects</td>
<td></td>
</tr>
<tr>
<td>Settings</td>
<td></td>
</tr>
<tr>
<td>Translations</td>
<td></td>
</tr>
</tbody>
</table>

### Recent Actions

- My actions
- None available

---

### 2.23. Management Interface

- Welcome to Weblate
- Documentation
- Change password
- Sign out
In the *Reports* section, you can check the status of your site, tweak it for *Production setup*, or manage SSH keys used to access *Accessing repositories*.

Manage database objects under any of the sections. The most interesting one is probably *Weblate translations*, where you can manage translatable projects, see *Project configuration* and *Component configuration*.

*Weblate languages* holds language definitions, explained further in *Dil tanımları*.

### Adding a project

Adding a project serves as container for all components. Usually you create one project for one piece of software, or book (See *Project configuration* for info on individual parameters):

**Weblate administration**

[Add Project form]

**Add Project**

Required fields are marked in bold.

- **Project name:** Webblate
- **URL slug:** weblate
- **Translation instructions:**
  
  You can use Markdown and mention users by @username.

- **Access control:**
  
  - **Protected**

- **Language aliases:**
  
  Comma-separated list of language code mappings, for example: en,DE,es,US,en

[Save and add another, Save and continue editing, SAVE]
Ayrıca bakınız:

Project configuration

Bilingual components

Once you have added a project, translation components can be added to it. (See Component configuration for info regarding individual parameters):
Ayrıca bakınız:

Component configuration, Bilingual and monolingual formats

**Monolingual components**

For easier translation of these, provide a template file containing the mapping of message IDs to its respective source language (usually English). (See Component configuration for info regarding individual parameters):
**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/](https://weblate.org/support/).

### 2.24.1 Integrating support

3.8 sürümünde geldi.

Purchased support packages can optionally be integrated into your Weblate subscription management interface, from where you will find a link to it. Basic instance details about your installation are also reported back to Weblate this way.

### 2.24.2 Data submitted to the Weblate

- URL where your Weblate instance is configured
- Your site title
- The Weblate version you are running
- Tallies of some objects in your Weblate database (projects, components, languages, source strings and users)
- The public SSH key of your instance

Additionally, when *Weblate’ı Keşfedin* is turned on:

- List of public projects (name, URL and website)

No other data is submitted.
2.24.3 Integration services

- See if your support package is still valid
- Weblate provisioned backup storage
- Weblate’i Keşfedin

İpucu: Purchased support packages are already activated upon purchase, and can be used without integrating them.

2.24.4 Weblate’i Keşfedin

4.5.2 sürümünde geldi.

Not: Bu özellik şu anda erken beta aşamasındadır.


Listelenme

İpucu: Weblate’i Keşfet’e katılmak, Weblate’in sunucunuz hakkında bazı bilgiler göndermesini sağlar, lütfen bkz: Data submitted to the Weblate.

Weblate’i Keşfet’te sunucunuzu etkin bir destek aboneliği ile listelemek için (bkz: Integrating support) yönetim panelinden bunu açmanız yeterlidir:
Weblate'i Keşfet'te sunununuz destek aboneliği olmadan listeleme:

1. <https://weblate.org/user/> adresinde kaydolun
2. Weblate sununuzu <https://weblate.org/subscription/discovery/> adresindeki keşif veri tabanına kaydédin
3. Weblate’inizde hizmet etkinleştirmesini onaylayın ve Keşfi etkinleştir düğmesini kullanarak Weblate yönetim sayfanızda keşif listesini açın:
Listelemeyi özelleştirme

<https://weblate.org/user/> adresinde bir metin ve resim (570 x 260 piksel) sağlayarak listelemeyi özelleştirebilirsiniz.

2.25 Legal documents

Not: Herein you will find various legal information you might need to operate Weblate in certain legal jurisdictions. It is provided as a means of guidance, without any warranty of accuracy or correctness. It is ultimately your responsibility to ensure that your use of Weblate complies with all applicable laws and regulations.

2.25.1 ITAR and other export controls

Weblate can be run within your own datacenter or virtual private cloud. As such, it can be used to store ITAR or other export-controlled information, however, end users are responsible for ensuring such compliance.

The Hosted Weblate service has not been audited for compliance with ITAR or other export controls, and does not currently offer the ability to restrict translations access by country.
2.25.2 US encryption controls

Weblate does not contain any cryptographic code, but might be subject export controls as it uses third party components utilizing cryptography for authentication, data-integrity and confidentiality.

Most likely Weblate would be classified as ECCN 5D002 or 5D992 and, as publicly available libre software, it should not be subject to EAR (see Encryption items NOT Subject to the EAR).

Software components used by Weblate (listing only components related to cryptographic function):

**Python** See https://wiki.python.org/moin/PythonSoftwareFoundationLicenseFaq#Is_Python_subject_to_export_laws.3F

**GnuPG** Optionally used by Weblate

**Git** Optionally used by Weblate

**curl** Used by Git

**OpenSSL** Used by Python and cURL

The strength of encryption keys depends on the configuration of Weblate and the third party components it interacts with, but in any decent setup it will include all export restricted cryptographic functions:

- In excess of 56 bits for a symmetric algorithm
- Factorisation of integers in excess of 512 bits for an asymmetric algorithm
- Computation of discrete logarithms in a multiplicative group of a finite field of size greater than 512 bits for an asymmetric algorithm
- Discrete logarithms in a group different than above in excess of 112 bits for an asymmetric algorithm

Weblate doesn’t have any cryptographic activation feature, but it can be configured in a way where no cryptography code would be involved. The cryptographic features include:

- Accessing remote servers using secure protocols (HTTPS)
- Generating signatures for code commits (PGP)

**Ayrıca bakınız:**

Export Controls (EAR) on Open Source Software
3.1 Weblate’e Katkıda Bulunma

Weblate’i iyileştirmenin düzenelence yolu vardır. Kodlama, grafik tasarım, belgelendirme, sponsorluk veya fikir olsun, kendiniz rahat hissedeceğiniz birini seçebilirsiniz:

- Reporting issues in Weblate
- Weblate kodlarına katkıda bulunmaya başlama
- Weblate‘i çevirtme
- Weblate belgelendirmesine katkıda bulunun
- Weblate tartışmaları
- Weblate‘in geliştirilmesine maddi destek verme

3.1.1 Weblate’i çevirme

Weblate, Weblate’nin kendisi kullanılarak sürekli olarak çevrilmektedir. Weblate’i mümkün olduğu kadar çok dilde kullanabilir hale getirme çabasında yer almakta çekinmemeyin. Bu, Weblate‘i kullanıcılarına karşı çıkmaktadır!

Kaynak dizgi olası bir hata bulursanız, bunu Weblate düzenleyicisinde bir yorumda işaretleyebilirsiniz. Bu şekilde tartışılabilir ve düzeltilebilir. Emininiz, Kaynak dizgi konumu bölümündeki bağlantıyla tıklayabilir ve düzeltmenizle birlikte bir çekme isteği gönderbilirsiniz.

3.1.2 Weblate belgelendirmesine katkıda bulunun

İstediğiniz belgelendirme sayfasını iyileşirebilirsiniz. Bunu, sayfanın sağ üst köşesinde GitHub’da düzenle düşmesine tıklayarak kolayca yapın.

Lütfen yazarken şu yönergelere uyun:

1. Geçerliyse, belgelendirmenin bir kısmını kaldırmayın.

4. Belgeleri takip ederken açıklanan elemleri gerçekleştirek değişikliklerini doğrulayın.

5. Gözden geçirmeyi ve birleştiriyi daha kolay ve daha hızlı hale getirmek için çekme isteğini küçük parçalar halinde değişikliklerle gönderin.

6. Büyük bir makaleyi yeniden yazmak ve onun yapısını değiştirmek istiyorsanız, bunu iki adımda yapın:
   1. Yeniden yazın
   2. Yeniden yazma gözden geçirildikten, iyileştirildikten ve birleştirildikten sonra, başka bir çekme isteğinde paragrafların yapısını değiştirin.

İpucu: Belgeleri çevirebilirsiniz.

3.1.3 Yerleşik dil tanımlarını genişletme

Dil tanımları weblate-language-data deposunda bulunmaktadır.

Eksik dil tanımlarını languages.csv dosyasına ekleyebilirsiniz, diğer dosyalar bu dosyadan oluşturulmaktadır.

3.1.4 Weblate tartışmaları

Bir fikriniz varsa ve bir soruna uygun olmadığını emin değilseniz endişelenmeyin. Topluluğa GitHub tartışmalarda katılabilirsiniz.

3.1.5 Weblate’ın geliştirilmesine maddi destek verme

Weblate’ın geliştirilmesini bağış sayfasında destekleyebilirsiniz. Burada toplanan fonlar, övgür yazılım projelerine için ücretsiz barındırmaya sağlanmış ve Weblate’ın daha da geliştirilmesi için kullanılmaktadır. Fonlama hedefleri ve gururu bir fon sağlayıcı olarak alacağımız ödüller gibi seçenekler için lütfen bağış sayfasına bakın.

Backers who have funded Weblate

List of Weblate supporters:

• Yashiro Ccs
• Cheng-Chia Tseng
• Timon Reinhard
• Cassidy James
• Loic Dachary
• Marozed
• https://freedombox.org/
• GNU Solidario (GNU Health)
• BallotReady
• Richard Nespithal
• MyExpenses.Mobi

Do you want to be in the list? Please see options on the Donate to Weblate.

3.1. Weblate’e Katkıda Bulunma 381
3.2 Weblate kodlarına katkıda bulunmaya başlama

Understand the Weblate source code by going through Weblate kaynak kodları, Weblate frontend and Weblate internals.

3.2.1 Starting with the codebase

Familiarize yourself with the Weblate codebase, by having a go at the bugs labelled good first issue.

3.2.2 Weblate’i yerel olarak çalıştırma

Weblate geliştirmeye başlamak için en rahat yaklaşım Installing from sources bölümündeki talimatları izlemektir. Bu, size düzenlenebilir Weblate kaynaklarını içeren bir sanal ortam sağlayacaktır.

1. Clone the Weblate source code:

```bash
git clone https://github.com/WeblateOrg/weblate.git
cd weblate
```

2. Create a virtualenv:

```bash
virtualenv .venv
.venv/bin/activate
```

3. Install Weblate (for this you need some system dependencies, see Installing from sources):

```bash
pip install -e .
```

3. Gelişirme için faydalı olacak tüm bağımlılıkları kurun:

```bash
pip install -r requirements-dev.txt
```

4. Bir geliştirme sunucusu başlatın:

```bash
weblate runserver
```

5. Depending on your configuration, you might also want to start Celery workers:

```bash
./weblate/examples/celery start
```

6. To run a test (see Local testing for more details):

```bash
./scripts/test-database
./manage.py test
```

Ayrıca bakınız:

Installing from sources
3.2.3 Weblate’i Docker’da yerel olarak çalıştırma

If you have Docker and docker-compose installed, you can spin up the development environment by simply running:

```sh
./rundev.sh
```

It will create a development Docker image and start it. Weblate is running on <http://127.0.0.1:8080/> and you can sign in as the user admin using admin as the password. The new installation is empty, so you might want to continue with Adding translation projects and components.

The Dockerfile and docker-compose.yml for this are located in the dev-docker directory. The script also accepts some parameters, to execute tests, run it with the test parameter and then specify any test parameters, for example running only tests in the weblate.machine module:

```sh
./rundev.sh test --failfast weblate.machine
```

**Not:** Be careful that your Docker containers are up and running before running the tests. You can check that by running the `docker ps` command.

To display the logs:

```sh
./rundev.sh logs
```

To stop the background containers, run:

```sh
./rundev.sh stop
```

Running the script without arguments will re-create the Docker container and restart it.

**Not:** This is not a suitable setup for production, as it includes several hacks which are insecure, but they make development easier.

3.2.4 Coding Weblate with PyCharm

PyCharm is a known IDE for Python, here are some guidelines to help you set up your Weblate project in it.

Considering you have just cloned the GitHub repository to a folder, just open it with PyCharm. Once the IDE is open, the first step is to specify the interpreter you want to use:
You can either choose to let PyCharm create the virtualenv for you, or select an already existing one:

Don’t forget to install the dependencies once the interpreter is set: Either through the console (the console from the IDE will directly use your virtualenv by default), or through the interface when you get a warning about missing dependencies.

The second step is to set the right info to use Django natively inside PyCharm: The idea is to be able to immediately trigger the unit tests in the IDE. For that you need to specify the root path of the Django project and the path to its settings:
Be careful, the *Django project root* is the actual root of the repository, not the Weblate sub-directory. About the settings, you could use the `weblate/settings_test.py` from the repository, but you could create your own setting and set it there.

The last step is to run the server and to put breakpoints in the code to be able to debug it. This is done by creating a new *Django Server* configuration:

![Django Server configuration](image)

```
Unamed
```

Save 'Test: weblate.formatstests.test_exporters.CSVExporterTest' Configuration

```
Unnamed
```

Test: weblate.formatstests.test_exporters.CSVExporterTest
**İpucu:** Be careful with the property called *No reload*: It prevents the server from being reloaded live if you modify files. This allows the existing debugger breakpoints to persist, when they normally would be discarded upon reloading the server.

### 3.2.5 Bootstrapping your devel instance

You might want to use *import_demo* to create demo translations and *createadmin* to make an admin user.

### 3.3 Weblate kaynak kodları


**Ayrıca bakınız:**
Weblate’ın içeren nasıl görüldüğünü görmek için Weblate internals bölümüne bakın.
3.3.1 Security by Design Principles

Any code for Weblate should be written with Security by Design Principles in mind.

3.3.2 Coding standard

The code should follow PEP-8 coding guidelines and should be formatted using black code formatter.

To check the code quality, you can use flake8, the recommended plugins are listed in .pre-commit-config.yaml and its configuration is placed in setup.cfg.

The easiest approach to enforce all this is to install pre-commit. Weblate repository contains configuration for it to verify the committed files are sane. After installing it (it is already included in the requirements-lint.txt) turn it on by running pre-commit install in Weblate checkout. This way all your changes will be automatically checked.

You can also trigger check manually, to check all files run:

```
pre-commit run --all
```

3.4 Debugging Weblate

Bugs can behave as application crashes or as misbehavior. You are welcome to collect info on any such issue and submit it to the issue tracker.

3.4.1 Hata ayıklama kipi

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.

Ayrıca bakınız:

Disable debug mode

3.4.2 Weblate logs

Weblate can produce detailed logs of what is going in the background. In the default configuration it uses syslog and that makes the log appear either in /var/log/messages or /var/log/syslog (depending on your syslog daemon configuration).

The Celery process (see Background tasks using Celery) usually produces own logs as well. The example system-wide setups log to several files under /var/log/celery/.

Docker containers log to their output (as usual in the Docker world), so you can look at the logs using docker-compose logs.

Ayrıca bakınız:

Sample configuration contains LOGGING configuration.
3.4.3 Not processing background tasks

Lot of things happen in background Celery workers. In case things like sending out e-mails or component removal does not work, there might be some issue with it.

Bu durumda denetlenecek olanlar:

- Check Celery process is running, see Background tasks using Celery
- Check Celery queue status either in Management interface or using celery_queues
- Look into Celery logs for errors (see Weblate logs)

3.4.4 Not receiving e-mails from Weblate

You can verify whether outgoing e-mail is working correctly by using the sendtestemail management command (see Invoking management commands for instructions on how to invoke it in different environments) or using Management interface under the Tools tab.

These send e-mail directly, so this verifies that your SMTP configuration is correct (see Configuring outgoing e-mail). Most of the e-mails from Weblate are however sent in the background and there might be some issues with Celery involved as well, please see Not processing background tasks for debugging that.

3.4.5 Analyzing application crashes

In case the application crashes, it is useful to collect as much info about the crash as possible. The easiest way to achieve this is by using third-party services which can collect such info automatically. You can find info on how to set this up in Collecting error reports.

3.4.6 Silent failures

Lots of tasks are offloaded to Celery for background processing. Failures are not shown in the user interface, but appear in the Celery logs. Configuring Collecting error reports helps you to notice such failures easier.

3.4.7 Performance issues

In case Weblate performs badly in some situation, please collect the relevant logs showing the issue, and anything that might help figuring out where the code might be improved.

In case some requests take too long without any indication, you might want to install dogslove along with Collecting error reports and get pinpointed and detailed tracebacks in the error collection tool.

3.5 Weblate internals

Not: This chapter will give you basic overview of Weblate internals.

Weblate derives most of its code structure from, and is based on Django.
3.5.1 Directory structure

Quick overview of directory structure of Weblate main repository:

**docs** Source code for this documentation, which can be built using Sphinx.

**dev-docker** Docker code to run development server, see *Weblate'i Docker'da yerel olarak çalıştırma*.

**weblate** Source code of Weblate as a Django application, see *Weblate internals*.

**weblate/static** Client files (CSS, Javascript and images), see *Weblate frontend*.

3.5.2 Modules

Weblate consists of several Django applications (some optional, see *Optional Weblate modules*):

**accounts**  
User account, profiles and notifications.

**addons**  
Add-ons to tweak Weblate behavior, see *Eklentiler*.

**api**  
API based on Django REST framework.

**auth**  
Authentication and permissions.

**billing**  
The optional *Faturalandırma* module.

**checks**  
Translation string *Quality checks* module.

**fonts**  
Font rendering checks module.

**formats**  
File format abstraction layer based on translate-toolkit.

**gitexport**  
The optional *Git exporter* module.

**lang**  
Module defining language and plural models.

**legal**  
The optional *Yasal* module.

**machinery**  
Integration of machine translation services.

**memory**  
Dahili çeviri belleği, bkz *Çeviri Belleği*.

**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 Eklenti geliştirme

**Eklentiler** Weblate‘de yerelleştirme iş akışını özelleştirmenin bir yoldur.

```python
class weblate.addons.base.BaseAddon (storage=None)

    @classmethod
can_install (component, user)
        Eklentinin verilen bileşenle uyumlu olup olmadığını denetle.

    configure (settings)
        Yapılandırmayı kaydet.

    daily (component)
        Günlük tetiklenen kanca.

    @classmethod
get_add_form (user, component, **kwargs)
        Yeni eklenti eklemek için yapilandırma formunu döndür.

    get_settings_form (user, **kwargs)
        Bu eklenti için yapilandırma formunu döndür.

    post_add (translation)
        Yeni çeviri eklendikten sonra tetiklenen kanca.

    post_commit (component)
        Değişiklikler depoya işledikten sonra tetiklenen kanca.

    post_push (component)
        Depo yukarı akışa yollandıktan sonra tetiklenen kanca.

    post_update (component, previous_head: str, skip_push: bool)
        Depo yukarı akışta güncellemekten sonra tetiklenen kanca.

    save_state ()
        Eklenti durum bilgilerini kaydet.
```

**Parametreler**

- **previous_head (str)** – Güncellemeden önce deponun HEAD’si, ilk klonlamada boş olabilir.
- **skip_push (bool)** – Eklenti işleminin değişiklikleri yukarı akışa yollamayı atlayıp atlamayacağı. Genellikle bunu temeldeki yöntemlere commit_and_push veya commit_pending olarak iletebilirsiniz.

**pre_commit (translation, author)**

Değişiklikler depoya işlenmeden önce tetiklenen kanca.

**pre_push (component)**

Depo yukarı akışa yollanmadan önce tetiklenen kanca.

**pre_update (component)**

Depo yukarı akışta güncellemeden önce tetiklenen kanca.

**save_state ()**

Eklenti durum bilgilerini kaydet.
stay_on_create = False
Weblate ekentileri için temel sınıf.

store_post_load (translation, store)
Bir dosya ayrıştırıldından sonra tetiklenen kanca.

Argüman olarak bir dosya biçimi sınıfının bir örneğini alır.
Bu, örneğin dosyanın nasıl kaydedileceğini ayarlamak gibi dosya biçimi sınıfı parametrelerini değiştirmek için faydalıdır.

unit_pre_create (unit)
Yeni birim oluşturulmadan önce tetiklenen kanca.

İşte örnek bir eklenti:

```python
from django.utils.translation import gettext_lazy as _
from weblate.addons.base import BaseAddon
from weblate.addons.events import EVENT_PRE_COMMIT

class ExampleAddon(BaseAddon):
    # Filter for compatible components, every key is
    # matched against property of component
    compat = {
        "file_format": ["po", "po-mono"]
    }
    # List of events add-on should receive
    events = (EVENT_PRE_COMMIT,)
    # Add-on unique identifier
    name = "weblate.example.example"
    # Verbose name shown in the user interface
    verbose = _("Example add-on")
    # Detailed add-on description
    description = _("This add-on does nothing it is just an example.")

    # Callback to implement custom behavior
    def pre_commit(self, translation, author):
        return
```

3.6. Eklenti geliştirme 391
3.7 Weblate frontend

The frontend is currently built using Bootstrap, jQuery and few third party libraries.

3.7.1 Desteklenen tarayıcılar

Weblate supports the latest, stable releases of all major browsers and platforms. Alternative browsers which use the latest version of WebKit, Blink, or Gecko, whether directly or via the platform’s web view API, are not explicitly supported. However, Weblate should (in most cases) display and function correctly in these browsers as well. Older browsers might work, but some features might be limited.

3.7.2 Dependency management

The yarn package manager is used to update third party libraries. The configuration lives in `scripts/yarn` and there is a wrapper script `scripts/yarn-update` to upgrade the libraries, build them and copy to correct locations in `weblate/static/vendor`, where all third party frontend code is located. The Weblate specific code should be placed directly in `weblate/static` or feature specific subdirectories (for example `weblate/static/editor`).

Adding new third-party library typically consists of:

```
# Add a yarn package
yarn --cwd scripts/yarn add PACKAGE

# Edit the script to copy package to the static folder
edit scripts/yarn-update

# Run the update script
./scripts/yarn-update

# Add files to git
git add .
```

3.7.3 Coding style

Weblate relies on Prettier for the code formatting for both JavaScript and CSS files. We also use ESLint to check the JavaScript code.

3.7.4 Yerelleştirme

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'
```

Ayrıca bakınız:

Translation topic in the Django documentation
3.7.5 Icons

Weblate currently uses material design icons. In case you are looking for new symbol, check Material Design Icons or Material Design Resources.

Additionally, there is scripts/optimize-svg to reduce size of the SVG as most of the icons are embedded inside the HTML to allow styling of the paths.

3.8 Reporting issues in Weblate

Weblate hatat takibi GitHub üzerinde barındırılıyor.

Feel welcome to report any issues you have, or suggest improvement for Weblate there. There are various templates prepared to comfortably guide you through the issue report.

If what you have found is a security issue in Weblate, please consult the Güvenlik sorunları section below.

If you are not sure about your bug report or feature request, you can try Weblate tartışmaları.

3.8.1 Güvenlik sorunları

In order to give the community time to respond and upgrade, you are strongly urged to report all security issues privately. HackerOne is used to handle security issues, and can be reported directly at HackerOne. Once you submit it there, community has limited but enough time to solve the incident.

Alternatively, report to security@weblate.org, which ends up on HackerOne as well.

If you don't want to use HackerOne, for whatever reason, you can send the report by e-mail to michal@cihar.com. You can choose to encrypt it using this PGP key 3CB 1DF1 EF12 CF2A C0EE 5A32 9C27 B313 42B7 511D. You can also get the PGP key from Keybase.

**Not:** Weblate depends on third-party components for many things. In case you find a vulnerability affecting one of those components in general, please report it directly to the respective project.

Some of these are:

- Django
- Django REST framework
- Python Social Auth

3.9 Weblate testsuite and continuous integration

Testsuites exist for most of the current code, increase coverage by adding testcases for any new functionality, and verify that it works.
### 3.9.1 Continuous integration

Current test results can be found on GitHub Actions and coverage is reported on Codecov.

There are several jobs to verify different aspects:

- Unit tests
- Documentation build and external links
- Migration testing from all supported releases
- Code linting
- Setup verification (ensures that generated dist files do not miss anything and can be tested)

The configuration for the CI is in `.github/workflows` directory. It heavily uses helper scripts stored in `ci` directory. The scripts can be also executed manually, but they require several environment variables, mostly defining Django settings file to use and database connection. The example definition of that is in `scripts/test-database`:

```bash
# Simple way to configure test database from environment
# Database backend to use postgresql / mysql / mariadb
export CI_DATABASE=$((1:postgresql))

# Database server configuration
export CI_DB_USER=weblate
export CI_DB_PASSWORD=weblate
export CI_DB_HOST=127.0.0.1

# Django settings module to use
export DJANGO_SETTINGS_MODULE=weblate.settings_test
```

The simple execution can look like:

```bash
./scripts/test-database
./ci/run-migrate
./ci/run-test
./ci/run-docs
```

### 3.9.2 Local testing

To run a testsuite locally, use:

```bash
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py test
```

**Ipucu:** 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:

```bash
# Simple way to configure test database from environment
# Database backend to use postgresql / mysql / mariadb
export CI_DATABASE=$((1:postgresql))

# Database server configuration
export CI_DB_USER=weblate
```

(sonraki sayfada devam)
Prior to running tests you should collect static files as some tests rely on them being present:

```bash
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
```

You can also specify individual tests to run:

```bash
export DJANGO_SETTINGS_MODULE=weblate.settings_test
./manage.py test weblate.gitexport
```

**İpucu:** The tests can also be executed inside developer docker container, see [Weblate’i Docker’da yerel olarak çalışmaya](https://weblate.org/docs/).  

**Ayrıca bakınız:**  
See [Testing in Django](https://weblate.org/docs/) for more info on running and writing tests for Django.

## 3.10 Data schemas

Weblate uses JSON Schema to define layout of external JSON files.

### 3.10.1 Weblate Çeviri Belleği Şemasi

<table>
<thead>
<tr>
<th>tür</th>
<th>Çeviri Belleği Ögesi</th>
</tr>
</thead>
<tbody>
<tr>
<td>items</td>
<td>array</td>
</tr>
<tr>
<td>özellikler</td>
<td></td>
</tr>
<tr>
<td>• category</td>
<td>Dizgi Kategorisi</td>
</tr>
<tr>
<td></td>
<td>1 is global, 2 is shared, 10000000+ are project specific, 20000000+ are user specific</td>
</tr>
<tr>
<td></td>
<td>tür</td>
</tr>
<tr>
<td></td>
<td>örnekler</td>
</tr>
<tr>
<td></td>
<td>minimum</td>
</tr>
<tr>
<td></td>
<td>öntanımlı</td>
</tr>
<tr>
<td>• origin</td>
<td>The String Origin</td>
</tr>
<tr>
<td></td>
<td>Dizgi adı veya bileşen adı</td>
</tr>
<tr>
<td></td>
<td>tür</td>
</tr>
<tr>
<td></td>
<td>örnekler</td>
</tr>
<tr>
<td></td>
<td>öntanımlı</td>
</tr>
<tr>
<td>• source</td>
<td>Kaynak Dizgi</td>
</tr>
<tr>
<td></td>
<td>tür</td>
</tr>
<tr>
<td></td>
<td>örnekler</td>
</tr>
<tr>
<td></td>
<td>minLength</td>
</tr>
<tr>
<td></td>
<td>öntanımlı</td>
</tr>
<tr>
<td>• source_language</td>
<td>Kaynak Dili</td>
</tr>
<tr>
<td></td>
<td>tür</td>
</tr>
<tr>
<td></td>
<td>ISO 639-1 / ISO 639-2 / IETF BCP 47</td>
</tr>
</tbody>
</table>
Tablo 1 – önceki sayfadan devam

<table>
<thead>
<tr>
<th>Özellik</th>
<th>Tür</th>
<th>Örnekler</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td></td>
<td></td>
<td>^[^]+$</td>
</tr>
<tr>
<td>target_language</td>
<td></td>
<td></td>
<td>cs</td>
</tr>
</tbody>
</table>

Sonraki sayfaya devam

Ayrıca bakınız:
Çeviri Belleği, dump_memory, import_memory

3.10.2 Weblate kullanıcı verilerini dışa aktarma

https://weblate.org/schemas/weblate-userdata.schema.json

<table>
<thead>
<tr>
<th>Özellik</th>
<th>Tür</th>
<th>Nesne</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic</td>
<td>Temel</td>
<td>Nesne</td>
</tr>
<tr>
<td>username</td>
<td>Kullanıcı adı</td>
<td>Düzenli</td>
</tr>
<tr>
<td>full_name</td>
<td>Ad soyad</td>
<td>Düzenli</td>
</tr>
<tr>
<td>email</td>
<td>E-posta</td>
<td>Düzenli</td>
</tr>
<tr>
<td>date Joined</td>
<td>Katıldığ tarih</td>
<td>Düzenli</td>
</tr>
<tr>
<td>profile</td>
<td>Profil</td>
<td>Nesne</td>
</tr>
<tr>
<td>language</td>
<td>Dil</td>
<td>Düzenli</td>
</tr>
<tr>
<td>suggested</td>
<td>Önerilen dizgi sayısı</td>
<td>Sayı</td>
</tr>
</tbody>
</table>

sonraki sayfaya devam
<table>
<thead>
<tr>
<th>Özellik</th>
<th>Tür</th>
<th>Öntanımlı Değer</th>
<th>Öntanımlı Olmadı Değer</th>
</tr>
</thead>
<tbody>
<tr>
<td>translated</td>
<td>Çevrilen dizgi sayısı</td>
<td>integer</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>tür</td>
<td>integer</td>
<td>24</td>
</tr>
<tr>
<td>uploaded</td>
<td>Yüklenen ekran görüntü sayısı</td>
<td>integer</td>
<td>0</td>
</tr>
<tr>
<td>hi-de_completed</td>
<td>Tamamlanan çevirileri panelde gizle</td>
<td>boolean</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>örnekler</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>secondary_in_zon</td>
<td>İkincil çevirileri Zen kipinde göster</td>
<td>boolean</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>örnekler</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>hide_source_secondary</td>
<td>İkincil çeviri varsa kaynağı gizle</td>
<td>boolean</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>örnekler</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>editor_link</td>
<td>Düzenleyici bağlantısı</td>
<td>dizgi</td>
<td>^.*$</td>
</tr>
<tr>
<td>language</td>
<td>Çevrilen diller</td>
<td>array</td>
<td>cs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>^.*$</td>
</tr>
</tbody>
</table>

3.10. Data schemas
<table>
<thead>
<tr>
<th>İkincil dil turü</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>öntanımlı items</td>
<td>Dil kodu</td>
</tr>
<tr>
<td>örnekler</td>
<td>sk</td>
</tr>
<tr>
<td>pattern</td>
<td>^.*$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>İzlenen projeler turü</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>öntanımlı items</td>
<td>Proje kısaltması</td>
</tr>
<tr>
<td>örnekler</td>
<td>weblate</td>
</tr>
<tr>
<td>pattern</td>
<td>^.*$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Denetim günlüğü tur</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>öntanımlı items</td>
<td>Ögeler</td>
</tr>
<tr>
<td>örnekler</td>
<td>nesne</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP adresi turü</th>
<th>dizgi</th>
</tr>
</thead>
<tbody>
<tr>
<td>örnekler</td>
<td>127.0.0.1</td>
</tr>
<tr>
<td>pattern</td>
<td>^.*$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kullanıcı tanıtcı turü</th>
<th>dizgi</th>
</tr>
</thead>
<tbody>
<tr>
<td>örnekler</td>
<td>PC / Linux / Firefox 70.0</td>
</tr>
<tr>
<td>pattern</td>
<td>^.*$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Zaman damgası turü</th>
<th>dizgi</th>
</tr>
</thead>
<tbody>
<tr>
<td>örnekler</td>
<td>2019-11-18T18:58:30.845Z</td>
</tr>
<tr>
<td>pattern</td>
<td>^.*$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Etkinlik turü</th>
<th>dizgi</th>
</tr>
</thead>
<tbody>
<tr>
<td>örnekler</td>
<td>oturum aç</td>
</tr>
<tr>
<td>pattern</td>
<td>^.*$</td>
</tr>
</tbody>
</table>

Ayrıca bakınız:

Kullanıcı profili, dumpuserdata

Bölüm 3. Contributor docs
3.11 Releasing Weblate

3.11.1 Releasing schedule

Weblate has a two-month release cycle for releases (x.y). These are usually followed by a bunch of bugfix releases to fix issues which slip into them (x.y.z).

The change in the major version indicates that the upgrade process can not skip this version - you always have to upgrade to x.0 before upgrading to higher x.y releases.

**Ayrıca bakınız:**

*Upgrading Weblate*

3.11.2 Release planning

The features for upcoming releases are collected using GitHub milestones, you can see our roadmap at [https://github.com/WeblateOrg/weblate/milestones](https://github.com/WeblateOrg/weblate/milestones).

3.11.3 Release process

Things to check prior to release:

1. Check newly translated languages by `./scripts/list-translated-languages`.
2. Set final version by `./scripts/prepare-release`.
3. Make sure screenshots are up to date `make -C docs update-screenshots`.
4. Merge any possibly pending translations `wlc push; git remote update; git merge origin/weblate`

Perform the release:

5. Create a release `./scripts/create-release --tag` (see below for requirements).

Post release manual steps:

6. Update Docker image.
7. Close GitHub milestone.
8. Once the Docker image is tested, add a tag and push it.
9. Update Helm chart to new version.
10. Include new version in `.github/workflows/migrations.yml` to cover it in migration testing.
11. Increase version in the website download links.
12. Increase version in the repository by `./scripts/set-version`.

To create tags using the `./scripts/create-release` script you will need following:

- GnuPG with private key used to sign the release
- Push access to Weblate git repositories (it pushes tags)
- Configured hub tool and access to create releases on the Weblate repo
- SSH access to Weblate download server (the Website downloads are copied there)
3.12 Güvenlik ve gizlilik

**Tüyo:** Weblate’de güvenlik, kullanıcılarınızın gizliliğine değer veren bir ortam sağlar.

Development of Weblate adheres to the **Best Practices of the Linux Foundation’s Core Infrastructure Initiative.**

**Ayrıca bakınız:**

*Güvenlik sorunları*

3.12.1 Tracking dependencies for vulnerabilities

Security issues in our dependencies are monitored using **Dependabot.** This covers the Python and JavaScript libraries, and the latest stable release has its dependencies updated to avoid vulnerabilities.

**İpucu:** There might be vulnerabilities in third-party libraries which do not affect Weblate, so those are not addressed by releasing bugfix versions of Weblate.

3.12.2 Docker container security

The Docker containers are scanned using **Anchore** and **Trivy.**

This allows us to detect vulnerabilities early and release improvements quickly.

You can get the results of these scans at GitHub — they are stored as artifacts on our CI in the SARIF format (Static Analysis Results Interchange Format).

**Ayrıca bakınız:**

*Continuous integration*

3.13 Weblate hakkında

3.13.1 Proje hedefleri

Çevirmenlerin katkıda bulunmasını kolaylaştıran, geniş bir **Supported file formats** yelpazesini destekleyen stok **Sürüm denetimi tümleşimi** bütünleşmesine sahip web tabanlı sürekli yerelleştirme aracı.

3.13.2 Proje adı

“Weblate”, “web” ve “translate” kelimelerinin birleşimidir.
3.13.3 Proje web sitesi


3.13.4 Proje logoları


3.13.5 Liderlik

Bu proje, michal@cihar.com adresinden ulaşlabilen Michal Čihař tarafından sürdürülmektedir.

3.13.6 Yazarlar

Weblate, Michal Čihař tarafından başlatıldı. 2012 yılında kurulduğundan bu yana binlerce kişi katkıda bulundu.

3.14 Lisans

Copyright (C) 2012 - 2021 Michal Čihař <michal@cihar.com>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.
4.1 Weblate 4.9

Henüz yayınlanmamıştır.

4.2 Weblate 4.8.1

Released on September 10th 2021.

- Django yöneticisi arayüzünde kullanıcı kaldırma düzeltildi.
- Eklenti parametreleri daha ayrıntılı olarak belgelendirildi.
- Sözlükte JavaScript hatası düzeltildi.
- Tutarlılık denetimlerinde eşleşme sayısına sınır ekle.
- Makine çevirilerinde yer tutucuların işlemesi iyileştirildi.
- API kullanarak eklenti oluşturma düzeltildi.
- Altbilgiye gizlilik ilkesi bağlanış eklemek için PRIVACY_URL ayarı eklendi.
- Hide member e-mail addresses from project admins.
- Improved gettext PO merging in case of conflicts.
- Improved glossary highlighting.
- Improved safe-html flag behavior with XML checks.
- Fixed commit messages on linked components.

Ayrıntılı tüm değişiklikler.
4.3 Weblate 4.8

21 Ağustos 2021 tarihinde yayınlandı.

- Apple stringsdict biçimi için destek eklendi.
- Tam arama işlemi artık PostgreSQL'de büyük/küçük harfe duyarlı.
- Bazı durumlarda sözlük açıklamalarının kaydedilmesi düzlendi.
- Belgeleme iyileştirmeleri.
- Performans iyileştirmeleri.
- Gerrit ile sıkıştırma eklentisi uyumluluğu iyileştirildi.
- Tek dilli sözlük bileşenlere dijital ekleme sorunu düzlendi.
- Çeşitlerin işlenmesi performansı iyileştirildi.
- Sıkıştırma eklentisinin, bazen yukarı akış değişikliklerini ayrıtırmanızı atlaması düzlendi.
- İndirmeler için dosya uzantısını koru.
- Fluent biçimini için destek eklendi.
- JSON biçimlerini girintilemek için sekmeleri kullanma desteği eklendi.

Ayrıntılı tüm değişiklikler.

4.4 Weblate 4.7.2

15 Temmuz 2021 tarihinde yayınlandı.

- Bir projede daha fazla dil takma düzenin yapılandırılmasını destekle.
- API'de arama dizisi doğrulaması düzlendi.
- Bir etki alanı değişikliğinden sonra Git dışa aktarınca URL'leri düzlendi.
- Windows RC dosyaları için temizleme eklentisi düzlendi.
- XLIFF güncellemesinde olması çökme düzlendi.

Ayrıntılı tüm değişiklikler.

4.5 Weblate 4.7.1

30 Haziran 2021 tarihinde yayınlandı.

- Sözlüğe terim ekleme açıklar penceresi iyileştirildi.
- LibreTranslate makine çevirisi hizmeti için destek eklendi.
- Yeni projeler oluşturma hız sınırlaması düzlendi.
- Dosya güncellemelerinin performansı iyileştirildi.

Ayrıntılı tüm değişiklikler.
4.6 Weblate 4.7

17 Haziran 2021 tarihinde yayınlandı.

- Yapılandırma durum denetimi iyileştirildi.
- gettext PO'da kullanılan object-pascal-format için destek eklendi, bkz: Object Pascal biçimi.
- Amacı daha iyi açıklamak için Nearby keys, Similar keys olarak yeniden adlandırıldı.
- mi18n lang dosyaları için destek eklendi.
- SAML kimlik doğrulama bütünleştirmesi iyileştirildi.
- Ayrıca, 것으로 açıklanabilir düşündüğünde kullanılanlar davet etme sorunu düzeltildi.
- Weblate artık Django 3.2 gerektiriyor.
- E-posta kimlik doğrulaması devredışı bırakıldığından kullanıcıları davet etme sorunu düzeltildi.
- Dökümler paylaşılır.
- Kullanıcıların bir projeye katkıda bulunanların engellemek için destek eklendi.
- Sözler dillerini otomatik olarak oluşturup etme düzeltilti.
- Eklentiler hakkında belgelendirme genişletildi.
- Biçim senaryolar sahibi dillerdeki çözümlerini düzenledi.
- Ücretsiz DeepL API’si için destek eklendi.
- Kullanıcı yönetimi artık Django yönetici arayüzune ihtiyaç duymuyor.

Ayrıntılı tüm değişiklikler.

4.7 Weblate 4.6.2

8 Mayıs 2021 tarihinde yayınlandı.

- Paylaşılan bileşen projeler arasında taşındıktan sonra oluşan çözme düzeltildi.
- Boş özellikler dosyalarına yeni dizgilerin eklenmesi düzeltildi.
- Sağdan sola dillerinde kopya simgesi hizalama düzeltildi.
- Bilgi sekmesinde düzgi istatistikleri genişletildi.
- Git’te yok sayılan çeviri dosyalarının işlemesi düzeltildi.
- Ölçüm performansı iyileştirildi.
- Sözlerin kaydetmedeki olası hata düzeltildi.
- Farklı çoğul kuralları sahip dillerde tutarlılık denetimi davranışı düzeltildi.

Ayrıntılı tüm değişiklikler.

Ayrıntılı tüm değişiklikler.
4.8 Weblate 4.6.1

2 Mayıs 2021 tarihinde yayınlandı.

- Eski istenmeyen ileti koruma kodunu kaldırdı.
- Kaynak çokul denetimi doğruluğu iyileştirildi.
- Docker’da kullanıcı arayüzü dillerinin listesi güncellendi.
- Çekme istekleri oluşturulken alan hata mesajları iyileştirildi.
- Pagure’de çekme istekleri oluşturma sorunu düzeltildi.
- Otomatik olarak kurulan eklentilerin tetiklenmesi düzeltildi.
- Yükeltme sırasında olası önbelge alma sorunları düzeltildi.
- Yüklemeyi kullanarak tek dilli çevirilere yeni birimler eklenmesi düzeltildi.

Ayrıntılı tüm değişiklikler.

4.9 Weblate 4.6

19 Nisan 2021 tarihinde yayınlandı.

- auto_translate yönetim komutu artık çeviri modunu belirtmek için bir parametreye sahip.
- Metin dosyaları için destek eklendi.
- Tüm nesneler için eğilimler ve ölçümler eklendi.
- İkincil dillerden doğrudan metin kopyalama desteği eklendi.
- Değişikliklere göz atarken tarihi filtreleme eklendi.
- Etkinlik çizelgeleri iyileştirildi.
- İletişim formu e-postaları için gönderen artık yapılabilir.
- Bileşen oluşturma API’sinde parametre doğrulaması iyileştirildi.
- The rate limiting no longer applies to superusers.
- Otomatik çeviri eklenmesi performansı ve güvenciliği iyileştirildi.
- Docker konteynerinde artık oran sınırlaması özelleştirilebilir.
- Bileşen oluşturma API’sı artık otomatik olarak Weblate internal URLs kullanıyor.
- Dizgileri listelerken durumun gösterilmesi basitleştirildi.
- Parola karma artık varsayılan olarak Argon2'yi kullanıyor.
- Çeviri durumunu gösteren ilerleme çubukları basitleştirildi.
- Amacı neleştirmek için Eksik dilleri ekle yeniden adlandırıldı.
- Dizgi durumunu XLIFF olarak kaydetme düzeltildi.
- Dil genelinde arama eklendi.
- Docker dağıtımımda Yatay ölçeklendirme için ilk destek.

Ayrıntılı tüm değişiklikler.
4.10 Weblate 4.5.3

1 Nisan 2021 tarihinde yayınlandı.

• Ölçümlerin toplanması düzeltildi.
• Dizgleler eklken olası çıkma düzeltildi.
• Ara sorgusu örnekleri iyileştirildi.
• Değiştirme yüklemesinde yeni eklenen dizglerin olası kaybı düzeltildi.

4.11 Weblate 4.5.2

26 Mart 2021 tarihinde yayınlandı.

• Otomatik çeviri için yapılandırılabilir zamanlama.
• Lua biçim denetimi ekledi.
• *Art arda yineленen sözçüker* denetiminde biçim dizglerni yok say.
• Çeviri sayfasında ekran görüntüsü yüklemeye izin ver.
• Depo bakımlarında zorunlu dosya eş zamanlanmasını ekledi.
• Daha uzun kodlul diller için otomatik önerler düzeltildi.
• Yeni dizglere ekleme performansı iyileştirildi.
• Nitelik denetimlerinde çeşitli hata düzeltilerleri.
• Çeşitli performans iyileştirmeleri.
• Weblate’i Keşfedin ile bütünleştirme ekledi.
• Salt okunur dizglerele denetim davranış düzeltildi.

Ayrıntılı tüm değişiklikler.

4.12 Weblate 4.5.1

5 Mart 2021 tarihinde yayınlandı.

• Bazı uç durumlarda sözük işaretlerinin düzenlenmesi düzeltildi.
• Birkaç sayfanın performansını artırmak için ölçümlerin kullanımı genişletildi.
• TMX dosyalarında doğru kaynak dili kaydet.
• API kullanarak tek dilli PO yüklenmelerini daha iyi yönet.
• Sözük bileşenlerinde uyarı davranışı iyileştirildi.
• Markdown bağlanç denetimleri iyileştirildi.
• İçerik haritalarında sözük ve kaynak dili belirt.
• Büyük projelerin bileşen listesinin sayfalandırılması.
• Çeviri, bileşen veya proje kaldırma performansı iyileştirildi.
• Toplu düzenleme performansı iyileştirildi.
• ODF dosyaları için “Düzenleme gerekli” ve “Onaylandı” durumlarının koruma düzeltildi.
• Çeviri dosyası indirmelerini özellikle arayüzü iyileştirildi.
Ayrıntılı tüm değişiklikler.

4.13 Weblate 4.5

19 Şubat 2021 tarihinde yayınlanıldı.

- gettext PO'ya kullanılan lua-format için destek eklendi.
- Bir bileşeni projeler arasında paylaşmak için destek eklendi.
- Birden çok biçim işaretliyle birden çok adsz değişken denetimi davranış düzeltildi.
- Posta listesi alanı projelerden kaldırdı, bunun yerine çevirmenler için genel talimatların kullanılması önerilir.
- Sözdə yerel oluştururma eklenmesi eklendi.
- TermBase eXchange dosyaları için destek eklendi.
- Bir işaret kullanarak dizgi çeşitleri el ile tanımlamak için destek eklendi.
- Tutarlılık denetimlerinin performansı iyileştiıldı.
- Uzun dizgiler için çeviri belleği performansı iyileştirildi.
- Açıklamalarda arama desteği eklendi.
- Dizgiler artık iki dilli biçimlerde de eklenebilir ve kaldırılabilir.
- Amazon Translate makine çevirisinde desteklenen dillerin listesini genişlet.
- Java Özellikleri için Java MessageFormat denetimlerini otomatik olarak etkinleştı.
- Bir çeviriye yeni dizgiler eklemek için yeni bir yükleme yöntem eklendi.
- Çeviriye göz atmak için basit bir arayüz eklendi.
- Sözlükler artık normal bileşenler olarak saklanıyor.
- Artık bileşen API’si kullanıldığı için sözlüklerde özel API kaldırıldı.
- Bazı bayrakları değiştirmek için basitleştirilmiş arayüz eklendi.
- Sözlükte çevrilemeyen veya yasak terimler için destek eklendi.
- Bir sözlükte terminoloji tanmlama desteği eklendi.
- Görsel klavye için daha fazla alan elde etmek için metin yönü geçişı taşıyঢ.
- Kullanıcının katkıda bulunduğu projeleri otomatik olarak izleme seçeneği eklendi.
- Çevirinin sözlükle eşleşip eşleşmediği denetimi eklendi.
- Gezinme metni rengini özelleştirme desteği eklendi.

Ayrıntılı tüm değişiklikler.

4.14 Weblate 4.4.2

14 Ocak 2021 tarihinde yayınlanıldı.

- Dağıtılan bir MO dosyasının bozulması düzeltildi.
4.15 Weblate 4.4.1

13 Ocak 2021 tarihinde yayınlandı.

- Çoklu değişiklikleri geri alma düzdüldü.
- Proje ayarlarına yardımcı görüntüleme düzdüldü.
- Kullanıcıların yönetilmesi iyileştirildi.
- Tek dilli PO dosyalarında bağlanan işlenmesi iyileştirildi.
- HTML, ODF, IDML ve Windows RC biçimleriyle temizleme eklentisi davranışı düzdüldü.
- CSV dosyalarından konunun ayrıştırılması düzdüldü.
- Dosya indirmeleri için içerik sıkıştırmasını kullan.
- ZIP dosyasından içe aktarma kullanıcı deneyimi iyileştirildi.
- Yüklemeler için dosya biçimlerinin algılanması iyileştirildi.
- Pagure’de yinelenen çekme isteklerinden kaçın.
- Hayalet çevirileri görüntüleme performansı iyileştirildi.
- Yerel tarayıcı metin almasını kullanması için çeviri düzenleyicisi yeniden uygulandı.
- Temizleme eklentisinin yeni dizgeler ekleme bozması düzdüldü.
- Eklentiler için API eklendi.

Ayrıntılı tüm değişiklikler.

4.16 Weblate 4.4

15 Aralık 2020 tarihinde yayınlandı.

- Bileşen oluştururken doğrulama iyileştirildi.
- Weblate artık Django 3.1 gerektiriyor.
- Yönetim arayüzünde görünüm özelleştirme desteği eklendi.
- Toplu düzenleme salt okunur durum işleme düzdüldü.
- CodeMirror bütünleştirilmesi iyileştirildi.
- Çeviri dosyalarından boş dizgileri kaldırmak için eklenti eklendi.
- Çeviriler için artık CodeMirror düzenleyicisi kullanılıyor.
- XML, HTML, Markdown ve reStructuredText için çeviri düzenleyicide söz dizimi vurgulama.
- Çeviri düzenleyicide yerleştirilebilir öğeleri vurgula.
- Standart olmayan dil kodları için destek iyileştirildi.
- Belirsiz dil kodlarını kullanırken uyarı eklendi.
- Kullanıcıya artık yeni bir çeviri eklernen filtrenlenmiş bir dil listesi sunuluyor.
- Geçmişte değişiklikler için arama yetenekleri genişletildi.
- Fatura ayırtı sayfaları ve özgür barındırma iş akışı iyileştirildi.
- Çeviri istatistikleri API’si genişletildi.
- Çeviri yaparken “diğer çeviriler” sekmesi iyileştirildi.
- Görevler API’si eklendi.
• Dosya yüklemelerinin performansı iyileştirildi.
• Kullanıcı tarafından tanımlanan özel karakterlerin gösterilmesi iyileştirildi.
• Otomatik çeviri performansı iyileştirildi.
• Kullanıcı arayüzünde birkaç küçük iyileştirme.
• ZIP indirmelerinin adlandırılması iyileştirildi.
• İzlenmeyen projelerde bildirim alma seçeneği eklendi.
  Ayrıntılı tüm değişiklikler.

4.17 Weblate 4.3.2

4 Kasım 2020 tarihinde yayınlandı.

• Belirli bileşen dosya maskelerinde çökme düzeltildi.
• Ard arda yinelenen sözcüklerin doğruluğu iyileştirildi.
• Pagure çekme istekleri için destek eklendi.
• Başarısız kayıtlar için hata mesajlarını iyileştirildi.
• Geliştirici yorumlarını Markdown olarak görüntüleme geri alındı.
• “master” dalından farklı bir varsayılan dala sahip Git depolarının kurulumu basitleştirildi.
• Yeni oluşturulan iç depolar artık varsayılan dal olarak main kullanıyor.
• reStructuredText çeviriırken değiştirilmemiş çevirinin yanlış algılanma oranı azaltıldı.
• Bazı durumlarda CodeMirror görüntüleme sorunları düzeltildi.
• Anlamını netleştirmek için Şablon grubu “Kaynaklar” olarak yeniden adlandırıldı.
• Daha uzun yolları olan depolarda GitLab çekme istekleri düzeltildi.
  Ayrıntılı tüm değişiklikler.

4.18 Weblate 4.3.1

21 Ekim 2020 tarihinde yayınlandı.

• Otomatik çeviri performansı iyileştirildi.
• Kimliği doğrulanmış kullanıcılar için oturum sona ermesi düzeltildi.
• Sürüm bilgilerini gizleme desteği eklendi.
• Bitbucket Sunucusu ile kanca uyumluğu iyileştirildi.
• Çeviri belleği güncellemelerinin performansı iyileştirildi.
• Bellek kullanımını azalttı.
• Matris görüntüümünün performansı iyileştirildi.
• Bir projeden kullanıcı kaldırmadan önce onay eklendi.
  Ayrıntılı tüm değişiklikler.
4.19 Weblate 4.3

15 Ekim 2020 tarihinde yayınlandı.

- Kullanıcı istatistikleri API’ye dahil edildi.
- Sayfalandırılmış sayfalardaki bileşen sıralaması düzeltildi.
- Bir sözleşk için kaynak dili tanımlama.
- GitHub ve GitLab çekme istekleri desteği yeniden yazıldı.
- Öneriyi kaldırdıktan sonra istatistik sayıları düzeltildi.
- Açık kullanıcı profili genişletildi.
- Zorunlu denetimlerin yapılandırılması düzeltildi.
- Yerleşik yedeklemelerle ilgili belgeler iyileştirildi.
- Kaynak dil özniteliği projeden bileșene taşındı.
- Vue 118n biçimlendirmesi denetimi eklendi.
- Genel yer tutucu denetimi artık düzelli ifadeleri destekliyor.
- Matris kipinin görüntümü artık düzenelli ifadeleri destekliyor.
- Makine çevirisi artık otomatik öneriler olarak adlandırılıyor.
- Birden çok GitLab veya GitHub örneğiyle etkileşim için destek eklendi.
- Proje güncellemelerini, birim güncellemeleri ve kaldırmalarını ve sözlükleri kapsayacak şekilde API genişletildi.
- Birim API’si artık çok dizgi düzenleri düzgün şekilde işliyor.
- Bileşen oluşturuma artık ZIP dosyası veya belge yüklemesini işleyebilir.
- API yanıt durum kodları birleştirildi.
- Katılımcı sözleşmesinde Markdown desteği.
- Kaynak dizgi izleme işleytiildi.
- JSON, YAML ve CSV biçimleri uyumluluğu işleytiildi.
- Dizgileri kaldırmada desteği eklendi.
- Dosya indirmelerinin performansı işleytiildi.
- Depo yönetimini görünümü işleytiildi.
- Android için java biçimini otomatik olarak etkinleştire.
- Yerelleştirilmiş ekran görüntüleri desteği eklendi.
- Python 3.9 desteği eklendi.
- Belirli koşullar altında HTML dosyalarının çevrilmesi düzeltildi.

Ayrıntılı tüm değişiklikler.
4.20 Weblate 4.2.2

2 Eylül 2020 tarihinde yayınlandı.

- JSON biçimleri için kaynak dizgilerinin eşleşmesi düzeltildi.
- Bazı kimlik doğrulama yapılandırmaları için oturum açma yeniden yönlendirmesi düzeltildi.
- Grup eşitleme ile LDAP kimlik doğrulaması düzeltildi.
- Otomatik çeviri ilerlemesini başarırlaştırmak için duruma dönüştürüldü.
- Artbilgiler etkinken Git işleme toplarlaması düzeltildi.
- API kullanarak yerel VCS bileşenleri oluşturma düzeltildi.

4.21 Weblate 4.2.1

21 Ağustos 2020 tarihinde yayınlandı.

- Android kaynaklarında bazı yerel ayarlar için çokulların kaydedilmesi düzeltildi.
- Bazı XLIFF dosyaları için temizleme eklentisindeki çokme düzeltildi.
- Docker kalıbında yerelleştirme CDN'si kurtarma izin ver.

4.22 Weblate 4.2

18 Ağustos 2020 tarihinde yayınlandı.

- Kullanıcı sayfaları iyileştirildi ve kullanıcı listesi eklendi.
- 3.x sürümlerinden geçiş desteği bırakıldı, 4.1 veya 4.0'dan geçiş yapın.
- Birkaç tek dilli biçime dışa aktarılmak için eklendi.
- Etkinlik çizelgeleri iyileştirildi.
- Görüntülenen yakındaki dizgilerin sayısı yapılandırılabilir.
- Depo hataları yaşayan bileşenleri kilitleme desteği eklendi.
- Ana gezinme basitleştirildi (diğerler simgelerle değiştirildi).
- Google Çeviri bütenleştirmesinde dil kodu işlenmesi iyileştirildi.
- Git toparlama eklentisi Co-authored-by: artbilgileri oluşturabilir.
- Araç sorgusu ayrıncılarını iyileştirildi.
- Biçim dizgileri denetimlerinden kullanıcı geri bildirimini iyileştirdi.
- Toplu durum değişikliklerinin performansı iyileştirildi.
- Proje veya bileşen yeniden adlandırılduktan sonra uyumluluk yeniden yönlendirmeleri eklendi.
- Dizgi onayı, bileşen kilitleme ve lisans değişikliği için bildirimler eklendi.
- ModernMT için destek eklendi.
- Dosya yüklemede onaylanan çevirilerin üzerine yazmaktan kaçınmak için izin ver.
- Bazı uyumluluk URL yönlendirmeleri için destek bırakıldı.
- ECMAScript şablon değişmezleri denetimi eklendi.
- Bir bileşeni izleme seçeneği eklendi.
• JSON birim anahtarlarının başındaki nokta kaldırdı.
• Çeviri belleği için ayrı Celery kuyruğu kaldırdı.
• Tüm bileşenlerin bir dili bir kerede çevirmesine izin ver.
• Content-Security-Policy HTTP başlıklarını yapılandırmaya izin ver.
• Proje düzeyinde dillere kod adı vermek için destek eklendi.
• HTML veya JavaScript yerelleştirmesine yardımcı olmak için yeni eklenti, bkz: JavaScript yerelleştirme CDN.
• Weblate etki alanı artık ayarlardayı yapılandırdı, bkz: SITE_DOMAIN.
• Bileşene ve projeye göre arama desteği ekle.

4.23 Weblate 4.1.1

Released on June 19th 2020.
• Fixed changing autofix or add-ons configuration in Docker.
• Fixed possible crash in “About” page.
• Improved installation of byte-compiled locale files.
• Fixed adding words to glossary.
• Fixed keyboard shortcuts for machinery.
• Removed debugging output causing discarding log events in some setups.
• Fixed lock indication on project listing.
• Fixed listing GPG keys in some setups.
• Added option for which DeepL API version to use.
• Added support for acting as SAML Service Provider, see SAML authentication.

4.24 Weblate 4.1

Released on June 15th 2020.
• Added support for creating new translations with included country code.
• Added support for searching source strings with screenshot.
• Extended info available in the stats insights.
• Improved search editing on “Translate” pages.
• Improve handling of concurrent repository updates.
• Include source language in project creation form.
• Include changes count in credits.
• Fixed UI language selection in some cases.
• Allow to whitelist registration methods with registrations closed.
• Improved lookup of related terms in glossary.
• Improved translation memory matches.
• Group same machinery results.
• Add direct link to edit screenshot from translate page.
• Improved removal confirmation dialog.
• Include templates in ZIP download.
• Add support for Markdown and notification configuration in announcements.
• Extended details in check listings.
• Consistently use dismissed as state of dismissed checks.
• Add support for configuring default add-ons to enable.
• Fixed editor keyboard shortcut to dismiss checks.
• Improved machine translation of strings with placeholders.
• Show ghost translation for user languages to ease starting them.
• Improved language code parsing.
• Show translations in user language first in the list.
• Renamed shapings to more generic name variants.
• Added new quality checks: Birden çok adımsız değişken, Uzun süre çevrilmeyen, Art arda yinelenen sözcükler.
• 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 Customizing behavior using flags.
• Always show rendered text check if enabled.
• API now supports filtering of changes.
• Added support for sharing glossaries between projects.

4.25 Weblate 4.0.4

7 Mayıs 2020 tarihinde yaymlandi.

• 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.
4.26 Weblate 4.0.3

2 Mayıs 2020 tarihinde yayınlandı.

- 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.

4.27 Weblate 4.0.2

Released on April 27th 2020.

- Improved performance of translation stats.
- Improved performance of changing labels.
- Toplul düzenleme performansı iyileştirildi.
- 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.

4.28 Weblate 4.0.1

Released on April 16th 2020.

- Fixed package installation from PyPI.
4.29 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 i18n next interpolation and nested.
- Added check and highlighter for percent placeholders.
- Önerilerin kusur denetimlerini göster.
- 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 Değiştilmiş çeviri avoid internal blacklist.
- Improved comments extraction from monolingual po files.
- Renamed whiteboard messages to announcements.
- Fixed occasional problems with registration mails.
- Improved LINGUAS update add-on to handle more syntax variants.
- Fixed editing monolingual XLIFF source file.
- Added support for exact matching in Searching.
- Extended API to cover screenshots, users, groups, component lists 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.

4.30 Weblate 3.x series

4.30.1 Weblate 3.11.3

Released on March 11th 2020.

- Fixed searching for fields with certain priority.
- Fixed predefined query for recently added strings.
- Fixed searching returning duplicate matches.
- Fixed notifications rendering in Gmail.
- Fixed reverting changes from the history.
- Added links to events in digest notifications.
- Fixed email for account removal confirmation.
• Added support for Slack authentication in Docker container.
• Avoid sending notifications for not subscribed languages.
• Include Celery queues in performance overview.
• Fixed documentation links for add-ons.
• Reduced false negatives for unchanged translation check.
• Raised bleach dependency to address CVE-2020-6802.
• Fixed listing project level changes in history.
• Fixed stats invalidation in some corner cases.
• Fixed searching for certain string states.
• Improved format string checks behavior on missing percent.
• Fixed authentication using some third party providers.

4.30.2 Weblate 3.11.2

Released on February 22nd 2020.
• Fixed rendering of suggestions.
• Fixed some strings wrongly reported as having no words.

4.30.3 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.

4.30.4 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.
• Dizgi şekillerini gruplama desteği eklendi.
• Recommend upgrade to new Weblate versions in the system checks.
• Provide more detailed analysis for duplicate language alert.
• Include more detailed license info on the project pages.
• Automatically unshallow local copies if needed.
• Fixed download of strings needing action.
• New alert to warn about using the same filemask twice.
• Improve XML placeables extraction.
• The `SINGLE_PROJECT` can now enforce redirection to chosen project.
• Added option to resolve comments.
• Added bulk editing of flags.
• Added support for labels.
• Added bulk edit add-on.
• Added option for `Enforcing checks`.
• Increased default validity of confirmation links.
• Improved Matomo integration.
• Fixed Çevrilmiş to correctly handle source string change.
• Extended automatic updates configuration by `AUTO_UPDATE`.
• LINGUAS add-ons now do full sync of translations in Weblate.

4.30.5 Weblate 3.10.3

Released on January 18th 2020.

• Support for translate-toolkit 2.5.0.

4.30.6 Weblate 3.10.2

Released on January 18th 2020.

• Add lock indication to projects.
• Fixed CSS bug causing flickering in some web browsers.
• Fixed searching on systems with non-English locales.
• Improved repository matching for GitHub and Bitbucket hooks.
• Fixed data migration on some Python 2.7 installations.
• Allow configuration of Git shallow cloning.
• Improved background notification processing.
• Fixed broken form submission when navigating back in web browser.
• New add-on to configure YAML formatting.
• Fixed same plurals check to not fire on single plural form languages.
• Fixed regex search on some fields.
4.30.7 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.
- Veri temizleme performansı iyileşirildi.
- 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.

4.30.8 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.
4.30.9 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.

4.30.10 Weblate 3.9

Released on October 15th 2019.

- Include Weblate metadata in downloaded files.
- Improved UI for failing checks.
- Indicate missing strings in format checks.
- Separate check for French punctuation spacing.
- Add support for fixing some of quality checks errors.
- Add separate permission to create new projects.
- Extend stats for char counts.
- Improve support for Java style language codes.
- Added new generic check for placeholders.
- Added support for WebExtension JSON placeholders.
- Added support for flat XML format.
- Extended API with project, component and translation removal and creation.
- Added support for Gitea and Gitee webhooks.
- Added new custom regex based check.
- Allow to configure contributing to shared translation memory.
- Added ZIP download for more translation files.
- Make XLIFF standard compliant parsing of maxwidth and font.
- Added new check and fixer for safe HTML markup for translating web applications.
- Add component alert on unsupported configuration.
- Added automatic translation add-on to bootstrap translations.
- Extend automatic translation to add suggestions.
- Display add-on parameters on overview.
• Sentry is now supported through modern Sentry SDK instead of Raven.
• Changed example settings to be better fit for production environment.
• Added automated backups using BorgBackup.
• Split cleanup add-on for RESX to avoid unwanted file updates.
• Added advanced search capabilities.
• Allow users to download their own reports.
• Added localization guide to help configuring components.
• Added support for GitLab merge requests.
• Improved display of repository status.
• Perform automated translation in the background.

4.30.11 Weblate 3.8

Released on August 15th 2019.

• Added support for simplified creating of similar components.
• Added support for parsing translation flags from the XML based file formats.
• Log exceptions into Celery log.
• Improve performance of repository scoped add-ons.
• Improved look of notification e-mails.
• Fixed password reset behavior.
• Improved performance on most of translation pages.
• Fixed listing of languages not known to Weblate.
• Add support for cloning add-ons to discovered components.
• Add support for replacing file content with uploaded.
• Add support for translating non VCS based content.
• Added OpenGraph widget image to use on social networks.
• Added support for animated screenshots.
• Improved handling of monolingual XLIFF files.
• Avoid sending multiple notifications for single event.
• Add support for filtering changes.
• Extended predefined periods for reporting.
• Added webhook support for Azure Repos.
• New opt-in notifications on pending suggestions or untranslated strings.
• Add one click unsubscribe link to notification e-mails.
• Fixed false positives with Has been translated check.
• New management interface for admins.
• String priority can now be specified using flags.
• Added language management views.
• Add checks for Qt library and Ruby format strings.
• Added configuration to better fit single project installations.
• Notify about new string on source string change on monolingual translations.
• Added separate view for translation memory with search capability.

4.30.12 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.

4.30.13 Weblate 3.7

Released on June 21st 2019.

• Added separate Celery queue for notifications.
• Use consistent look with application for API browsing.
• Include approved stats in the reports.
• Report progress when updating translation component.
• Allow to abort running background component update.
• Extend template language for filename manipulations.
• Use templates for editor link and repository browser URL.
• Indicate max length and current characters count when editing translation.
• Improved handling of abbreviations in unchanged translation check.
• Refreshed landing page for new contributors.
• Add support for configuring msgmerge add-on.
• Delay opening SMTP connection when sending notifications.
• Improved error logging.
• Allow custom location in MO generating add-on.
• Added add-ons to cleanup old suggestions or comments.
• Added option to enable horizontal mode in the Zen editor.
• Improved import performance with many linked components.
• Fixed examples installation in some cases.
• Improved rendering of alerts in changes.
• Added new horizontal stats widget.
• Improved format strings check on plurals.
• Added font management tool.
• New check for rendered text dimensions.
- Added support for subtitle formats.
- Include overall completion stats for languages.
- Added reporting at project and global scope.
- Improved user interface when showing translation status.
- New Weblate logo and color scheme.
- New look of bitmap badges.

### 4.30.14 Weblate 3.6.1

Released on April 26th 2019.
- Improved handling of monolingual XLIFF files.
- Fixed digest notifications in some corner cases.
- Fixed add-on script error alert.
- Fixed generating MO file for monolingual PO files.
- Fixed display of uninstalled checks.
- Indicate administered projects on project listing.
- Allow update to recover from missing VCS repository.

### 4.30.15 Weblate 3.6

Released on April 20th 2019.
- Add support for downloading user data.
- Add-ons are now automatically triggered upon installation.
- Improved instructions for resolving merge conflicts.
- Cleanup add-on is now compatible with app store metadata translations.
- Configurable language code syntax when adding new translations.
- Warn about using Python 2 with planned termination of support in April 2020.
- Extract special characters from the source string for visual keyboard.
- Extended contributor stats to reflect both source and target counts.
- Admins and consistency add-ons can now add translations even if disabled for users.
- Fixed description of toggle disabling Language-Team header manipulation.
- Notify users mentioned in comments.
- Removed file format autodetection from component setup.
- Fixed generating MO file for monolingual PO files.
- Added digest notifications.
- Added support for muting component notifications.
- Added notifications for new alerts, whiteboard messages or components.
- Notifications for administered projects can now be configured.
- Improved handling of three letter language codes.
4.30.16 Weblate 3.5.1

Released on March 10th 2019.

- Fixed Celery systemd unit example.
- Fixed notifications from HTTP repositories with login.
- Fixed race condition in editing source string for monolingual translations.
- Include output of failed add-on execution in the logs.
- Improved validation of choices for adding new language.
- Allow to edit file format in component settings.
- Update installation instructions to prefer Python 3.
- Performance and consistency improvements for loading translations.
- Make Microsoft Terminology service compatible with current Zeep releases.
- Localization updates.

4.30.17 Weblate 3.5

Released on March 3rd 2019.

- Improved performance of built-in translation memory.
- Added interface to manage global translation memory.
- Improved alerting on bad component state.
- Added user interface to manage whiteboard messages.
- Add-on commit message now can be configured.
- Reduce number of commits when updating upstream repository.
- Fixed possible metadata loss when moving component between projects.
- Improved navigation in the Zen mode.
- Added several new quality checks (Markdown related and URL).
- Added support for app store metadata files.
- Added support for toggling GitHub or Gerrit integration.
- Added check for Kashida letters.
- Added option to squash commits based on authors.
- Improved support for XLSX file format.
- Compatibility with Tesseract 4.0.
- Billing add-on now removes projects for unpaid billings after 45 days.
4.30.18  Weblate 3.4

Released on January 22nd 2019.

• Added support for XLIFF placeholders.
• Celery can now utilize multiple task queues.
• Added support for renaming and moving projects and components.
• Include characters counts in reports.
• Added guided adding of translation components with automatic detection of translation files.
• Customizable merge commit messages for Git.
• Added visual indication of component alerts in navigation.
• Improved performance of loading translation files.
• New add-on to squash commits prior to push.
• Improved displaying of translation changes.
• Changed default merge style to rebase and made that configurable.
• Better handle private use subtags in language code.
• Improved performance of fulltext index updates.
• Extended file upload API to support more parameters.

4.30.19  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 rename as context when available.
• Added support for XLIFF states.
• Added check for non writable files in DATA_DIR.
• Improved CSV export for changes.

4.30.20  Weblate 3.2.2

Released on October 20th 2018.

• Remove no longer needed Babel dependency.
• Updated language definitions.
• Improve documentation for add-ons, LDAP and Celery.
• Fixed enabling new dos-coll 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.
4.30.21 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.

4.30.22 Weblate 3.2

Released on October 6th 2018.

- Add install_addon management command for automated add-on installation.
- Allow more fine grained ratelimit settings.
- Added support for export and import of Excel files.
- Improve component cleanup in case of multiple component discovery add-ons.
- Rewritten Microsoft Terminology machine translation backend.
- Weblate now uses Celery to offload some processing.
- Improved search capabilities and added regular expression search.
- Added support for Youdao Zhiyun API machine translation.
- Added support for Baidu API machine translation.
- Integrated maintenance and cleanup tasks using Celery.
- Improved performance of loading translations by almost 25%.
- Removed support for merging headers on upload.
- Removed support for custom commit messages.
- Configurable editing mode (zen/full).
- Added support for error reporting to Sentry.
- Added support for automated daily update of repositories.
- Added support for creating projects and components by users.
- Built-in translation memory now automatically stores translations done.
- Users and projects can import their existing translation memories.
- Better management of related strings for screenshots.
- Added support for checking Java MessageFormat.

See 3.2 milestone on GitHub for detailed list of addressed issues.
4.30.23  Weblate 3.1.1

Released on July 27th 2018.

- Fix testsuite failure on some setups.

4.30.24  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 libavatar service.
- Değiştirilmemiş çevirileri “Düzenleme gerekli” olarak işaretlemek için yeni eklendi.
- 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.

4.30.25  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.

4.30.26  Weblate 3.0

Released on June 1st 2018.

- Rewritten access control.
- Several code cleanups that lead to moved and renamed modules.
- New add-on for automatic component discovery.
- The import_project management command has now slightly different parameters.
- Added basic support for Windows RC files.
- New add-on to store contributor names in PO file headers.
- The per component hook scripts are removed, use add-ons instead.
- Add support for collecting contributor agreements.
• Access control changes are now tracked in history.
• New add-on to ensure all components in a project have same translations.
• Support for more variables in commit message templates.
• Add support for providing additional textual context.

4.31 Weblate 2.x series

4.31.1 Weblate 2.20

Released on April 4th 2018.
• Improved speed of cloning subversion repositories.
• Changed repository locking to use third party library.
• Added support for downloading only strings needing action.
• Added support for searching in several languages at once.
• New add-on to configure gettext output wrapping.
• New add-on to configure JSON formatting.
• Added support for authentication in API using RFC 6750 compatible Bearer authentication.
• Added support for automatic translation using machine translation services.
• Added support for HTML markup in whiteboard messages.
• Added support for mass changing state of strings.
• Translate-toolkit at least 2.3.0 is now required, older versions are no longer supported.
• Dahili çeviri belleği eklendi.
• Added component lists overview to dashboard and per component list overview pages.
• Added support for DeepL machine translation service.
• Machine translation results are now cached inside Weblate.
• İşlenmiş değişiklikleri yeniden sıralama desteği eklendi.

4.31.2 Weblate 2.19.1

Released on February 20th 2018.
• Fixed migration issue on upgrade from 2.18.
• Improved file upload API validation.

4.31.3 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.
• Dahili makine çevirisi hizmeti geliştirildi.
• Added support for SAP Translation Hub service.
• Added support for Microsoft Terminology service.
• Removed support for advertisement in notification e-mails.
• Improved translation progress reporting at language level.
• Improved support for different plural formulas.
• Added support for Subversion repositories not using stdlayout.
• Added add-ons to customize translation workflows.

4.31.4 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.

4.31.5 Weblate 2.17.1

Released on October 13th 2017.

• Fixed running testsuite in some specific situations.
• Locales updates.
4.31.6 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.

4.31.7 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.

4.31.8 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.

### 4.31.9 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.

### 4.31.10 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.
4.31.11 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.

4.31.12 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.

4.31.13 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.
4.31.14 Weblate 2.11

Released on Jan 31st 2017.

• Include language detailed information on language page.
• Mercurial backend improvements.
• Added option to specify translation component priority.
• More consistent usage of Group ACL even with less used permissions.
• Added WL_BRANCH variable to hook scripts.
• Improved developer documentation.
• Better compatibility with various Git versions in Git exporter add-on.
• Included per project and component stats.
• Added language code mapping for better support of Microsoft Translate API.
• Moved fulltext cleanup to background job to make translation removal faster.
• Fixed displaying of plural source for languages with single plural form.
• Improved error handling in import_project.
• Various performance improvements.

4.31.15 Weblate 2.10.1

Released on Jan 20th 2017.

• Do not leak account existence on password reset form (CVE-2017-5537).

4.31.16 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.
4.31.17  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.

4.31.18  Weblate 2.8

Released on Aug 31st 2016.

- Belgelendirme iyileştirmeleri.
  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.
4.31.19 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.

4.31.20 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.

4.31.21 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.
• Belgelendirme iyileştirmeleri.
• 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.
• Kusur denetimleri için site genelinde arama düzeltildi.
• Added option to specify source language.
• Improved support for XLIFF files.
• Extended list of options for import_project.
• Improved targeting for whiteboard messages.
• Support for automatic translation across projects.
• Optimized fulltext search index.
• Added management command for auto translation.
• Added placeables highlighting.
• Added keyboard shortcuts for placeables, checks and machine translations.
• Improved translation locking.
• Added quality check for AngularJS interpolation.
• Added extensive group based ACLs.
• Clarified terminology on strings needing edit (formerly fuzzy).
• Clarified terminology on strings needing action and not translated strings.
• Support for Python 3.
• Dropped support for Django 1.7.
• Dropped dependency on msginit for creating new gettext PO files.
• Added configurable dashboard views.
• Improved notifications on parse errors.
• Added option to import components with duplicate name to import_project.
• Improved support for translating PHP files.
• Added XLIFF export for dictionary.
• Added XLIFF and gettext PO export for all translations.
• Belgelendirme iyileştirmeleri.
• Added support for configurable automatic group assignments.
• Improved adding of new translations.

4.31.22 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.

4.31.23 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.
4.31.24 Weblate 2.2

Released on Feb 19th 2015.

• Performans iyileştirmeleri.
• 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.

4.31.25 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.
• Belgelendirme iyileştirmeleri.
• 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.

4.31.26 Weblate 2.0

Released on Nov 6th 2014.

• New responsive UI using Bootstrap.
• Rewritten VCS backend.
• Belgelendirme iyileştirmeleri.
• 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.

4.32 Weblate 1.x series

4.32.1 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.

4.32.2 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.
• Belgelendirme iyileştirmeleri.
• Improved source strings review.
• Searching across all strings.
• Better tracking of source strings.
• Captcha protection for registration.

4.32.3 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.

4.32.4 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.
4.32.5 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.

4.32.6 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 tktit loader.
- Added sitemaps to allow easier access by crawlers.
- Provide direct links to string in notification e-mails or feeds.
- Various improvements to admin interface.
- Provide hints for production setup in admin interface.
- Added per language widgets and engage page.
- Improved translation locking handling.
- Show code snippets for widgets in more variants.
- Indicate failing checks or fuzzy strings in progress bars.
- More options for formatting commit message.
- Fixed error handling with machine translation services.
- Improved automatic translation locking behaviour.
• Support for showing changes from previous source string.
• Added support for substring search.
• Various quality checks improvements.
• Support for per project ACL.
• Basic code coverage by unit tests.

4.32.7 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.

4.32.8 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.
• Kusur denetimleri olan dizgileri sayısının ön belleğe alınması.
• Checking of requirements during setup.
• Belgelemdirme iyileştirileri.

4.32.9 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.

4.32.10 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.

4.33 Weblate 0.x series

4.33.1 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.
• Belgelendirme iyileştirmeleri.
• 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.

### 4.33.2 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.

### 4.33.3 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.
4.33.4 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.

4.33.5 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).

4.33.6 Weblate 0.4

Released on February 8th 2012.

- Added usage guide to documentation.
- Fixed API hooks not to require CSRF protection.

4.33.7 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.
4.33.8 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.

4.33.9 Weblate 0.1

Released on February 6th 2012.

• Initial release.
Python Modül Dizini

W
wlc, 137
wlc.config, 138
wlc.main, 138
<table>
<thead>
<tr>
<th>Route</th>
<th>Method</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANY /</td>
<td>GET</td>
<td>89</td>
</tr>
<tr>
<td>/api</td>
<td>GET</td>
<td>92</td>
</tr>
<tr>
<td>/api/addons</td>
<td>GET</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>PUT</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>PATCH</td>
<td>126</td>
</tr>
<tr>
<td>/api/addons/(int:id)</td>
<td>GET</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>PUT</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>PATCH</td>
<td>126</td>
</tr>
<tr>
<td>/api/changes</td>
<td>GET</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td>(int:id)</td>
<td>123</td>
</tr>
<tr>
<td>/api/component-lists</td>
<td>GET</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>POST</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>PUT</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>PATCH</td>
<td>128</td>
</tr>
<tr>
<td>/api/component-lists/(str:slug)</td>
<td>GET</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>POST</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>PUT</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>PATCH</td>
<td>128</td>
</tr>
<tr>
<td>/api/groups</td>
<td>GET</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>(int:id)</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>POST</td>
<td>97</td>
</tr>
<tr>
<td>/api/groups/(int:id)</td>
<td>componentlists/</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>components/</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>languages/</td>
<td>97</td>
</tr>
<tr>
<td>/api/groups/(int:id)/componentlists/</td>
<td></td>
<td>97</td>
</tr>
<tr>
<td>/api/groups/(int:id)/components/</td>
<td></td>
<td>109</td>
</tr>
<tr>
<td>/api/groups/(int:id)/languages/</td>
<td></td>
<td>109</td>
</tr>
<tr>
<td>/api/components</td>
<td>GET</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>(string:project)/components/</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>(string:project)/languages/</td>
<td>111</td>
</tr>
<tr>
<td></td>
<td>(string:project)/links/</td>
<td>115</td>
</tr>
</tbody>
</table>
POST /api/groups/(int:id)/projects/, 97
POST /api/groups/(int:id)/roles/, 97
PUT /api/groups/(int:id)/, 96
DELETE /api/groups/(int:id)/componentlists/(int:component_list_id), 98
DELETE /api/groups/(int:id)/components/(int:component_id), 97
DELETE /api/groups/(int:id)/languages/(string:language_code), 97
DELETE /api/groups/(int:id)/projects/(int:project_id), 97
PATCH /api/groups/(int:id)/, 96

/api/roles
GET /api/roles/, 98
GET /api/roles/(int:id)/, 98
POST /api/roles/, 98
PUT /api/roles/(int:id)/, 98
DELETE /api/roles/(int:id)/, 99
PATCH /api/roles/(int:id)/, 99

/api/screenshots
GET /api/screenshots/(int:id)/, 124
GET /api/screenshots/(int:id)/file/, 124
DELETE /api/screenshots/(int:id)/units/(int:unit_id), 125
PATCH /api/screenshots/(int:id)/, 125

/api/tasks
GET /api/tasks/, 128
GET /api/tasks/(str:uuid)/, 128

/api/translations
GET /api/translations/(string:project)/(string:component)/(string:language)/, 116
GET /api/translations/(string:project)/(string:component)/(string:language)/changes/, 118
GET /api/translations/(string:project)/(string:component)/(string:language)/file/, 119
GET /api/translations/(string:project)/(string:component)/(string:language)/repository/, 120
GET /api/translations/(string:project)/(string:component)/(string:language)/statistics/, 120
GET /api/translations/(string:project)/(string:component)/(string:language)/units/, 118
POST /api/translations/(string:project)/(string:component)/(string:language)/autotranslate/, 119
POST /api/translations/(string:project)/(string:component)/(string:language)/file/, 119
POST /api/translations/(string:project)/(string:component)/(string:language)/repository/, 120
POST /api/translations/(string:project)/(string:component)/(string:language)/units/, 118
DELETE /api/translations/(string:project)/(string:component)/(string:language)/, 118

/api/units
GET /api/units/, 121
GET /api/units/(int:id)/, 121
PUT /api/units/(int:id)/, 122
DELETE /api/units/(int:id)/, 123
PATCH /api/units/(int:id)/, 122

/api/users
GET /api/users/, 92
GET /api/users/(str:username)/, 92
GET /api/users/(str:username)/notifications/, 94
GET /api/users/(str:username)/notifications/(int:subscription_id)/, 94
GET /api/users/(str:username)/statistics/, 94
POST /api/users/, 92
POST /api/users/(str:username)/groups/, 94
POST /api/users/(str:username)/notifications/, 94
PUT /api/users/(str:username)/, 93
PUT /api/users/(str:username)/notifications/(int:subscription_id)/, 94
DELETE /api/users/(str:username)/, 94
DELETE /api/users/(str:username)/notifications/(int:subscription_id)/, 95
PATCH /api/users/(str:username)/, 93
PATCH /api/users/(str:username)/notifications/(int:subscription_id)/, 95

/exports
GET /exports/rss/, 132
GET /exports/rss/(string:project)/, 132
GET /exports/rss/(string:project)/(string:component)/, 132
GET /exports/rss/(string:project)/(string:component)/(string:language)/, 132
GET /exports/rss/language/(string:language)/, 132
GET /exports/stats/(string:project)/(string:component)/, 131

/hooks
GET /hooks/update/(string:project)/, 129
GET /hooks/update/(string:project)/(string:component)/, 129
POST /hooks/azure/, 130
POST /hooks/bitbucket/, 130
POST /hooks/gitea/, 130
POST /hooks/gitee/, 131
POST /hooks/github/, 129
POST /hooks/gitlab/, 130
POST /hooks/pagure/, 130
**Semboller**

- **.XML resource file file format**, 75
- **--add**
  - auto_translate komut satırı seçeneği, 347
- **--addon ADDON**
  - install_addon komut satırı seçeneği, 353
- **--age HOURS**
  - commit_pending komut satırı seçeneği, 348
- **--author USER@EXAMPLE.COM**
  - add_suggestions komut satırı seçeneği, 347
- **--base-file-template TEMPLATE**
  - import_project komut satırı seçeneği, 351
- **--check**
  - importusers komut satırı seçeneği, 353
- **--config PATH**
  - wlc komut satırı seçeneği, 134
- **--config-section SECTION**
  - wlc komut satırı seçeneği, 134
- **--configuration CONFIG**
  - install_addon komut satırı seçeneği, 353
- **--convert**
  - wlc komut satırı seçeneği, 135
- **--email USER@EXAMPLE.COM**
  - createadmin komut satırı seçeneği, 349
- **--file-format FORMAT**
  - import_project komut satırı seçeneği, 351
- **--force**
  - loadpo komut satırı seçeneği, 354
- **--force-commit**
  - pushgit komut satırı seçeneği, 355
- **--format {csv, json, text, html]**
  - wlc komut satırı seçeneği, 134
- **--ignore**
  - import_json komut satırı seçeneği, 350
- **--inconsistent**
  - auto_translate komut satırı seçeneği, 347
- **--input**
  - wlc komut satırı seçeneği, 135
- **--key KEY**
  - wlc komut satırı seçeneği, 134
- **--lang LANGUAGE**
  - loadpo komut satırı seçeneği, 354
- **--language-code**
  - list_translators komut satırı seçeneği, 354
- **--language-map LANGMAP**
  - import_memory komut satırı seçeneği, 350
- **--language-regex REGEX**
  - import_project komut satırı seçeneği, 351
- **--license NAME**
  - import_project komut satırı seçeneği, 351
- **--license-url URL**
  - import_project komut satırı seçeneği, 351
- **--main-component**
  - import_project komut satırı seçeneği, 351
- **--mode MODE**
  - import_json komut satırı seçeneği, 350
- **--mt MT**
  - auto_translate komut satırı seçeneği, 347
- **--name**
  - createadmin komut satırı seçeneği, 349
- **--name-template TEMPLATE**
  - import_project komut satırı seçeneği, 351
CELERY_TRANSLATE_OPTIONS, 144, 158
changes
  wlc komut satırı seçeneği, 135
CHECK_LIST
  setting, 305
checkgit
  weblate admin command, 348
cleanup
  wlc komut satırı seçeneği, 135
cleanuptrans
  weblate admin command, 348
Comma separated values
  file format, 76
Command (wlc:main içindeki smf), 139
COMMENT_CLEANUP_DAYS
  setting, 305
commit
  wlc komut satırı seçeneği, 134
commit_pending
  weblate admin command, 348
commit_pending komut satırı seçeneği
  --age HOURS, 348
COMMIT_PENDING_HOURS
  setting, 306
commitgit
  weblate admin command, 348
configure()
  (weblate.addons.base.BaseAddon yöntemi), 390
CONTACT_FORM
  setting, 306
createadmin
  weblate admin command, 349
createadmin komut satırı seçeneği
  --email USER@EXAMPLE.COM, 349
  --name, 349
  --no-password, 349
  --password PASSWORD, 349
  --update, 349
  --username USERNAME, 349
CSP_CONNECT_SRC
  setting, 305
CSP_FONT_SRC
  setting, 305
CSP_IMG_SRC
  setting, 305
CSP_SCRIPT_SRC
  setting, 305
CSP_STYLE_SRC
  setting, 305
CSV
  file format, 76
D
  daily()
    (weblate.addons.base.BaseAddon yöntemi), 390
DATA_DIR
  setting, 306
DATABASE_BACKUP
  setting, 307
DATABASES
  setting, 182
DEBUG
  setting, 182
DEFAULT_ACCESS_CONTROL
  setting, 307
DEFAULT_ADD_MESSAGE
  setting, 307
DEFAULT_ADDON_MESSAGE
  setting, 307
DEFAULT_ADDONS
  setting, 308
DEFAULT_AUTO_WATCH
  setting, 307
DEFAULT_COMMIT_MESSAGE
  setting, 307
DEFAULT_COMMITER_EMAIL
  setting, 308
DEFAULT_COMMITER_NAME
  setting, 308
DEFAULT_DELETE_MESSAGE
  setting, 307
DEFAULT_FROM_EMAIL
  setting, 182
DEFAULT_LANGUAGE
  setting, 308
DEFAULT_MERGE_MESSAGE
  setting, 307
DEFAULT_MERGE_STYLE
  setting, 309
DEFAULT_PAGE_LIMIT
  setting, 322
DEFAULT_PULL_MESSAGE
  setting, 309
DEFAULT_RESTRICTED_COMPONENT
  setting, 307
DEFAULT_SHARED_TM
  setting, 309
DEFAULT_TRANSLATION_PROPAGATION
  setting, 309
download
  wlc komut satırı seçeneği, 135
DTD
  file format, 77
dump_memory
  weblate admin command, 349
dumpuserdata
  weblate admin command, 349
E
  ENABLE_AVATARS
    setting, 309
  ENABLE_HOOKS
    setting, 309
  ENABLE_HTTPS
    setting, 310
  ENABLE_SHARING
F

file format
 .XML resource file, 75
 Android, 70
 Apple strings, 71
 ARB, 74
 Comma separated values, 76
 CSV, 76
 DTD, 77
 gettext, 65
 go-i18n, 74
 GWT properties, 68
 i18next, 73
 INI translations, 69
 Java properties, 68
 Joomla translations, 69
 JSON, 72
 mi18n lang, 68
 PHP strings, 71
 PO, 65
 Qt, 70
 RC, 78
 RESX, 75
 Ruby YAML, 77
 Ruby YAML Ain't Markup Language, 77
 string resources, 70
 TS, 70
 XLIFF, 66
 XML, 78
 YAML, 76
 YAML Ain't Markup Language, 76

G

get() (wlc.Weblate yöntemi), 137
get_add_form() (weblate.addons.base.BaseAddon sınıf yöntemi), 390
GET_HELP_URL
 setting, 310
get_settings_form() (weblate.addons.base.BaseAddon yöntem), 390
ggettext
 file format, 65

H

HIDE_REPO_CREDENTIALS
 setting, 312
HIDE_VERSION
 setting, 312

I

i18next
 file format, 73
import_demo
 weblate admin command, 349
import_json
 weblate admin command, 350
import_json komut satırı seçeneği
 --ignore, 350
 --main-component COMPONENT, 350
 --project PROJECT, 350
 --update, 350
import_memory
 weblate admin command, 350
import_memory komut satırı seçeneği
 --language-map LANGMAP, 350
import_project
 weblate admin command, 351
import_project komut satırı seçeneği
 --base-file-template TEMPLATE, 351
 --file-format FORMAT, 351
 --language-regex REGEX, 351
 --license NAME, 351
 --license-url URL, 351
 --main-component, 351
 --name-template TEMPLATE, 351
 --new-base-template TEMPLATE, 351
 --vcs NAME, 351
importuserdata
 weblate admin command, 353
importusers
 weblate admin command, 353
importusers komut satırı seçeneği
 --check, 353
INI translations
 file format, 69
install_addon
 weblate admin command, 353
install_addon komut satırı seçeneği
 --addon ADDON, 353
 --configuration CONFIG, 353
 --update, 353
IP_BEHIND_REVERSE_PROXY
 setting, 312
IP_PROXY_HEADER
 setting, 312
IP_PROXY_OFFSET

Dizin 453
setting, 313
iPad translation, 71
iPhone translation, 71

J
Java properties file format, 68
Joomla translations file format, 69
JSON file format, 72

L
LEGAL_URL setting, 313
LICENSE_EXTRA setting, 313
LICENSE_FILTER setting, 314
LICENSE_REQUIRED setting, 314
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH setting, 314
list_languages weblate admin command, 353
list_translators weblate admin command, 354
list_translators komut satırı seçeneği --language-code, 354
list_versions weblate admin command, 354
list-components wlc komut satırı seçeneği, 134
list-languages wlc komut satırı seçeneği, 134
list-projects wlc komut satırı seçeneği, 134
list-translations wlc komut satırı seçeneği, 134
load() (wlc.config.WeblateConfig yöntemi), 138
loadpo weblate admin command, 354
loadpo komut satırı seçeneği --force, 354
--lang LANGUAGE, 354
LOCALIZE_CDN_PATH setting, 314
LOCALIZE_CDN_URL setting, 314
lock wlc komut satırı seçeneği, 135
lock_translation weblate admin command, 354
lock-status wlc komut satırı seçeneği, 135
LOGIN_REQUIRED_URLS setting, 315
LOGIN_REQUIRED_URLS_EXCEPTIONS setting, 315
ls wlc komut satırı seçeneği, 134

M
MACHINE_TRANSLATION_SERVICES setting, 316
main() (wlc.main modülü içinde), 138
MATOMO_SITE_ID setting, 315
MATOMO_URL setting, 316
mi18n lang module, 68
modülü wlc, 137
wlc.config, 138
wlc.main, 138
monolingual translation, 62
wlc komut satırı seçeneği sete_language weblate admin command, 355
MT_APERTIUM_API setting, 317
MT_AWS_ACCESS_KEY_ID setting, 317
MT_AWS_REGION setting, 317
MT_AWS_SECRET_ACCESS_KEY setting, 317
MT_BAIDU_ID setting, 317
MT_BAIDU_SECRET setting, 317
MT_DEEPL_API_URL setting, 318
MT_DEEPL_KEY setting, 318
MT_GOOGLE_CREDENTIALS setting, 319
MT_GOOGLE_KEY setting, 319
MT_GOOGLE_LOCATION setting, 319
MT_GOOGLE_PROJECT setting, 319
MT_LIBRETRANSLATE_API_URL setting, 318
MT_LIBRETRANSLATE_KEY setting, 318
MT_MICROSOFT_BASE_URL setting, 319
MT_MICROSOFT_COGNITIVE_KEY setting, 319
MT_MICROSOFT_ENDPOINT_URL setting, 319
setting, 320
MT_MICROSOFT_REGION
setting, 320
MT_MODERNMT_KEY
setting, 320
MT_MODERNMT_URL
setting, 320
MT_MYMEMORY_EMAIL
setting, 320
MT_MYMEMORY_KEY
setting, 320
MT_MYMEMORY_USER
setting, 320
MT_NETEASE_KEY
setting, 321
MT_NETEASE_SECRET
setting, 321
MT_SAP_BASE_URL
setting, 321
MT_SAP_PASSWORD
setting, 322
MT_SAP_SANDBOX_APIKEY
setting, 322
MT_SAP_USE_MT
setting, 322
MT_SAP_USERNAME
setting, 322
MT_TMSERVER
setting, 321
MT_YANDEX_KEY
setting, 321
MT_YOUDAO_ID
setting, 321
MT_YOUDAO_SECRET
setting, 321

N
NEARBY_MESSAGES
setting, 322

O
ortam değişkeni
CELERY_BACKUP_OPTIONS, 144, 158
CELERY_BEAT_OPTIONS, 144, 158, 159
CELERY_MAIN_OPTIONS, 144, 158
CELERY_MEMORY_OPTIONS, 144, 158
CELERY_NOTIFY_OPTIONS, 144, 158
CELERY_TRANSLATE_OPTIONS, 144, 158
POSTGRES.Alter_ROLE, 155
POSTGRES_CONN_MAX_AGE, 155
POSTGRES_DATABASE, 154
POSTGRES_HOST, 154
POSTGRES_PASSWORD, 154
POSTGRES_PASSWORD_FILE, 154
POSTGRES_PORT, 154
POSTGRES_SSL_MODE, 154
POSTGRES_USER, 154
REDIS_DB, 155
REDIS_HOST, 155
REDIS_PASSWORD, 155
REDIS_PORT, 155
REDIS_TLS, 155
REDIS_VERIFY_SSL, 155
ROLLBAR_ENVIRONMENT, 157
ROLLBAR_KEY, 157
SENTRY_DSN, 157
SENTRY_ENVIRONMENT, 157
SOCIAL_AUTH_SLACK_SECRET, 154
UWSGI_WORKERS, 144, 158, 159
WEBLATE_ADD_ADDONS, 158
WEBLATE_ADD_APPS, 158
WEBLATE_ADD_AUTOFIX, 158
WEBLATE_ADD_CHECK, 158
WEBLATE_ADD_LOGIN_REQUIRED_URLS_EXCEPTIONS,
148
WEBLATE_ADMIN_EMAIL, 144--146, 150
WEBLATE_ADMIN_NAME, 144--146
WEBLATE_ADMIN_PASSWORD, 141, 144--146
WEBLATE_ADMIN_PASSWORD_FILE, 146
WEBLATE_AKISMET_API_KEY, 149, 363
WEBLATE_ALLOWED_HOSTS, 146, 182, 186,
187, 327
WEBLATE_AUTH_LDAP_BIND_DN, 151
WEBLATE_AUTH_LDAP_BIND_PASSWORD, 151
WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS,
151
WEBLATE_AUTH_LDAP_SERVER_URI, 151
WEBLATE_AUTH_LDAP_USER_ATTR_MAP, 151
WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE, 151
WEBLATE_AUTH_LDAP_USER_SEARCH, 151
WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER, 151
WEBLATE_AUTH_LDAP_USER_SEARCH_UNION, 151
WEBLATE_AUTH_LDAP_USER_SEARCH_UNION_DELIMITER, 151
WEBLATE_BASIC_LANGUAGES, 149
WEBLATE_CONTACT_FORM, 146
WEBLATE_CSP_CONNECT_SRC, 149
WEBLATE_CSP_FONT_SRC, 149
WEBLATE_CSP_IMG_SRC, 149
WEBLATE_CSP_SCRIPT_SRC, 149
WEBLATE_CSP_STYLE_SRC, 149
WEBLATE_DATABASE_BACKUP, 155
WEBLATE_DEBUG, 145
WEBLATE_DEFAULT_ACCESS_CONTROL, 148
WEBLATE_DEFAULT_AUTO_WATCH, 149
WEBLATE_DEFAULT_COMMITER_EMAIL, 148
WEBLATE_DEFAULT_COMMITER_NAME, 148
WEBLATE_DEFAULT_FROM_EMAIL, 146

Dizin

455
reset
wlc komut satırı seçeneği, 135
REST, 89
RESX
file format, 75
RFC
RFC 5646, 62
Ruby YAML
file format, 77
Ruby YAML Ain't Markup Language
file format, 77

S
save_state() (weblate.addons.base.BaseAddon yöntem), 390
SECRET_KEY
setting, 182
SENTRY_DSN
setting, 326
SERVER_EMAIL
setting, 183
SESSION_COOKIE_AGE_AUTHENTICATED
setting, 326
SESSION_ENGINE
setting, 182
setting
ADMIN, 182
AKISMET_API_KEY, 301
ALLOWED_HOSTS, 182
ANONYMOUS_USER_NAME, 302
AUDITLOG_EXPIRY, 302
AUTH_LOCK_ATTEMPTS, 302
AUTH_TOKEN_VALID, 303
AUTO_UPDATE, 302
AUTOFIX_LIST, 303
AVATAR_URL_PREFIX, 303
BACKGROUND_TASKS, 304
BASE_DIR, 304
BASIC_LANGUAGES, 304
CHECKLIST, 305
COMMENT_CLEANUP_DAYS, 305
COMMIT_PENDING_HOURS, 306
CONTACT_FORM, 306
CSP_CONNECT_SRC, 305
CSP_FONT_SRC, 305
CSP_IMG_SRC, 305
CSP_SCRIPT_SRC, 305
CSP_STYLE_SRC, 305
DATA_DIR, 306
DATABASE_BACKUP, 307
DATABASES, 182
DEBUG, 182
DEFAULT_ACCESS_CONTROL, 307
DEFAULT_ADD_MESSAGE, 307
DEFAULT_ADDON_MESSAGE, 307
DEFAULT_ADDONS, 308
DEFAULT_AUTO_WATCH, 307
DEFAULT_COMMIT_MESSAGE, 307
DEFAULT_COMMITTER_EMAIL, 308
DEFAULT_COMMITTER_NAME, 308
DEFAULT_DELETE_MESSAGE, 307
DEFAULT_FROM_EMAIL, 182
DEFAULT_LANGUAGE, 308
DEFAULT_MERGE_MESSAGE, 307
DEFAULT_MERGE_STYLE, 309
DEFAULT_PAGE_LIMIT, 322
DEFAULT_PULL_MESSAGE, 309
DEFAULT_RESTRICTED_COMPONENT, 307
DEFAULT_SHARED_TM, 309
DEFAULT_TRANSLATION_PROPAGATION, 309
ENABLE_AVATARS, 309
ENABLE_HOOKS, 309
ENABLE_HTTPS, 310
ENABLE_SHARING, 310
GITHUB_CREDENTIALS, 311
GITHUB_TOKEN, 311
GITHUB_USERNAME, 311
GITLAB_CREDENTIALS, 310
GITLAB_TOKEN, 311
GITLAB_USERNAME, 310
GOOGLE_ANALYTICS_ID, 311
HIDE_REPO_CREDENTIALS, 312
HIDE_VERSION, 312
IP_BEHIND_REVERSE_PROXY, 312
IP_PROXY_HEADER, 312
IP_PROXY_OFFSET, 313
LEGAL_URL, 313
LICENSE_EXTRA, 313
LICENSE_FILTER, 314
LICENSE_REQUIRED, 314
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH, 314
LOCALIZE_CDN_PATH, 314
LOCALIZE_CDN_URL, 314
LOGIN_REQUIRED_URLS, 315
LOGIN_REQUIRED_URLS_EXCEPTIONS, 315
MACHINE_TRANSLATION_SERVICES, 316
MATOMO_SITE_ID, 315
MATOMO_URL, 316
MT_APERTIUM_API, 317
MT_AWS_ACCESS_KEY_ID, 317
MT_AWS_REGION, 317
MT_AWS_SECRET_ACCESS_KEY, 317
MT_BAIDU_ID, 317
MT_BAIDU_SECRET, 317
MT_DEEPL_API_URL, 318
MT_DEEPL_KEY, 318
MT_GOOGLE_CREDENTIALS, 319
MT_GOOGLE_KEY, 319
MT_GOOGLE_LOCATION, 319
MT_GOOGLE_PROJECT, 319
MT_LIBRETRANSLATE_API_URL, 318
MT_LIBRETRANSLATE_KEY, 318
MT_MICROSOFT_BASE_URL, 319
setupgroups
weblate admin command, 356
setupgroups komut satırı seçeneği
--no-privs-update, 356
--no-projects-update, 356
setuplang
weblate admin command, 356
setuplang komut satırı seçeneği
--no-update, 356
show
wlc komut satırı seçeneği, 134
SIMPLOYEE_LANGUAGES
setting, 326
SINGLE_PROJECT
setting, 327
SITE_DOMAIN
setting, 327
SITE_TITLE
setting, 327
SPECIAL_CHARS
setting, 327
statistics
wlc komut satırı seçeneği, 135
STATUS_URL
setting, 327
stay_on_create (weblate.addons.base.BaseAddon
özneliği), 390
store_post_load() (webla-
te.addons.base.BaseAddon yöntemi), 391
string resources
file format, 70
SUGGESTION_CLEANUP_DAYS
setting, 327

T
translation
bilingual, 62
iPad, 71
iPhone, 71
monolingual, 62
TS
file format, 70

U
unit_pre_create() (webla-
te.addons.base.BaseAddon yöntemi), 391
unlock
wlc komut satırı seçeneği, 135
unlock_translation
weblate admin command, 355
UPDATE_LANGUAGES
setting, 327
updatechecks
weblate admin command, 356
updategit
weblate admin command, 356
upload
wlc komut satırı seçeneği, 135
YAML
  file format.76
YAML Ain't Markup Language
  file format.76