



The Weblate Manual

4.3.1

Michal Čihař

2020-10-21

???? ?

Weblate

?

Weblate [GNU gettext](#) [Android string resources](#) 1

Component configuration

?

??

?

?

1. ?
2. ?
3. ?

?

?

2.5

?

???: ?

Your profile

- Languages Preferences Notifications Account Profile Licenses Audit log API access

Preferences form with sections: Hide completed translations, Translation editor mode (Full editor), Zen editor mode (Top to bottom), Number of nearby strings (15), Show secondary translations, Editor link, Special characters, Default dashboard view (Watched translations), Default component list.

```
#####:
## ##### > ##### Weblate #####
## ##### ## #####
##### #####
##### Weblate #####
##### #####
##### #####
```

```
##: settings.py ##### SINGLE_PROJECT ##### Weblate
#####
```


?? ??

?? [Component configuration](#) ?? Web ??

?: [?? ?](#) [??](#) [??](#) VCS [??](#) [??](#) [Template markup](#)

?? editor://open/?file={{filename}}&line={{line}} ??

?:

?? URL [Nette documentation](#) ??

??

??

??

?? 2 ??

??

??

?:

Your profile

- [Languages](#)
- [Preferences](#)
- Notifications
- [Account](#)
- [Profile](#)
- [Licenses](#)
- [Audit log](#)
- [API access](#)

Watched projects

Watched projects

Search...

<p>Available:</p> <ul style="list-style-type: none"> <li style="margin-bottom: 5px;">WeblateOrg 	<p>Chosen:</p> <ul style="list-style-type: none"> <li style="margin-bottom: 5px;">WeblateOrg
---	--

You can receive notifications for watched projects and they are shown on the dashboard by default.
 Add all projects you want to translate to see them as watched projects on the dashboard.

Save

Notification settings

Watched projects
Managed projects

Component wide notifications

You will receive a notification for every such event in your watched projects.

Repository failure	<input type="text" value="Do not notify"/>
Repository operation	<input type="text" value="Do not notify"/>
Component locking	<input type="text" value="Do not notify"/>
Changed license	<input type="text" value="Do not notify"/>
Parse error	<input type="text" value="Do not notify"/>
Comment on own translation	<input type="text" value="Instant notification"/>
Mentioned in comment	<input type="text" value="Instant notification"/>
New language	<input type="text" value="Do not notify"/>
New translation component	<input type="text" value="Do not notify"/>
New announcement	<input type="text" value="Instant notification"/>
New alert	<input type="text" value="Do not notify"/>

Translation notifications

You will only receive these notifications for your translated languages in your watched projects.

New string	<input type="text" value="Do not notify"/>
New contributor	<input type="text" value="Do not notify"/>
New suggestion	<input type="text" value="Do not notify"/>
New comment	<input type="text" value="Do not notify"/>
Changed string	<input type="text" value="Do not notify"/>
Translated string	<input type="text" value="Do not notify"/>
Approved string	<input type="text" value="Do not notify"/>
Pending suggestions	<input type="text" value="Do not notify"/>
Strings needing action	<input type="text" value="Do not notify"/>

Save



[Weblate](#) [GitLab](#) [GitHub](#) [Google](#) [Facebook](#) [Bitbucket](#)
[Weblate](#) [OAuth 2.0](#)

??: [Weblate](#) [GitLab](#) [GitHub](#) [Google](#) [Facebook](#) [Bitbucket](#)
[OAuth 2.0](#)

 Your profile

- [Languages](#)
- [Preferences](#)
- [Notifications](#)
- Account
- [Profile](#)
- [Licenses](#)
- [Audit log](#)
- [API access](#)

Account ⓘ

Username

Username may only contain letters, numbers or the following characters: @ . + - _

Full name

E-mail

You can add another e-mail address below.

Your name and e-mail will appear as commit authorship.

Save

Current user identities ⓘ

Identity	User ID	Action
 Password	testuser	Change password
 E-mail	weblate@example.org	Disconnect
 Google	weblate@example.org	Disconnect
 GitHub	123456	Disconnect
 Bitbucket	weblate	Disconnect

Add new association



E-mail

Removal

Account removal deletes all your private data.

Remove my account

User data

You can download all your private data.

Download user data

????

IP

??:

Running behind reverse proxy

Weblate

Weblate

2

:

??:

??????

The screenshot shows the Weblate web interface. At the top, there is a navigation bar with 'Weblate', 'Dashboard', 'Projects', 'Languages', and 'Checks'. Below this is a header for 'WeblateOrg' with a 'translated 85%' indicator. A menu bar contains 'Components', 'Languages', 'Info', 'Search', 'Glossaries', 'Insights', 'Files', 'Tools', 'Manage', 'Share', and 'Not watching'. The main content area is a table with columns: Component, Translated, Untranslated, Untranslated words, Checks, Suggestions, and Comments. Two components are listed: 'Android' (79% translated, 30 untranslated words, 3 checks) and 'Language names' (95% translated, 4 untranslated words, 5 checks). At the bottom, there is a button 'Add new translation component' and a footer with 'Powered by Weblate 4.3' and various links.

Component	Translated	Untranslated	Untranslated words	Checks	Suggestions	Comments
Android	79%	30	30	3		
Language names	95%	4	5			

??????

?????

AA

??:

String variants

???

AA

??:

String labels

??

AA

AA

AA

2 AAA

AA

???

AA

1

AA 1 AAA

Webate AAA

AA Unicode AAA Language Plural Rules AAA

??:

??????

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 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

Glossary

English Czech

No related strings found in the glossary.

+ Add term to glossary

Source information

Screenshot context
No screenshot currently associated.

Explanation
No explanation currently provided.

Labels
No labels currently set.

Flags
python-format

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

String age
6 seconds ago

Source string age
6 seconds ago

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

????? ????????

????? 2.18 ????: ?????? ?????????? 2.18 ???

??

??

??

??

??

??

??

??

????? ???

????????????

??

????????????????????????????????????

???????????????????????? 1 ?????????

???????? ???????

????????????????????????

????????????????

??? ?????????

?????? ???????

??

????????????????????????????????????

????????????????????

Weblate ????????????????????????? RTL ?????????

SPECIAL_CHARS ?????????

The screenshot displays the Weblate interface for translating a string. The main editor shows the source string 'אבנים' (Abnim) in Hebrew. Below the editor, there's a table of other languages:

Language	Status	Translation	Edit
Czech	✓	Soubory	Edit
English	🔍	Files	Edit
Hungarian	✓	Fájlok	Edit

The right sidebar provides additional details:

- Glossary:** No related strings found in the glossary.
- Source information:** Screenshot context (No screenshot currently associated), Explanation (No explanation currently provided), Labels (No labels currently set), Flags (No flags currently set).
- Source string location:** [weblate/templates/translation.html:45](#), [weblate/trans/forms.py:1404](#)
- String age:** 13 seconds ago
- Source string age:** 14 seconds ago
- Translation file:** [weblate/locale/he/LC_MESSAGES/django.po](#), string 1

WeblateOrg

Catalan 0 Czech 1 Dutch 0 English 0 French 0 Galician 0 German 0 Hebrew 0 Hungarian 0
Chinese (Simplified) 0 Polish 0 Russian 0 Spanish 0

Glossary name
WeblateOrg

Color
Navy Blue Aqua Teal Olive Green Lime Yellow Orange Red Maroon Fuchsia Purple Black Gray Silver

Source language
English

Additional projects
Search...
Available: Chosen:
Choose additional projects where this glossary can be used.

Save Delete

Create new glossary

Glossary name
[Empty field]

Color
Navy Blue Aqua Teal Olive Green Lime Yellow Orange Red Maroon Fuchsia Purple Black Gray Silver

Source language
English

Additional projects
Search...
Available: Chosen:
Choose additional projects where this glossary can be used.

Save



Browse Add new word Import glossary Export glossary History

Search

Starting letter Any

Source	Translation	Glossary	
language	jazyk	WeblateOrg	Edit Delete

???

Webblate

??:

???

???

???

[Weblate](#) [Dashboard](#) [Projects](#) [Languages](#) [Checks](#) + Add ...

[WeblateOrg](#) / [Django](#) / [Czech](#) translated 96%

[Overview](#) [Info](#) [Search](#) [Glossary](#) [Insights](#) [Files](#) [Tools](#) [Manage](#) [Share](#) Watching

Automatic translation ⓘ

Automatic translation takes existing translations in this project and applies them to a different component. It can be used to push translations to a different branch, to fix inconsistent translations or to translate a new component using existing translations.

Automatic translation via machine translation uses active machine translation engines to get the best possible translations and applies them in this project.

Automatic translation mode

Add as suggestion

Search filter

Strings needing action

Automatic translation source

Other translation components
 Machine translation

Machine translation engines

Search...

Available:	Chosen:
Weblate Weblate Translation Memory	Weblate

Score threshold

80

Apply

Powered by Weblate 4.3 [About Weblate](#) [Legal](#) [Contact](#) [Documentation](#) [Donate to Weblate](#)

2 [translations](#):

- [Weblate](#) [translations](#)
- [translations](#)
- [translations](#)

[Weblate](#) [translations](#)

[translations](#) WEB [translations](#)

[Weblate](#):

Keeping translations same across components

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

Navigation: < 1/1 > Custom Search: '%(count)s word' Position: [icon]

Translation

English

Singular
%(count)s word

Plural
%(count)s words

Czech, One
[input field]

Czech, Few
několik slov

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](#)
[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
[input field]

You can use Markdown and mention users by @username.

Save

Things to check

Python format 1
Following format strings are missing: %(count)s
Dismiss **Dismiss for all languages**

Missing plurals 2
Some plural forms are not translated
Dismiss **Dismiss for all languages**

Glossary

English	Czech
No related strings found in the glossary.	

+ Add term to glossary

Source information

Screenshot context
No screenshot currently associated.

Explanation
No explanation currently provided.

Labels
No labels currently set.

Flags
python-format

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

String age
10 seconds ago

Source string age
11 seconds ago

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

???

??? Weblate

??:

AUTOFIX_LIST

???

Weblate

??:

CHECK_LIST

????

BBcode

BBcode

BBCode

?: BBcode

????

2

4.1

: Weblate

????

2

????

Weblate C 'c-format'gettext PO xgettext

Additional info on source stringsComponent configuration

: Weblate

Webplate Dashboard Projects Languages Checks

WeblateOrg / Django / Czech / Translate translated 96%

Custom Search '%(count)s word' Position and priority Zen

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

Glossary

English Czech

No related strings found in the glossary.

Add term to glossary

Source information

Screenshot context
No screenshot currently associated.

Explanation
No explanation currently provided.

Labels
No labels currently set.

Flags
python-format

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

String age
6 seconds ago

Source string age
6 seconds ago

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

Nearby strings 20 Comments Automatic suggestions Other languages History

No matching activity found.

Browse all component changes

Powered by Weblate 4.3 About Weblate Legal Contact Documentation Donate to Weblate

AngularJS `{{}}`

AngularJS `{{}}`

```

Your balance is {{amount}} {{ currency }}
angularjs-format

```

`{{}}`:

AngularJS: API: `$interpolate`

C `%`

C `%`

```

There are %d apples
Your balance is %1$d %2$s
c-format

```

`%`:

C format strings `%` C printf format

C# `???`

C# `????????????????`

```
?????????There are {0} apples
????????? c-sharp-format
```

`??`:

C# String Format

ECMAScript `??????` `????`

ECMAScript `??????` `????????????????`

```
??          There are ${number} apples
?????????es-format
```

`??`:

Template literals

i18next `??`

i18next `????????????`

`????` 4.0 `??`.

```
??          There are {{number}} apples
????????? There are $t(number) apples
??????????i18next-interpolation
```

`??`:

i18next interpolation

Java `??`

Java `????????????????`

```
?????????There are %d apples
?????????Your balance is %1$d %2$s
????????? java-format
```

`??`:

Java Format Strings

Java MessageFormat

Java `????` `????????????`

```
?????????There are {0} apples
????????? java-messageformat ??????????
auto-java-messageformat ??????????
```

`??`:

Java MessageFormat

JavaScript [??](#)

JavaScript [????????????????????](#)

```
?????????????There are %d apples
????????????? javascript-format
```

??:

JavaScript formatting strings

[????????????????????](#)

[????????????????????](#)

????? 4.0 ???.

```
?????????????There are %number% apples
????????????? percent-placeholders
```

Perl [??](#)

Perl [????????????????????](#)

```
?????????????There are %d apples
?????????????Your balance is %1$d %2$s
????????????? perl-format
```

??:

Perl sprintf [?](#) Perl Format Strings

PHP [??](#)

PHP [????????????????????](#)

```
?????????????There are %d apples
?????????????Your balance is %1$d %2$s
????????????? php-format
```

??:

PHP sprintf documentation [?](#) PHP Format Strings

Python [????????](#)

Python [????????????????](#)

```
????????????? There are {} apples
?????????????Your balance is {amount} {currency}
????????????? python-brace-format
```

??:

Python brace format [?](#) Python Format Strings

Python [??](#)

Python format string does not match source

```
????????????? There are %d apples
?????????????????Your balance is %(amount) %(currency)
????????????? python-format
```

??:

Python string formatting [Python Format Strings](#)

Qt [??](#)

Qt [????????????????????](#)

```
?????????????There are %1 apples
????????????? qt-format
```

??:

Qt QString::arg()

Qt [????](#)

Qt [????????????????????](#)

```
?????????????There are %Ln apple(s)
????????????? qt-plural-format
```

??:

Qt i18n guide

Ruby [??](#)

Ruby [????????????????](#)

```
????????????? There are %d apples
????????????? Your balance is %1$f %2$s
?????????????????Your balance is %+.2<amount>f %<currency>s
?????????????????Your balance is %{amount} %{currency}
????????????? ruby-format
```

??:

Ruby Kernel#sprintf

Vue I18n [??](#)

Vue I18n [????????](#)

```
????????????? There are {count} apples
Rails i18n ?? There are %{count} apples
????????????? @:message.dio @:message.the_world!
????????????? vue-format
```

??:

Vue I18n Formatting [Vue I18n Linked locale messages](#)

????

????????????????

????????????????VCS ?????????????? VCS ??????????????

??????

??

Webplate??

????????? 1 ???
????????? ??????????

??: ???
???????????????????? ???
????? ??

??:

Keeping translations same across components

Kashida ?????

???? Kashida ?????????????????

???? 3.5 ???.

Kashida ????????????????????????????????????? Tatweel ??????????

??:

Kashida on Wikipedia

Markdown ?????

????? ?????????????????

????? 3.5 ???.

????? ?????????????????

??:

Markdown links

Markdown ???

????? ?????????????????

????? 3.5 ???.

????? ?????????????????

??:

Markdown links

Markdown [?]

?

3.5 ?

Markdown ?

?:

Markdown span elements

?????

?

?

max-length:100 key:value

?:

replacements:

?????

?

3.7 ?

?

1 2

font-* ubuntu 22

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

?: Component configuration font-*

replacements:

?:

?

\n [?]

\n ?

\n

?????

?

?

?:

Colon on Wikipedia

Ellipsis

⋮

⋮ 3 ⋮ (.) ⋮ (. .)

⋮:

Ellipsis on Wikipedia

Exclamation mark

!

!:

Exclamation mark on Wikipedia

Full stop

.

.:

Full stop on Wikipedia

Question mark

?

?:

Question mark on Wikipedia

Semicolon

;

;

Semicolon on Wikipedia

Backslash

\

\ n

Gettext

Gettext is a system for internationalizing software.

Gettext 4.1.0

Gettext is a system for internationalizing software.

Gettext is a system for internationalizing software.

Python

Python is a high-level, interpreted, interactive, object-oriented programming language.

Python is a high-level, interpreted, interactive, object-oriented programming language.

Python is a high-level, interpreted, interactive, object-oriented programming language.

```
from gettext import gettext
print gettext('Selected %d file', 'Selected %d files', files) % files
```

gettext

gettext 3.9.0

gettext is a system for internationalizing software.

gettext is a system for internationalizing software.

The screenshot shows the Weblate web interface. At the top, there is a navigation bar with 'Weblate', 'Dashboard', 'Projects', 'Languages', and 'Checks'. Below this is a 'Dashboard' section with 'Watched translations' (0), 'Suggested translations' (0), and 'Insights'. A search bar is present. The main content area is titled 'Search' and contains a 'Custom Search' field, a 'Sort By' dropdown, and an 'Advanced query builder' section. The 'Advanced query builder' has fields for 'Source strings', 'String changed after', and 'Exact' checkbox. Below this is a 'Query examples' section with several rows of example queries and their corresponding results, each with an 'Add' button. The examples include: 'Review strings changed by other users' (changed:>=2020-09-14 AND NOT changed_by:testuser), 'Translated strings' (state:>=translated), 'Strings with comments' (has:comment), 'Strings with any failing checks' (has:check), 'Strings with suggestions from others' (has:suggestion AND NOT suggestion_author:testuser), 'Approved strings with suggestions' (state:approved AND has:suggestion), 'All untranslated strings added the past month' (added:>=2020-09-14 AND state:<=needs-editing), and 'Translated strings in a certain language' (is:translated AND language:cs). A 'Search' button is at the bottom of the search section.

?????

()
(") : "this is a quoted string" 'another quoted
string'

????

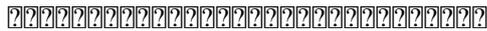
approved?translated?needs-editing?empty?read-only?rts
VCS
- plural?context?suggestion?comment?check?dismissed-
check?translation?variant?`screenshot`
pending?translated?untranslated?
Component slug?
Project slug?

?????

AND?OR?NOT: state:translated
AND (source:hello OR source:bar)

???????

:
translated?approved?
2019



Weblate Dashboard Projects Languages Checks

WeblateOrg / Django / Czech / Translate translated 96%

Navigation: < 1/1 >

Filter: Not translated strings state:empty

Translation

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

Czech Clone source

Needs editing

Save Suggest Skip

[Nearby strings](#) 16 [Comments](#) [Automatic suggestions](#) [Other languages](#) [History](#)

New comment

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

Scope
Translation comment, discussions with other translators

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

New comment

You can use Markdown and mention users by @username.

Save

- Position and priority
- Position
- Priority
- Labels
- Age of string
- Number of words
- Number of comments
- Number of failing checks
- Key

Glossary

English Czech

No related strings found in the glossary.

+ Add term to glossary

Source information

Screenshot context
No screenshot currently associated.

Explanation
No explanation currently provided.

Labels
No labels currently set.

Flags
No flags currently set.

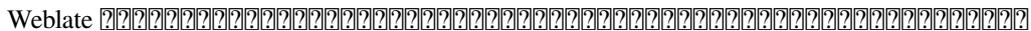
Source string location
weblate/checks/source.py:54

String age
8 seconds ago

Source string age
8 seconds ago

Translation file
weblate/locale/cs/LC_MESSAGES/django.po, string 26

Powered by Weblate 4.3 About Weblate Legal Contact Documentation Donate to Weblate



Starting with internationalization

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

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

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

Choosing internationalization framework

Choose whatever is standard on your platform, try to avoid reinventing the wheel by creating your own framework to handle localizations. Weblate supports most of the widely used frameworks, see *Supported file formats* for more information (especially *Translation types capabilities*).

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

Platform	Recommended format
Android	<i>Android string resources</i>
iOS	<i>Apple iOS strings</i>
Qt	<i>Qt Linguist .ts</i>
Python	<i>GNU gettext</i>
PHP	<i>GNU gettext¹</i>
C/C++	<i>GNU gettext</i>
C#	<i>.XML resource files</i>
Perl	<i>GNU gettext</i>
Ruby	<i>Ruby YAML files</i>
Web extensions	<i>WebExtension JSON</i>
Java	<i>XLIFF²</i>
JavaScript	<i>JSON i18next files³</i>

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

Translating software using GNU Gettext

Translating documentation using Sphinx

Translating HTML and JavaScript using Weblate CDN

Integrating with Weblate

Getting translations updates from Weblate

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

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

Web:

Web `weblate --update`

Pushing string changes to Weblate

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

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

Web:

Web `weblate --push`

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

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

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

Translating software using GNU Gettext

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

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

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

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

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

Sample program

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

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

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

Extracting translatable strings

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

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

⚠: There are alternative programs to extract strings from the code, for example [pybabel](#).

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

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
```

```

msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2015-10-23 11:02+0200\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=INTEGER; plural=EXPRESSION;\n"

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

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

```

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

Starting new translation

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

```

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

```

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

```

# Czech translations for PACKAGE package.
# Copyright (C) 2015 THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# Automatically generated, 2015.
#
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2015-10-23 11:02+0200\n"
"PO-Revision-Date: 2015-10-23 11:02+0200\n"
"Last-Translator: Automatically generated\n"
"Language-Team: none\n"
"Language: cs\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=ASCII\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=3; plural=(n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2;\n"

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

#: main.c:20

```

```
msgid "Thank you for using Weblate."
msgstr ""
```

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

Updating strings

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

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

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

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

Importing to Weblate

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

Field	Value
XXXXXXXXXXXXXXXXXXXX	URL of the VCS repository with your project
File mask	po/*.po
XXXXXXXXXXXXXXXXXXXX	po/hello.pot
XXXXXX	Choose <i>Gettext PO file</i>
XXXXXX	Choose <i>Create new language file</i>

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

☞:

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

Translating documentation using Sphinx

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

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

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

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

Table 1 Component configuration

XXXXXXXXXX	Documentation
File mask	docs/locales/*/LC_MESSAGES/index.po
XXXXXXXXXXXXXXXXXXXX	docs/locales/index.pot
XXXXXX	<i>gettext PO file</i>
XXXXXX	<i>rst-text</i>

Table2

```
docs/locales/(?P<language>[^\./]*)/LC_MESSAGES/(?
P<component>[^\./]*)\.po
Documentation: {{ component|title }}
docs/locales/{{ component }}.pot
```

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

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

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

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

Odorik:

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

Translating HTML and JavaScript using Weblate CDN

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

This feature is configured on Hosted Weblate. It requires additional configuration on your installation, see `LOCALIZE_CDN_URL` and `LOCALIZE_CDN_PATH`.

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

Getting started

First, you need to create a monolingual component which will hold your strings, see *Adding translation projects and components* for generic instructions on that.

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

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

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

Weblate CDN [? ? ? ? ? ?](#)

The *JavaScript [? ? ? ? ?](#) CDN* addon provides few configuration options.

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

Configures which strings from the HTML documents are translatable, see *Weblate CDN [? ? ? ? ? ?](#)* and *Weblate CDN [? ? ? ? ? ?](#) HTML [? ? ? ? ?](#)*.

Name of cookie which contains user selected language. Used in the JavaScript snippet for *Weblate CDN [? ? ? ? ? ?](#) HTML [? ? ? ? ?](#)*.

List of files in the repository or URLs where Weblate will look for translatable strings and offer them for a translation, see *Weblate CDN [? ? ? ? ? ?](#)*.

Weblate CDN [? ? ? ? ? ?](#)

The translation strings have to be present in Weblate. You can either manage these manually, use API to create them or list files or URLs using *Extract strings from HTML files* and Weblate will extract them automatically. The files have to present in the repository or contain remote URLs which will be download and parsed regularly by Weblate.

The default configuration for *CSS selector* extracts elements with CSS class `l10n`, for example it would extract two strings from following snippets:

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

In case you don't want to modify existing code, you can also use `*` as a selector to process all elements.

??: Right now, only text of the elements is extracted. This addon doesn't support localization of element attributes or elements with childs.

Weblate CDN [? ? ? ? ? ?](#) HTML [? ? ? ? ?](#)

To localize a HTML document, you need to load the `weblate.js` script:

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

Upon loading, this will automatically find all matching translatable elements (based on *CSS selector* configuration) and replace their text with a translation.

The user language is detected from the configured cookie and falls back to user preferred languages configured in the browser.

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

JavaScript

The individual translations are exposed as bilingual JSON files under the CDN. To fetch one you can use following code:

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

The actual localization logic needs to be implemented in this case.

Translation component alerts

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

Currently the following is covered:

Duplicated source strings in translation files

Duplicated languages within translations

Merge or update failures in the source repository

Unused new base in component settings

Parse errors in the translation files

Duplicate filemask used for linked components

Broken URLs

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

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

[Weblate](#)
[Dashboard](#)
[Projects](#)
[Languages](#)
[Checks](#)
+ Add

[WeblateOrg](#) / [Duplicates](#)
translated 37%

[Translations](#)
[Info](#)
[Alerts](#)
[Search](#)
[Glossaries](#)
[Insights](#)
[Files](#)
[Tools](#)
[Manage](#)
[Share](#)
Not watching

Duplicated string found in the file.

The component contains several duplicated translation strings.

The following occurrences were found:

Language	Source
Italian	Thank you for using Weblate.

Please fix this by removing duplicated strings with same identifier from the translation files.

Appeared a second ago, last seen a second ago

Duplicated translation.

The component contains several translation files mapped to a single language in Weblate. Please fix this by removing one of the translation files.

Please consider the following:

- Avoid having translation files for both the plain language code and its equivalent territory designation (for example de and de_DE).

The following occurrences were found:

Language	Language codes
Czech	cs_CZ, cs

Appeared a second ago, last seen a second ago

License info missing.

Any publicly available project should have defined license to indicate what terms apply to contributions.

Appeared a second ago, last seen a second ago

Powered by Weblate 4.3
 [About Weblate](#)
[Legal](#)
[Contact](#)
[Documentation](#)
[Donate to Weblate](#)

??:

Using custom certificate authority

Building translators community

????????????????????

????? 3.9 ???.

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

Webate Dashboard Projects Languages Checks

WeblateOrg / Duplicates / Community localization checklist translated 37%

Community localization checklist

Here you can find guidance to make your localization project attractive to the community.

Version control integration

- Configure repository hooks for automated flow of updates to Weblate. [Configure](#)
- Configure push URL for automated flow of translations from Weblate. [Configure](#)

Building community

- Define translation instructions to give translators a guideline. [Configure](#)
- Make your translations available under a libre license. [Configure](#)
- Fix this component to clear its alerts. [Configure](#)

Provide context to the translators

- Add screenshots to show where strings are being used. [Configure](#)
- Use flags to indicate special strings in your translation. [Configure](#)

Workflow customization

- Enable addon: Update LINGUAS file
Updates the LINGUAS file when a new translation is added. [Configure](#)
- Enable addon: Update ALL_LINGUAS variable in the "configure" file
Updates the ALL_LINGUAS variable in "configure", "configure.in" or "configure.ac" files, when a new translation is added. [Configure](#)

[Return to the component](#)

Powered by Weblate 4.3 [About Weblate](#) [Legal](#) [Contact](#) [Documentation](#) [Donate to Weblate](#)

Managing translations

Adding new translations

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

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

 Project admins can always start translation within Weblate directly.

Language files added manually to the VCS are added to the component when Weblate updates the repository. About repository update settings, see *Updating repositories*.

String variants

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

WebplateOrg / Android / Settings

Basic Translation Version control Commit messages Files

Suggestions

Turn on suggestions
Whether to allow translation suggestions at all.

Suggestion voting
Whether users can vote for suggestions.

Autoaccept suggestions ⓘ

0
Automatically accept suggestions with this number of votes, use 0 to turn it off.

Translation settings

Allow translation propagation
Whether translation updates in other components will cause automatic translation in this one

Translation flags ⓘ

Additional comma-separated flags to influence quality checks. Possible values can be found in the documentation.

Variants regular expression ⓘ

_(short|min)\$
Regular expression used to determine variants of a string.

Enforced checks ⓘ

Search...

Available:	Chosen:
AngularJS interpolation string	
BBcode markup	
C format	
C# format	
Consecutive duplicated words	

List of checks which can not be ignored.

Priority ⓘ

Medium
Components with higher priority are offered first to translators.

Restricted component
Restrict access to the component to only those explicitly given permission.

Save

Powered by Weblate 4.3 [About Weblate](#) [Legal](#) [Contact](#) [Documentation](#) [Donate to Weblate](#)

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

The following table lists some usage examples:

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

The variant is later grouped when translating:

The screenshot shows the Weblate interface for a string with the key `dow_monday`. The source string is "Monday" and the target string is "Monday". The interface includes a navigation bar, a search bar, and a sidebar with various panels.

Things to check

- Variants**: There are 3 variants for this string. [View](#)

Glossary

English English
No related strings found in the glossary.
[+ Add term to glossary](#)

Source information

- Screenshot context**: No screenshot currently associated.
- Explanation**: No explanation currently provided.
- Key**: `dow_monday`
- Labels**: No labels currently set.
- Flags**: java-format
- String age**: 7 seconds ago
- Source string age**: 7 seconds ago
- Translation file**: `app/src/main/res/values/strings.xml`, string 11

String list

Key	English	State
<code>dow_monday</code>	Monday	✓
<code>dow_monday_min</code>	M	✓
<code>dow_monday_short</code>	Mon	✓

String labels

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

[Weblate](#)
[Dashboard](#)
[Projects](#)
[Languages](#)
[Checks](#)

[+ Add](#)

[WeblateOrg](#) / Labels

Label name	Color	
Current sprint	Green	Edit Delete
Next sprint	Aqua	Edit Delete

Add label

Label name

Color

[Navy](#)
[Blue](#)
[Aqua](#)
[Teal](#)
[Olive](#)
[Green](#)
[Lime](#)
[Yellow](#)
[Orange](#)
[Red](#)
[Maroon](#)
[Fuchsia](#)
[Purple](#)
[Black](#)
[Gray](#)
[Silver](#)

[Save](#)

Powered by Weblate 4.3 [About Weblate](#) [Legal](#) [Contact](#) [Documentation](#) [Donate to Weblate](#)

🔗: Labels can be assigned to units in *Additional info on source strings* by bulk editing, or using the [🔗🔗🔗🔗](#) addon.

Reviewing strings

Activity reports

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

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

Dashboard

Watched translations 0

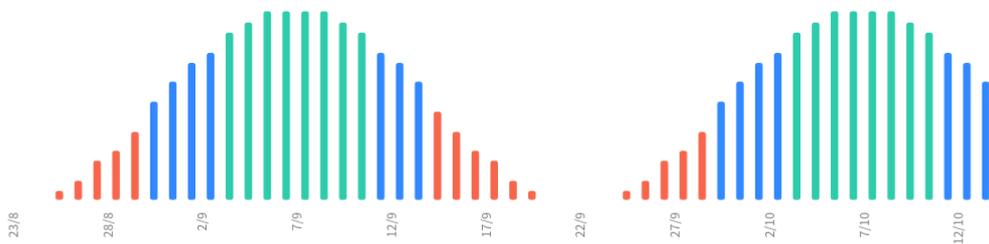
Suggested translations 0

Insights

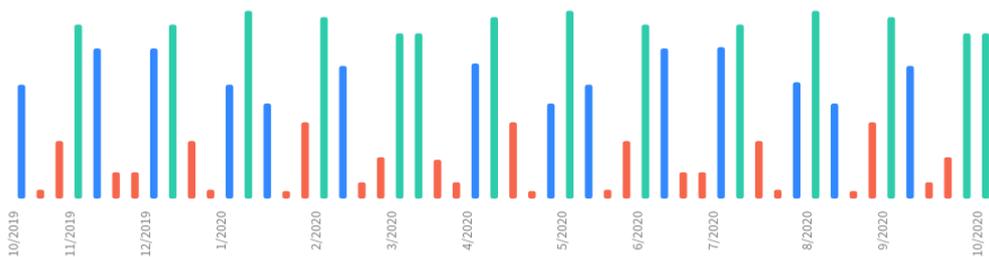
Search



Daily activity



Weekly activity



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

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

Source strings checks

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

Translation string checks

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

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

The screenshot shows the Weblate interface for a component named 'WeblateOrg / Android / English'. The top navigation bar includes 'Dashboard', 'Projects', 'Languages', and 'Checks'. The main content area is divided into several sections:

- Source strings:** A summary showing 13 strings and 46 words, both at 100% translation. A 'Translate' button is visible.
- Strings status:** A summary showing 13 items: All strings (46 words), Translated strings (46 words), and Strings without a label (46 words).
- Other components:** A table listing other components with their translation status and check counts.

Component	Translated	Untranslated	Untranslated words	Checks	Suggestions	Comments
Language names 🇺🇸🇪🇸🇮	✓					
Django 🇺🇸🇪🇸🇮	✓			1		

At the bottom, there is a 'Browse all components' button and a footer with 'Powered by Weblate 4.3' and various links.

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

The detailed listing is a per language overview:

Webplate Dashboard Projects Languages Checks

WeblateOrg / Android / English / Translate translated 100%

1/3 Custom Search Monday Position and priority Zen

Source string

Key
dow_monday

English
Monday

Needs editing 6/100

Save Suggest Skip Remove

Things to check

Variants
There are 3 variants for this string.
View

Glossary
English English
No related strings found in the glossary.
Add term to glossary

Source information

Screenshot context
No screenshot currently associated.

Explanation
No explanation currently provided.

Key
dow_monday

Labels
No labels currently set.

Flags
java-format

String age
7 seconds ago

Source string age
7 seconds ago

Translation file
app/src/main/res/values/strings.xml, string 11

Nearby strings 13 Nearby keys 13 Variants 3 Comments Other languages History

Key	English	State
auth_activity_title	Authenticate	✓
auth_hint_password	Password	✓
auth_hint_pin	PIN	✓
auth_msg_authenticate	Please authenticate to start andOTP!	✓
auth_msg_confirm_encryption	Please confirm your authentication to generate the new encryption key!	✓
auth_button_unlock	Unlock	✓
auth_toast_password_missing	Please set a password in the Settings!	✓
auth_toast_pin_missing	Please set a PIN in the Settings!	✓
auth_toast_password_again	Wrong password, please try again!	✓
auth_toast_pin_again	Wrong PIN, please try again!	✓
dow_monday	Monday	✓
dow_monday_short	Mon	✓
dow_monday_min	M	✓

Powered by Weblate 4.3 About Weblate Legal Contact Documentation Donate to Weblate

String comments

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

Promoting the translation

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



🏠 WeblateOrg / Widgets

Promoting translation projects

You can point newcomers to the introduction page at <http://localhost:49135/engage/weblateorg/>.

Promoting specific translations

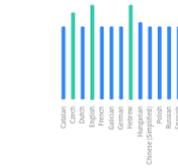
Besides promoting the whole translation project, you can also choose a specific language or component to promote:

Image widgets

You can use the following widgets to promote translation of your project. They can increase the visibility of your translation projects and bring in new contributors.



Vertical language bar chart



Horizontal language bar chart



Big status badge



Small status badge



Panel

Color variants:



HTML code

```
<a href="http://localhost:49135/engage/weblateorg/">  

```

Powered by Weblate 4.3 [About Weblate](#) [Legal](#) [Contact](#) [Documentation](#) [Donate to Weblate](#)

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

Get involved in **WeblateOrg**

Hello and thank you for your interest – WeblateOrg is being translated using Weblate, a web tool designed to ease translating for both developers and translators.

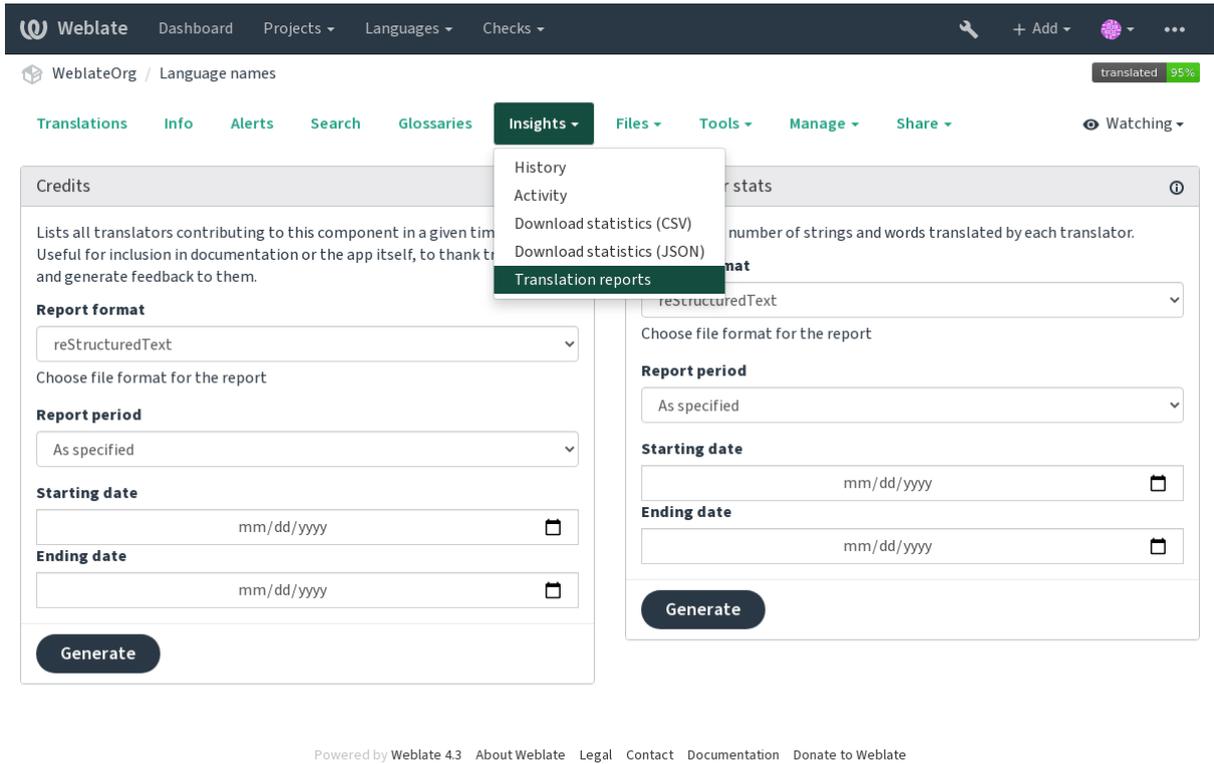


The translation project for WeblateOrg currently contains **35 string** for translation. It is being translated into **13 languages**. Overall, these translations are **85.2% complete**. If you would like to contribute to translation of WeblateOrg, you need to register on this server. This translation is open only to a limited group of translators, if you want to contribute please get in touch with the project maintainers.

[Translate](#) [View project languages](#)



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



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

Translator credits

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

```
* Czech
  * Michal Čihař <michal@cihar.com> (10)
  * John Doe <john@example.com> (5)
* Dutch
  * Jane Doe <jane@example.com> (42)
```

It will render as:

????

Michal Čihař <michal@cihar.com> (10)

John Doe <john@example.com> (5)

??????

Jae Doe <jane@example.com> (42)

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

????????

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

```
=====
->=====
->=====
->=====
->=====
->=====
->=====
->=====
->=====
->=====
->=====
Name
->      Count total      Source words total      Source chars
->total      Target words total      Target chars total      Count new
->      Source words new      Source chars new      Target
->words new      Target chars new      Count approved
->Source words approved      Source chars approved      Target words approved
->      Target chars approved      Count edited      Source words edited
->      Source chars edited      Target words edited      Target chars edited
=====
->=====
->=====
->=====
->=====
->=====
->=====
->=====
->=====
->=====
->=====
Michal Čiharř
->      1      3      21      24      0      0      0
->      24      1      3      21      0      0      0
->      3      0      0      0      0      0      0
->      0      0      0      0      0      0      0
Allan Nordhøy
->      2      4      28      24      0      0      0
->      25      2      3      21      0      0      0
->      3      0      0      0      0      0      0
->      0      0      0      0      0      0      0
=====
->=====
->=====
->=====
->=====
->=====
->=====
->=====
->=====
->=====
->=====
```

And it will get rendered as:

????

?? Weblate ?????????????????
????????????????????????????
??

?? Value??
???????? off ?????????????????
???????? on ???
???????? off
?????????0
???????????????????????? ?????????? ? Translate ?
????????????????????????????

????

??
????????????????????????????
????????????????????????????
??

?? Value??
???????? off ?????????????????
???????? on
???????? off
?????????1 ???
???????????????????????? ?????????? ? Translate ?
????????????????????????????

???????

?????? 2.18 ????: ????????????????? Weblate 2.18 ??????
????????????????2 ????????? ?????????????1 ????????????????????? 1 ?????????????????????????????????????
????????????????????????????
??? ?????????????????
??? ?????????????????????????????????????
????????????????????????????????????

?? Value??
???????? on ?????????????????
???????? off ?????????????????????????????????????
???????? off
?????????0
???????????????????????? ?????????? ? Translate ?
????????????????????????????

??????

Workflow Manage → Settings:

Webate Dashboard Projects Languages Checks

WebateOrg / Settings

Basic Access **Workflow** Components

- Set "Language-Team" header**
Lets Weblate update the "Language-Team" file header of your project.
- Use shared translation memory**
Uses the pool of shared translations between projects.
- Contribute to shared translation memory**
Contributes to the pool of shared translations between projects.
- Enable hooks**
Whether to allow updating this repository by remote hooks.

Language aliases

Comma-separated list of language code mappings, for example: en_GB:en,en_US:en

- Enable reviews**
Requires dedicated reviewers to approve translations.
- Enable source reviews**
Requires dedicated reviewers to approve source strings.

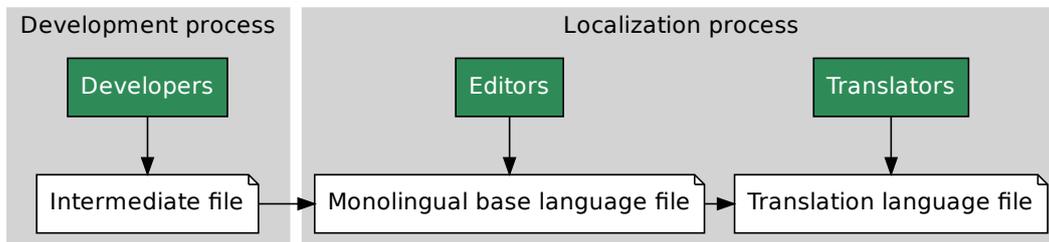
Save

Powered by Weblate 4.3 [About Weblate](#) [Legal](#) [Contact](#) [Documentation](#) [Donate to Weblate](#)

Webate Hosted Weblate

??????

Workflow Manage → Settings:



??:

Bilingual and monolingual formats

????

????? ?

????? reviews`????? - ??????:guiLabel:`?? ?

????? Source needs re- view` ?????`

??:

Bilingual and monolingual formats`String labels

Frequently Asked Questions

??

?????

Weblate ?????

- 1.???? Weblate ????? Git ?????
- 2.Weblate ? *Component configuration* ????? URL ?????Weblate ?????
- 3.Weblate ? *Project configuration* ? push-on-commit ?????Weblate ????? Weblate ?????

??:

Web ?Avoiding merge conflicts

SSH ?

SSH ?????

?????

?????Weblate ? upstream ?????msgmerge
?????Weblate ?????Weblate ?????
?????

????? Weblate
????? upstream ?????Weblate
?????

??: ?????Weblate ?????Weblate ????? *Git exporter*
????? API ?????

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

# Update weblate remote:
git remote update weblate

# Merge Weblate changes:
git merge weblate/master
```

(XXXXXXXXXX)

```

# 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

```

Weblate XXX

```

# 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 master branch:
git checkout master
git merge weblate-second/master
... # Resolve conflicts
git commit

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

```

gettext PO XXX

Weblate Git XXXXXXXXXXXXXXX XXXupstream ? Git XXXXXXXXXXXXXXXXXXXXXXX XXXupstream ? Git XXXXXXXXXXXXXXXXXXXXXXX 2 XXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

```

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

# Update Weblate remote:
git remote update weblate

# Merge Weblate changes:
git merge weblate/master

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

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

```

??:

How to export the Git repository that Weblate uses? Web XXXXXXXXXXXXXXXXXXXXXXX Avoiding merge conflicts

????????????????????

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

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

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

🔗:

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.

🔗:

Keeping translations same across components

How to export the Git repository that Weblate uses?

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

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

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

What are the options for pushing changes back upstream?

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

Weblate automatically pushes and merges changes (see *????????????????????????????????*).

You manually tell Weblate to push (it needs push access to the upstream repository).

Somebody manually merges changes from the Weblate git repository into the upstream repository.

Somebody rewrites history produced by Weblate (e.g. by eliminating merge commits), merges changes, and tells Weblate to reset the content in the upstream repository.

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

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

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

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

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

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

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

Please consult the *git submodule* documentation for more details.

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

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

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

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

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

??:

Component configuration

Usage

How do I review the translations of others?

You can subscribe to any changes made in ?? and then check others contributions as they come in by e-mail.

There is a review tool available at the bottom of the translation view, where you can choose to browse translations made by others since a given date.

How do I provide feedback on a source string?

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

How can I use existing translations while translating?

Use the import functionality to load compendium as translations, suggestions or translations needing review. This is the best approach for a one-time translation using a compendium or a similar translation database.

You can set up *tmserver* with all databases you have and let Weblate use it. This is good when you want to use it several times during translation.

Another option is to translate all related projects in a single Weblate instance, which will make it automatically pick up translations from other projects as well.

??:

????? ??????

Does Weblate update translation files besides translations?

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

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

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

??:

Processing repository with scripts

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

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

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

Can Weblate highlight changes in a fuzzy string?

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

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

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

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

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

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

Tip: 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 add-on *POT to PO (msgmerge)*.

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
```

Tip:

`DATA_DIR`

Why does Weblate use language codes such sr_Latn or zh_Hant?

These are language codes defined by [RFC 4646](#) to better indicate that they are really different languages instead previously wrongly used modifiers (for @latin variants) or country codes (for Chinese).

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

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.

??:

Translation Related File Formats

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

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 `strings` within Weblate, though the naming might vary in your paradigm.

Additionally this workflow can be extended by utilizing `strings` to include strings provided by developers, but not to be used as is in the final strings.

????

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).

Translation types capabilities

Capabilities of all supported formats:

Format	Linguality ¹	Plurals ²	Comments ³	Context ⁴	Location ⁵	Flags ⁸	Additional states ⁶
<i>GNU gettext</i>	bilingual	yes	yes	yes	yes	yes ⁹	needs editing
<i>Monolingual gettext</i>	mono	yes	yes	yes	yes	yes ⁹	needs editing
<i>XLIFF</i>	both	yes	yes	yes	yes	yes ¹⁰	needs editing, approved
<i>Java proper-ties</i>	both	no	yes	no	no	no	
<i>GWT proper-ties</i>	mono	yes	yes	no	no	no	
<i>Joomla translations</i>	mono	no	yes	no	yes	no	
<i>Qt Linguist .ts</i>	both	yes	yes	no	yes	yes ¹⁰	needs editing
<i>Android string resources</i>	mono	yes	yes ⁷	no	no	yes ¹⁰	
<i>Apple iOS strings</i>	bilingual	no	yes	no	no	no	
<i>PHP</i> <code>???</code>	mono	no ¹¹	yes	no	no	no	

????????

Table 3 – [gettext file formats](#)

Format	Linguality ¹	Plurals ²	Comments ³	Context ⁴	Location ⁵	Flags ⁸	Additional states ⁶
<i>JSON files</i>	mono	no	no	no	no	no	
<i>JSON i18next files</i>	mono	yes	no	no	no	no	
<i>go-i18n JSON files</i>	mono	yes	no	no	no	no	
<i>ARB File</i>	mono	yes	yes	no	no	no	
<i>WebExtension JSON</i>	mono	yes	yes	no	no	no	
<i>.XML resource files</i>	mono	no	yes	no	no	yes ¹⁰	
<i>CSV files</i>	mono	no	yes	yes	yes	no	needs editing
<i>YAML files</i>	mono	no	yes	no	no	no	
<i>Ruby YAML files</i>	mono	yes	yes	no	no	no	
<i>DTD files</i>	mono	no	no	no	no	no	
<i>Flat XML</i>	mono	no	no	no	no	yes ¹⁰	
<i>Windows RC files</i>	mono	no	yes	no	no	no	
<i>Excel XML</i>	Openmono	no	yes	yes	yes	no	needs editing
gettext	gettext mono	no	no	no	no	no	
gettext	gettext						
<i>Subtitle files</i>	mono	no	no	no	yes	no	
<i>HTML files</i>	mono	no	no	no	no	no	
<i>OpenDocument Format</i>	mono	no	no	no	no	no	
<i>IDML Format</i>	mono	no	no	no	no	no	
<i>INI translations</i>	mono	no	no	no	no	no	
<i>Inno Setup INI</i>	mono	no	no	no	no	no	

GNU gettext

Most widely used format for translating libre software. This was first format supported by Weblate and still has the best support.

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

The bilingual gettext PO file typically looks like this:

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

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

[gettext file formats](#)

See *Bilingual and monolingual formats*

Plurals are necessary to properly localize strings with variable count.

Comments can be used to pass additional info about the string to translate.

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).

Location of a string in source code might help proficient translators figure out how the string is used.

See [gettext file formats](#)

Additional states supported by the file format in addition to "Not translated" and "Translated".

The gettext type comments are used as flags.

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*.

XML comment placed before the `<string>` element, parsed as a developer comment.

The plurals are supported only for Laravel which uses in string syntax to define them, see [Localization in Laravel](#).

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

Typical Weblate *Component configuration*

```
XXXXXXXXXX po/*.po
XXXXXXXXXXXXXXXXXXXXXXXXXX Empty
XXXXXXXXXXXXXXXXXXXXXXXXXX po/messages.pot
XXXXXXXXXX Gettext PO file
```

☞:

[Translating software using GNU Gettext](#) [Translating documentation using Sphinx](#) [Gettext on Wikipedia](#) [PO Files](#) [configure](#) [ALL_LINGUAS](#) [gettext](#) [LINGUAS](#) [MO](#) [POT](#) [PO](#) [\(msgmerge\)](#)

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:

```
#: 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*

```
XXXXXXXXXX po/*.po
XXXXXXXXXXXXXXXXXXXXXXXXXX po/en.po
XXXXXXXXXXXXXXXXXXXXXXXXXX po/messages.pot
XXXXXXXXXX Gettext PO file (monolingual)
```

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.

[\[?\]:](#)

XML Localization Interchange File Format (XLIFF) specification

Translation states

[\[?\]\[?\]\[?\]\[?\]](#) 3.3 [\[?\]\[?\]](#): Weblate ignored the state attribute prior to the 3.3 release.

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

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

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

The `state` attribute is only added in case string is marked as needing edit.

The `approved` attribute is only added in case string has been reviewed.

In other cases the attributes are not added, but they are updated in case they are present.

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

See [\[?\]\[?\]\[?\]\[?\]\[?\]](#).

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

Whitespace and newlines in XLIFF

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

For example:

```
<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 [\[?\]\[?\]\[?\]\[?\]\[?\]\[?\]\[?\]](#)) by using the `weblate-flags` attribute. Weblate also understands `maxwidth` and `font` attributes from the XLIFF specification:

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

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

INI translations

4.1

INI file format for translations.

INI translations are usually used as monolingual translations.

Typical Weblate	Component configuration
	language/*.ini
	language/en.ini
	Empty
	INI File

:

INI Files Joomla translations Inno Setup INI

Inno Setup INI

4.1

Inno Setup INI

Inno Setup INI

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

Typical Weblate	Component configuration
	language/*.isl
	language/en.isl
	Empty
	Inno Setup INI File

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

:

INI Files Joomla translations INI translations

Joomla translations

2.12

Native Joomla format for translations.

Joomla translations are usually used as monolingual translations.

Typical Weblate	Component configuration
	language/*/com_foobar.ini
	language/en-GB/com_foobar.ini
	Empty
	Joomla Language File

:

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

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

???	i18n/app.*.ts
????????????????	Empty
????????????????	i18n/app.de.ts
???????	Qt Linguist Translation File

Typical Weblate *Component configuration* when using as monolingual

???????	i18n/app.*.ts
????????????????	i18n/app.en.ts
????????????????	i18n/app.en.ts
???????	Qt Linguist Translation File

??:

Qt Linguist manual ?Qt .ts?Bilingual and monolingual formats

Android string resources

Android specific file format for translating applications.

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

Typical Weblate *Component configuration*

???????	res/values-*/strings.xml
????????????????	res/values/strings.xml
????????????????	Empty
???????	Android String Resource

??:

Android string resources documentation ?Android string resources

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

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

become:

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

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

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

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*

```
resources/*.lproj/Localizable.strings
resources/en.lproj/Localizable.strings or Resources/Base.lproj/
Localizable.strings
Empty
iOS Strings (UTF-8)
```

🔗:

Apple "strings files" documentation 📄 Mac OSX strings

PHP 📄📄

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

Example file:

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

Typical Weblate *Component configuration*

```
lang/*/texts.php
lang/en/texts.php
lang/en/texts.php
PHP strings
```

Laravel PHP 📄📄

📄📄 4.1 📄📄.

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

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

🔗:

PHP📄 Localization in Laravel

JSON files

📄📄📄 2.0 📄📄.

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

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

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

Weblate currently supports several variants of JSON translations:

Simple key / value files, used for example by *vue-i18n* or *react-intl*.

Files with nested keys.

JSON i18next files

go-i18n JSON files
WebExtension JSON
ARB File

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

Example file:

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

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

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

???: The *JSON file* and *JSON nested structure file* can both handle same type of files. The only difference between them is when adding new strings. The nested variant tries to parse the key and insert the new string into the matching structure.

Typical Weblate *Component configuration*

```
????????? langs/translation-*.json
????????????????????? langs/translation-en.json
????????????????????? Empty
????????? JSON nested structure file
```

??:

JSON **JSON** **????????????????????? ?** **????????????????????????????**,

JSON i18next files

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

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

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

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

Example file:

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

Typical Weblate *Component configuration*

```
lang/**.json
lang/en.json
Empty
i18next JSON file
```

JSON:

JSON i18next JSON Format [JSON](#) [i18next JSON Format](#)

go-i18n JSON files

[go-i18n](#) 4.1 [JSON](#).

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

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

Typical Weblate *Component configuration*

```
lang/**.json
lang/en.json
Empty
go-i18n JSON file
```

JSON:

JSON go-i18n [JSON](#) [go-i18n JSON](#)

ARB File

[ARB](#) 4.1 [JSON](#).

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

Typical Weblate *Component configuration*

```
lib/l10n/intl_*.arb
lib/l10n/intl_en.arb
Empty
ARB file
```

JSON:

JSON [Application Resource Bundle Specification](#) [Internationalizing Flutter apps](#) [JSON](#)

WebExtension JSON

[WebExtension JSON](#) 2.16 [JSON](#): 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.

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

Example file:

```

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

```

Typical Weblate *Component configuration*

```

_locales/*/messages.json
_locales/en/messages.json
Empty
WebExtension JSON file

```

🔗:

JSON🔗Google chrome.i18n🔗Mozilla Extensions Internationalization

.XML resource files

🔗🔗🔗 2.3 🔗🔗.

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

Typical Weblate *Component configuration*

```

Resources/Language.*.resx
Resources/Language.resx
Empty
.NET resource file

```

🔗:

.NET Resource files (.resx)🔗ref:addon-weblate.cleanup.generic🔗

CSV files

🔗🔗🔗 2.4 🔗🔗.

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

Files with header defining fields (source, translation, location, ...). This is the recommended approach, as it is the least error prone.

Files with two fields—source and translation (in this order), choose *Simple CSV file* as file format

Files with fields as defined by translate-toolkit: location, source, target, ID, fuzzy, context, translator_comments, developer_comments

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

Example file:

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

Typical Weblate *Component configuration*

```
????????? locale/*.csv
????????????????????? Empty
????????????????????? locale/en.csv
????????? CSV file
```

??:
CSV

YAML files

?????? 2.9 ????

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: ""
```

Typical Weblate *Component configuration*

```
????????? translations/messages.*.yaml
????????????????????? translations/messages.en.yaml
????????????????????? Empty
????????? YAML file
```

??:
YAML *Ruby YAML files*

Ruby YAML files

?????? 2.9 ????

Ruby i18n YAML files with language as root node.

Example Ruby i18n YAML file:

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

Typical Weblate *Component configuration*

```
????????? translations/messages.*.yaml
????????????????????? translations/messages.en.yaml
????????????????????? Empty
????????? Ruby YAML file
```

??:
YAML *YAML files*

DTD files

2.18

Example DTD file:

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

Typical Weblate *Component configuration*

```
locale/*.*dtd
locale/en.*dtd
Empty
DTD file
```

:

Mozilla DTD format

Flat XML files

3.9

Example of a flat XML file:

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

Typical Weblate *Component configuration*

```
locale/*.*xml
locale/en.*xml
Empty
Flat XML file
```

:

Flat XML

Windows RC files

4.1: Support for Windows RC files has been rewritten.

: 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*

```
lang/* .rc
lang/en-US .rc
lang/en-US .rc
RC file
```

Windows RC files

lang/* .rc

3.5 Metadata

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

Triple-T gradle-play-publisher

Fastlane

F-Droid

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

Typical Weblate *Component configuration*

```
fastlane/android/metadata/*
fastlane/android/metadata/en-US
fastlane/android/metadata/en-US
App store metadata files
```

Subtitle files

3.7 Metadata

Weblate lang/en-US

SubRip subtitle file (*.srt)

MicroDVD subtitle file (*.sub)

Advanced Substation Alpha subtitles file (*.ass)

Substation Alpha subtitle file (*.ssa)

Typical Weblate *Component configuration*

```
path/*.srt
path/en.srt
path/en.srt
SubRip subtitle file
```

Subtitles

Subtitles

Excel Open XML

3.2 Metadata

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.

HTML files

4.1

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.

??:

HTML

OpenDocument Format

4.1

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.

??:

OpenDocument Format

IDML Format

4.1

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.

???

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.

??:

Translation Related File Formats

Adding new translations

2.18: In versions prior to 2.18 the behaviour of adding new translations was file format specific.

Weblate can automatically start new translation for all of the file formats.

Some formats expect to start with an empty file and only translated strings to be included (for example *Android string resources*), while others expect to have all keys present (for example *GNU gettext*). In some situations this really doesn't depend on the format, but rather on the framework you use to handle the translation (for example with *JSON files*).

When you specify `translation_template` in *Component configuration*, Weblate will use this file to start new translations. Any exiting translations will be removed from the file when doing so.

When *Template for new translations* is empty and the file format supports it, an empty file is created where new strings will be added once they are translated.

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

Dependent on file format, for most of them POSIX is used.

Typically used by gettext and related tools, produces language codes like *pt_BR*.

POSIX style language code including the country code even when not necessary (for example *'cs_CZ'*).

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').

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

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

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

Read-only strings

Since Weblate 3.10

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

Supported VCS

Weblate supports [Git](#), [GitHub](#), [Gerrit](#), [Subversion](#) and [Mercurial](#).

Git

Use the following URL to clone Weblate from GitHub: `https://github.com/WeblateOrg/weblate.git`

Hosted Weblate

Hosted Weblate is available on [GitHub](#), [Bitbucket](#), [Codeberg](#), [GitLab](#) and [Weblate push user](#).

GitHub repository ID is 5.

SSH URL: `git@github.com:WeblateOrg/weblate.git`

SSH

SSH URL: `ssh://git@github.com:WeblateOrg/weblate.git`

Tip: GitHub repository ID is 1 for Hosted Weblate.

Weblate SSH URL: `ssh://git@github.com:WeblateOrg/weblate.git`

Weblate SSH URL: `ssh://git@github.com:WeblateOrg/weblate.git`

Public SSH key

Weblate currently uses this SSH key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCAQDVraQDSG/jX3xVJN9KIkwWliZO13s7358s4xrlIMLgvTOpuqBZhv+jyvgbGFen5uZUEJJPMo3e4LAGzydVFHHnkT9RJACcde4ZJaw
```

[Download private key](#)

Known host keys

Hostname	Key type	Fingerprint
github.com	ssh-rsa	nThbg6kXUpJWGI7E1IGOCspRomTxdCARLviKw6E5SY8

Add host key

To access SSH hosts, its host key needs to be verified. You can get the host keys by entering a domain name or IP for the host in the form below.

Hostname
Port

[Submit](#)

Weblate SSH

Weblate [About](#)

[SSH keys](#) Weblate

SSH

Weblate SSH

SSH

Weblate [SSH](#)

[Add](#) [host](#) [key](#) [gitlab.com](#) [Submit](#)

Added host key for github.com with fingerprint nThbg6kXUpJWGI7E1IGOCspRomTxdCARLviKw6E5SY8 (ssh-rsa), please verify that it is correct.

Webplate status Backups Translation memory Performance report SSH keys Alerts Repositories Users Tools

Public SSH key
Webplate currently uses this SSH key:
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCAQDVRaQD5G/jX3xVJN9K1kwWlIZO13s7358s4xrlIMLgVTOpuqBZhv+jyvgbGFen5uZUEJJPMo3e4LAGzydVFHnkt9RJACcde4ZJaw
Download private key

Known host keys table with columns: Hostname, Key type, Fingerprint. Row: github.com, ssh-rsa, nThbg6kXUpJWGI7E1IGOCspRomTxdCARLviKw6E5SY8

Add host key form with fields for Hostname (github.com) and Port, and a Submit button.

GitHub

SSH SSH GitHub

push upstream

HTTPS GitHub Creating an access token for command-line use

Webplate Webplate SSH Weblate SSH webplate Hosted Webplate weblate

:

Hosted Webplate

Webplate URL

1 weblate://project/component URL VCS project/component

Webplate URL

:

1

Webplate 1 Git exporter

Git

HTTPS

HTTPS URL Weblate URL

GitHub URL: https://user:your_access_token@github.com/WeblateOrg/weblate.git

: URL https://user%40example.com:%24password%23@bitbucket.org/...

HTTP/HTTPS VCS VCS

documentation http_proxy https_proxy all_proxy VCS:

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

: Weblate *Filesystem permissions* `HOME=$DATA_DIR/home` DATA_DIR Weblate Git`

:

The cURL manpage Git config documentation

Git

:

Git

Git

```
: upstream
```

Git

Weblate VCS HOME=\$DATA_DIR/home DATA_DIR DATA_DIR/home/.git

Git

remote helpers

Bazaar Mercurial GitHub: git-remote-hg git-remote-bzr ~/bin

Weblate

Bazaar Launchpad gnuhello

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

Mercurial selenic.com hello

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

Tip: Git Mercurial tip

GitHub

2.3

GitHub API Git

Git GitHub Git

Tip:

Pushing changes from Weblate

GitHub

1

You need to configure API credentials to make this work.

Tip:

GITHUB_USERNAME GITHUB_TOKEN GITHUB_CREDENTIALS

GitLab

3.9

GitLab API Git

Git Git GitLab

Tip:

Pushing changes from Weblate

GitLab

1

You need to configure API credentials to make this work.

Tip:

GITLAB_USERNAME GITLAB_TOKEN GITLAB_CREDENTIALS

Gerrit

2.2

git-review Gerrit Git

Gerrit

Mercurial

2.1

Mercurial Weblate 1 VCS

URL: Mercurial Weblate

URL:

Subversion

2.8

Weblate git-svn Subversion Git Subversion Perl

URL: Weblate Subversion URL
branches/tags/trunk/ git-svn documentation
URL

2.19:

Subversion

Weblate DATA_DIR 'svn'
\$HOME DATA_DIR

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

URL:

DATA_DIR

3.8

Weblate VCS Weblate

Weblate Git VCS Weblate

Weblate's REST API

2.6: The REST API is available since Weblate 2.6.

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

Authentication and generic parameters

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

ANY /

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

format -- Response format (overrides `Accept`). Possible values depends on REST framework setup, by default `json` and `api` are supported. The latter provides web browser interface for API.

Request Headers

`Accept` -- the response content type depends on `Accept` header

`Authorization` -- optional token to authenticate

Response Headers

`Content-Type` -- this depends on `Accept` header of request

`Allow` -- list of allowed HTTP methods on object

Response JSON Object

detail (*string*) -- verbose description of failure (for HTTP status codes other than 200 OK)

count (*int*) -- total item count for object lists

next (*string*) -- next page URL for object lists

previous (*string*) -- previous page URL for object lists

results (*array*) -- results for object lists

url (*string*) -- URL to access this resource using API

web_url (*string*) -- URL to access this resource using web browser

Status Codes

200 OK -- when request was correctly handled

400 Bad Request -- when form parameters are missing

403 Forbidden -- when access is denied

429 Too Many Requests -- when throttling is in place

Authentication examples

Example request:

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

Example response:

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

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

CURL example:

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

Passing Parameters Examples

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

Form request example:

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

operation=pull
```

JSON request example:

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

?????

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

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

The status of rate limiting is reported in following headers:

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

????? 4.1 ????: Added ratelimiting status headers.

API Entry Point

GET /api/

The API root entry point.

Example request:

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

Example response:

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

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

????

???? 4.0 ??

GET /api/users/

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

??:

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

POST /api/users/

Creates a new user.

Parameters

username (*string*) -- ?????

full_name (*string*) -- User full name

email (*string*) -- User email

is_superuser (*boolean*) -- Is user superuser? (optional)

is_active (*boolean*) -- Is user active? (optional)

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

Returns information about users.

Parameters

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

Response JSON Object

username (*string*) -- username of a user

full_name (*string*) -- full name of a user

email (*string*) -- email of a user

is_superuser (*boolean*) -- whether the user is a super user

is_active (*boolean*) -- whether the user is active

date_joined (*string*) -- date the user is created

groups (*array*) -- link to associated groups; see *GET /api/groups/(int:id)/*

Example JSON data:

```

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

```

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

Changes the user parameters.

Parameters

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

Response JSON Object

username (*string*) -- username of a user

full_name (*string*) -- full name of a user

email (*string*) -- email of a user

is_superuser (*boolean*) -- whether the user is a super user

is_active (*boolean*) -- whether the user is active

date_joined (*string*) -- date the user is created

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

Changes the user parameters.

Parameters

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

Response JSON Object

username (*string*) -- username of a user

full_name (*string*) -- full name of a user

email (*string*) -- email of a user

is_superuser (*boolean*) -- whether the user is a super user

is_active (*boolean*) -- whether the user is active

date_joined (*string*) -- date the user is created

DELETE /api/users/ (str: username) /

Deletes all user information and marks the user inactive.

Parameters

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

POST /api/users/ (str: username) /groups/

Associate groups with a user.

Parameters

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

Form Parameters

string group_id -- The unique group ID

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

List statistics of a user.

Parameters

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

Response JSON Object

translated (*int*) -- ??????????????

suggested (*int*) -- ??????????????

uploaded (*int*) -- ??????????????????

commented (*int*) -- ??????????????

languages (*int*) -- ??????????????????

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

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

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

username (*string*) -- User's username
Request JSON Object

notification (*string*) -- Name of notification registered

scope (*int*) -- Scope of notification from the available choices

frequency (*int*) -- Frequency choices for notifications

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

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

subscription_id (*int*) -- ??????? ID

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

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

subscription_id (*int*) -- ??????? ID
Request JSON Object

notification (*string*) -- Name of notification registered

scope (*int*) -- Scope of notification from the available choices

frequency (*int*) -- Frequency choices for notifications

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

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

subscription_id (*int*) -- ??????? ID
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

????

???? 4.0 ???

GET /api/groups/

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

??:

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

POST /api/groups/

Creates a new group.

Parameters

name (*string*) -- ?????

project_selection (*int*) -- Group of project selection from given options

language_selection (*int*) -- Group of languages selected from given options

GET /api/groups/(int: id) /

Returns information about group.

Parameters

id (*int*) -- Group's ID

Response JSON Object

name (*string*) -- name of a group

project_selection (*int*) -- integer corresponding to group of projects

language_selection (*int*) -- integer corresponding to group of languages

roles (*array*) -- link to associated roles; see *GET /api/roles/(int:id)/*

projects (*array*) -- link to associated projects; see *GET /api/projects/(string:project)/*

components (*array*) -- link to associated components; see *GET /api/components/(string:project)/(string:component)/*

componentlist (*array*) -- ?????? ??????????????????; ?? *GET /api/component-lists/(str:slug)/*

Example JSON data:

```

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

```

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

Changes the group parameters.

Parameters

id (*int*) -- Group's ID

Response JSON Object

name (*string*) -- name of a group

project_selection (*int*) -- integer corresponding to group of projects

language_selection (*int*) -- integer corresponding to group of Languages

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

id (*int*) -- Group's ID
 Response JSON Object

name (*string*) -- name of a group

project_selection (*int*) -- integer corresponding to group of projects

language_selection (*int*) -- integer corresponding to group of languages

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

id (*int*) -- Group's ID

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

id (*int*) -- Group's ID
 Form Parameters

string role_id -- The unique role ID

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

id (*int*) -- Group's ID
 Form Parameters

string component_id -- The unique component ID

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

id (*int*) -- Group's ID

component_id (*int*) -- The unique component ID

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

id (*int*) -- Group's ID
 Form Parameters

string project_id -- The unique project ID

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

id (*int*) -- Group's ID

project_id (*int*) -- The unique project ID

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

id (*int*) -- Group's ID
 Form Parameters

string language_code -- The unique language code

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

id (*int*) -- Group's ID

language_code (*string*) -- The unique language code

POST /api/groups/ (int: id) /componentlists/
 ????????? ?????????????????????????????????

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

??

GET /api/roles/

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

??:

Roles object attributes are documented at [GET /api/roles/\(int:id\)/](#).

POST /api/roles/

Creates a new role.

Parameters

name (*string*) -- Role name

permissions (*array*) -- List of codenames of permissions

GET /api/roles/(int: id) /

Returns information about a role.

Parameters

id (*int*) -- Role ID

Response JSON Object

name (*string*) -- Role name

permissions (*array*) -- list of codenames of permissions

Example JSON data:

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

PUT /api/roles/(int: id) /

Changes the role parameters.

Parameters

id (*int*) -- Role's ID

Response JSON Object

name (*string*) -- Role name

permissions (*array*) -- list of codenames of permissions

PATCH /api/roles/(int: id) /

Changes the role parameters.

Parameters

id (*int*) -- Role's ID

Response JSON Object

name (*string*) -- Role name

permissions (*array*) -- list of codenames of permissions

DELETE /api/roles/(int: id) /

Deletes the role.

Parameters

id (*int*) -- Role's ID

??

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

??:

Language object attributes are documented at [GET /api/languages/\(string:language\)/](#).

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

code (*string*) -- ???

name (*string*) -- ???

direction (*string*) -- Language direction

plural (*object*) -- Language plural formula and number

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

language (*string*) -- ?????

Response JSON Object

code (*string*) -- ?????

direction (*string*) -- ?????

plural (*object*) -- Object of language plural information

aliases (*array*) -- Array of aliases for language

Example JSON data:

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

PUT /api/languages/(string: language) /
Changes the language parameters.
Parameters

language (*string*) -- Language's code
Request JSON Object

name (*string*) -- ???

direction (*string*) -- Language direction

plural (*object*) -- Language plural details

PATCH /api/languages/(string: language) /
Changes the language parameters.

Parameters

language (*string*) -- Language's code
Request JSON Object

name (*string*) -- `???`

direction (*string*) -- Language direction

plural (*object*) -- Language plural details

DELETE /api/languages/ (string: language) /
Deletes the Language.
Parameters

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

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

language (*string*) -- `?????`
Response JSON Object

total (*int*) -- total number of strings

total_words (*int*) -- total number of words

last_change (*timestamp*) -- last changes in the language

recent_changes (*int*) -- total number of changes

translated (*int*) -- number of translated strings

translated_percent (*float*) -- percentage of translated strings

translated_words (*int*) -- number of translated words

translated_words_percent (*int*) -- percentage of translated words

translated_chars (*int*) -- number of translated characters

translated_chars_percent (*int*) -- percentage of translated characters

total_chars (*int*) -- number of total characters

fuzzy (*int*) -- number of fuzzy strings

fuzzy_percent (*int*) -- percentage of fuzzy strings

failing (*int*) -- number of failing strings

failing -- percentage of failing strings

`??????`

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

`??`:

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

POST /api/projects/
`?????? 3.9 ???`.

Creates a new project.
Parameters

name (*string*) -- `????????`

slug (*string*) -- Project slug

web (*string*) -- `????????` Web `???`

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

project (*string*) -- `????????` URL `?????`
Response JSON Object

name (*string*) -- project name

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/`

Example JSON data:

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

PATCH /api/projects/(string: project) /
4.3

Edit a project by a patch request.
Parameters

project (*string*) -- URL

component (*string*) -- URL

PUT /api/projects/(string: project) /
4.3

Edit a project by a put request.
Parameters

project (*string*) -- URL

DELETE /api/projects/(string: project) /
3.9

Deletes a project.
Parameters

project (*string*) -- URL

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*) -- URL

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*) -- URL

Response JSON Object

needs_commit (*boolean*) -- whether there are any pending changes to commit

needs_merge (*boolean*) -- whether there are any upstream changes to merge

needs_push (*boolean*) -- whether there are any local changes to push

Example JSON data:

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

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

Performs given operation on the VCS repository.

Parameters

project (*string*) -- `???????` URL `?????`

Request JSON Object

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

Response JSON Object

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

CURL example:

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

JSON request example:

```
POST /api/projects/hello/repository/ HTTP/1.1
```

```
Host: example.com
```

```
Accept: application/json
```

```
Content-Type: application/json
```

```
Authorization: Token TOKEN
```

```
Content-Length: 20
```

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

JSON response example:

```
HTTP/1.0 200 OK
```

```
Date: Tue, 12 Apr 2016 09:32:50 GMT
```

```
Server: WSGIServer/0.1 Python/2.7.11+
```

```
Vary: Accept, Accept-Language, Cookie
```

```
X-Frame-Options: SAMEORIGIN
```

```
Content-Type: application/json
```

```
Content-Language: en
```

```
Allow: GET, POST, HEAD, OPTIONS
```

```
{"result": true}
```

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

Returns a list of translation components in the given project.

Parameters

project (*string*) -- `???????` URL `?????`

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 `?????`.

`???????` 4.3 `?????`: The zipfile and docfile parameters are now accepted for VCS less components, see `??????` `??????`.

Creates translation components in the given project.

Parameters

project (*string*) -- `???????` URL `?????`

Request JSON Object

zipfile (*file*) -- ZIP file to upload into Weblate for translations initialization

docfile (*file*) -- `????????????????`

Response JSON Object

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

CURL example:

```

curl \
  --data-binary '{
    "branch": "master",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "Weblate",
    "slug": "weblate",
    "repo": "file:///home/nijel/work/weblate-hello",
    "template": "",
    "new_base": "",
    "vcs": "git"
  }' \
  -H "Content-Type: application/json" \
  -H "Authorization: Token TOKEN" \
  http://example.com/api/projects/hello/components/

```

JSON request example:

```

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

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

```

JSON response example:

```

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

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

```

(?)(?)(?)(?)(?)(?)(?)

```

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

```

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

XXXXXXXX 3.8 XXX.

Parameters

project (*string*) -- XXXXXXXX URL XXXX

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.

XXXXXXXX 3.8 XXX.

Parameters

project (*string*) -- XXXXXXXX URL XXXX

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

???????

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

??:

Component object attributes are documented at *GET /api/components/(string:project)/(string:component)/*.

GET /api/components/(string: project) /
string: *component* / Returns information about translation component.
Parameters

project (*string*) -- ??????? URL ????

component (*string*) -- ??????? URL ????

project (*object*) -- the translation project; see *GET /api/projects/(string:project)/*

name (*string*) -- ??????????

slug (*string*) -- *Component slug*

vcs (*string*) -- ??????????????

repo (*string*) -- ??????????????

git_export (*string*) -- ?????????????????? URL

branch (*string*) -- ??????????

push_branch (*string*) -- ?????? *push*

filemask (*string*) -- *File mask*

template (*string*) -- ??????????????????

edit_template (*string*) -- ??????????

intermediate (*string*) -- ??????????

new_base (*string*) -- ??????????????

file_format (*string*) -- ????????

license (*string*) -- ??????????

agreement (*string*) -- ??????????

new_lang (*string*) -- ??????????

language_code_style (*string*) -- ?????? ?????

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

push (*string*) -- ?????????????? URL

check_flags (*string*) -- ???????

priority (*string*) -- ??????

enforced_checks (*string*) -- ???????

restricted (*string*) -- *Restricted access*

repoweb (*string*) -- ?????? ??????

report_source_bugs (*string*) -- ??????????????????

merge_style (*string*) -- ??????????

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*) -- ??????????????

enable_suggestions (*string*) -- ??????????

suggestion_voting (*string*) -- ??????????

suggestion_autoaccept (*string*) -- [????????](#)
push_on_commit (*string*) -- [????????????????](#)
commit_pending_age (*string*) -- [????????????????](#)
auto_lock_error (*string*) -- [????????](#)
language_regex (*string*) -- [??????](#)
variant_regex (*string*) -- [????????](#)
repository_url (*string*) -- URL to repository status; see `GET /api/components/(string:project)/(string:component)/repository/`
translations_url (*string*) -- URL to translations list; see `GET /api/components/(string:project)/(string:component)/translations/`
lock_url (*string*) -- URL to lock status; see `GET /api/components/(string:project)/(string:component)/lock/`
changes_list_url (*string*) -- URL to changes list; see `GET /api/components/(string:project)/(string:component)/changes/`

Example JSON data:

```

{
  "branch": "master",
  "file_format": "po",
  "filemask": "po/*.po",
  "git_export": "",
  "license": "",
  "license_url": "",
  "name": "Weblate",
  "slug": "weblate",
  "project": {
    "name": "Hello",
    "slug": "hello",
    "source_language": {
      "code": "en",
      "direction": "ltr",
      "name": "English",
      "url": "http://example.com/api/languages/en/",
      "web_url": "http://example.com/languages/en/"
    },
    "url": "http://example.com/api/projects/hello/",
    "web": "https://weblate.org/",
    "web_url": "http://example.com/projects/hello/"
  },
  "source_language": {
    "code": "en",
    "direction": "ltr",
    "name": "English",
    "url": "http://example.com/api/languages/en/",
    "web_url": "http://example.com/languages/en/"
  },
  "repo": "file:///home/nijel/work/weblate-hello",
  "template": "",
  "new_base": "",
  "url": "http://example.com/api/components/hello/weblate/",
  "vcs": "git",
  "web_url": "http://example.com/projects/hello/weblate/"
}

```

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

project (*string*) -- [??????](#) URL [????](#)
component (*string*) -- [??????](#) URL [????](#)
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
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{
  "branch": "master",
  "file_format": "po",
  "filemask": "po/*.po",
  "git_export": "",
  "license": "",
  "license_url": "",
  "name": "new name",
  "slug": "weblate",
  "project": {
    "name": "Hello",
    "slug": "hello",
    "source_language": {
      "code": "en",
      "direction": "ltr",
      "name": "English",
      "url": "http://example.com/api/languages/en/",
      "web_url": "http://example.com/languages/en/"
    },
    "url": "http://example.com/api/projects/hello/",
    "web": "https://weblate.org/",
    "web_url": "http://example.com/projects/hello/"
  },
  "repo": "file:///home/nijel/work/weblate-hello",
  "template": "",
  "new_base": "",
  "url": "http://example.com/api/components/hello/weblate/",
  "vcs": "git",
  "web_url": "http://example.com/projects/hello/weblate/"
}
```

PUT /api/components/ (*string*: *project*) /
string: *component* / Edit a component by a put request.
Parameters

project (*string*) -- `??????` URL `?????`

component (*string*) -- `???????? URL ?????`
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 ????.*

Deletes a component.
Parameters

project (*string*) -- `???????? URL ?????`
component (*string*) -- `???????? URL ?????`

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*) -- `???????? URL ?????`
component (*string*) -- `???????? URL ?????`
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*) -- `???????? URL ?????`
component (*string*) -- `???????? URL ?????`
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*) -- `???????? URL ?????`
component (*string*) -- `???????? URL ?????`
Response JSON Object

locked (*boolean*) -- whether component is locked for updates

Example JSON data:

```
{
  "locked": false
}
```

POST /api/components/ (string: project) /
string: *component/lock/* Sets component lock status.

Response is same as `GET /api/components/(string:project)/(string:component)/lock/`.
Parameters

project (*string*) -- `???????? URL ?????`
component (*string*) -- `???????? URL ?????`
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*) -- *???????* URL *?????*

component (*string*) -- *???????* URL *?????*

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*) -- *???????* URL *?????*

component (*string*) -- *???????* URL *?????*

Request JSON Object

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

Response JSON Object

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

CURL example:

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

JSON request example:

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

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

JSON response example:

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

{"result": true}
```

GET /api/components/(string: project) /
string: *component/monolingual_base/* Downloads base file for monolingual translations.
Parameters

project (*string*) -- *???????* URL *?????*

component (*string*) -- *?????????* URL *?????*

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

project (*string*) -- *?????????* URL *?????*

component (*string*) -- *?????????* URL *?????*

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

project (*string*) -- *?????????* URL *?????*

component (*string*) -- *?????????* URL *?????*

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*) -- *?????????* URL *?????*

component (*string*) -- *?????????* URL *?????*

Request JSON Object

language_code (*string*) -- translation language code; see *GET /api/languages/(string:language)/*

Response JSON Object

result (*object*) -- new translation object created

CURL example:

```
curl \
  -d language_code=cs \
  -H "Authorization: Token TOKEN" \
  http://example.com/api/projects/hello/components/
```

JSON request example:

```
POST /api/projects/hello/components/ HTTP/1.1
```

```
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20
```

```
{"language_code": "cs"}
```

JSON response example:

```
HTTP/1.0 200 OK
```

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

```
{
  "failing_checks": 0,
  "failing_checks_percent": 0,
  "failing_checks_words": 0,
  "filename": "po/cs.po",
  "fuzzy": 0,
  "fuzzy_percent": 0.0,
  "fuzzy_words": 0,
  "have_comment": 0,
  "have_suggestion": 0,
  "is_template": false,
  "is_source": false,
  "language": {
    "code": "cs",
    "direction": "ltr",
    "name": "Czech",
    "url": "http://example.com/api/languages/cs/",
    "web_url": "http://example.com/languages/cs/"
  },
  "language_code": "cs",
  "id": 125,
  "last_author": null,
  "last_change": null,
  "share_url": "http://example.com/engage/hello/cs/",
  "total": 4,
  "total_words": 15,
  "translate_url": "http://example.com/translate/hello/weblate/cs/",
  "translated": 0,
  "translated_percent": 0.0,
  "translated_words": 0,
  "url": "http://example.com/api/translations/hello/weblate/cs/",
  "web_url": "http://example.com/projects/hello/weblate/cs/"
}
```

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

string: component/statistics/ Returns paginated statistics for all translations within component.

2.7.

Parameters

project (string) -- URL

component (string) -- URL

Response JSON Object

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

??

GET /api/translations/

Returns a list of translations.

??:

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*) -- `?????? URL ?????`

component (*string*) -- `???????? URL ?????`

language (*string*) -- Translation language code

Response JSON Object

component (*object*) -- component object; see `GET /api/components/(string:project)/(string:component)/`

failing_checks (*int*) -- `???????????????`

failing_checks_percent (*float*) -- `?????????????????`

failing_checks_words (*int*) -- `???????????????`

filename (*string*) -- translation filename

fuzzy (*int*) -- number of strings marked for review

fuzzy_percent (*float*) -- percentage of strings marked for review

fuzzy_words (*int*) -- number of words marked for review

have_comment (*int*) -- number of strings with comment

have_suggestion (*int*) -- number of strings with suggestion

is_template (*boolean*) -- `?????????????????`

language (*object*) -- source language object; see `GET /api/languages/(string:language)/`

language_code (*string*) -- language code used in the repository; this can be different from language code in the language object

last_author (*string*) -- name of last author

last_change (*timestamp*) -- last change timestamp

revision (*string*) -- revision hash for the file

share_url (*string*) -- URL for sharing leading to engagement page

total (*int*) -- total number of strings

total_words (*int*) -- total number of words

translate_url (*string*) -- URL for translating

translated (*int*) -- number of translated strings

translated_percent (*float*) -- percentage of translated strings

translated_words (*int*) -- number of translated words

repository_url (*string*) -- URL to repository status; see `GET /api/translations/(string:project)/(string:component)/(string:language)/repository/`

file_url (*string*) -- URL to file object; see `GET /api/translations/(string:project)/(string:component)/(string:language)/file/`

changes_list_url (*string*) -- URL to changes list; see `GET /api/translations/(string:project)/(string:component)/(string:language)/changes/`

units_list_url (*string*) -- URL to strings list; see `GET /api/translations/(string:project)/(string:component)/(string:language)/units/`

Example JSON data:

```

{
  "component": {
    "branch": "master",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
    "license": "",
    "license_url": "",
    "name": "Weblate",
    "new_base": "",
    "project": {
      "name": "Hello",
      "slug": "hello",
      "source_language": {
        "code": "en",
        "direction": "ltr",
        "name": "English",
        "url": "http://example.com/api/languages/en/",
        "web_url": "http://example.com/languages/en/"
      },
      "url": "http://example.com/api/projects/hello/",
      "web": "https://weblate.org/",
      "web_url": "http://example.com/projects/hello/"
    },
    "repo": "file:///home/nijel/work/weblate-hello",
    "slug": "weblate",
    "template": "",
    "url": "http://example.com/api/components/hello/weblate/",
    "vcs": "git",
    "web_url": "http://example.com/projects/hello/weblate/"
  },
  "failing_checks": 3,
  "failing_checks_percent": 75.0,
  "failing_checks_words": 11,
  "filename": "po/cs.po",
  "fuzzy": 0,
  "fuzzy_percent": 0.0,
  "fuzzy_words": 0,
  "have_comment": 0,
  "have_suggestion": 0,
  "is_template": false,
  "language": {
    "code": "cs",
    "direction": "ltr",
    "name": "Czech",
    "url": "http://example.com/api/languages/cs/",
    "web_url": "http://example.com/languages/cs/"
  },
  "language_code": "cs",
  "last_author": "Weblate Admin",
  "last_change": "2016-03-07T10:20:05.499",
  "revision": "7ddfafe6daaf57fc8654cc852ea6be212b015792",
  "share_url": "http://example.com/engage/hello/cs/",
  "total": 4,
  "total_words": 15,
  "translate_url": "http://example.com/translate/hello/weblate/cs/",
  "translated": 4,
  "translated_percent": 100.0,
  "translated_words": 15,
  "url": "http://example.com/api/translations/hello/weblate/cs/",
  "web_url": "http://example.com/projects/hello/weblate/cs/"
}

```

DELETE /api/translations/(string: *project*)/
string: *component*/string: *language*/ ????? 3.9 ???.

Deletes a translation.

Parameters

project (*string*)-- ??????? URL ?????

component (*string*) -- `????????? URL ?????`

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*) -- `????????? URL ?????`

component (*string*) -- `????????? URL ?????`

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*) -- `????????? URL ?????`

component (*string*) -- `????????? URL ?????`

language (*string*) -- Translation language code

q (*string*) -- Search query string `??` (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*) -- `????????? URL ?????`

component (*string*) -- `????????? URL ?????`

language (*string*) -- Translation language code
Request JSON Object

key (*string*) -- Name of translation unit

value (*string*) -- The translation unit value

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

project (*string*) -- `????????? URL ?????`

component (*string*) -- `????????? URL ?????`

language (*string*) -- Translation language code
Request JSON Object

mode (*string*) -- `?????????`

filter_type (*string*) -- Automatic translation filter type

auto_source (*string*) -- `??????`

component (*string*) -- `??`

engines (*string*) -- `???????????`

threshold (*string*) -- `???????????`

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

??: 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, xliiff, xliiff11, tbx, csv, xlsx, json, aresource, strings
Parameters

project (*string*) -- `????????? URL ?????`

component (*string*) -- [URL](#)

language (*string*) -- Translation language code

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

string: *component/string: language/file/* Upload new file with translations.

Parameters

project (*string*) -- [URL](#)

component (*string*) -- [URL](#)

language (*string*) -- Translation language code

Form Parameters

string conflicts -- How to deal with conflicts (ignore, replace-translated or replace-approved)

file file -- Uploaded file

string email -- [URL](#)

string author -- [URL](#)

string method -- [URL](#) (translate?approve?suggest?fuzzy?replace, source)? see [URL](#)

string fuzzy -- Fuzzy strings processing (*empty*, process, approve)

CURL example:

```
curl -X POST \  
  -F file=@strings.xml \  
  -H "Authorization: Token TOKEN" \  
  http://example.com/api/translations/hello/android/cs/file/
```

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

string: *component/string: language/repository/* Returns information about VCS repository status.

The response is same as for *GET /api/components/(string:project)/(string:component)/repository/*.

Parameters

project (*string*) -- [URL](#)

component (*string*) -- [URL](#)

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*) -- [URL](#)

component (*string*) -- [URL](#)

language (*string*) -- Translation language code

Request JSON Object

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

Response JSON Object

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

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

string: *component/string: language/statistics/* Returns detailed translation statistics.

[URL](#) 2.7 [URL](#).

Parameters

project (*string*) -- [URL](#)

component (*string*) -- [URL](#)

language (*string*) -- Translation language code

Response JSON Object

code (*string*) -- language code

failing (*int*) -- number of failing checks

failing_percent (*float*) -- percentage of failing checks

fuzzy (*int*) -- number of strings needing review
fuzzy_percent (*float*) -- percentage of strings needing review
total_words (*int*) -- total number of words
translated_words (*int*) -- number of translated words
last_author (*string*) -- name of last author
last_change (*timestamp*) -- date of last change
name (*string*) -- language name
total (*int*) -- total number of strings
translated (*int*) -- number of translated strings
translated_percent (*float*) -- percentage of translated strings
url (*string*) -- URL to access the translation (engagement URL)
url_translate (*string*) -- URL to access the translation (real translation URL)

Units

2.10

GET /api/units/

Returns list of translation units.

??:

Unit object attributes are documented at [GET /api/units/\(int:id\)/](#).

GET /api/units/(int: id) /

4.3: The `target` and `source` are now arrays to properly handle plural strings.

Returns information about translation unit.

Parameters

id (*int*) -- Unit ID

Response JSON Object

translation (*string*) -- URL of a related translation object

source (*array*) -- source string

previous_source (*string*) -- previous source string used for fuzzy matching

target (*array*) -- target string

id_hash (*string*) -- unique identifier of the unit

content_hash (*string*) -- unique identifier of the source string

location (*string*) -- location of the unit in source code

context (*string*) -- translation unit context

note (*string*) -- translation unit note

flags (*string*) -- translation unit flags

state (*int*) -- unit state, 0 - not translated, 10 - needs editing, 20 - translated, 30 - approved, 100 - read only

fuzzy (*boolean*) --

translated (*boolean*) --

approved (*boolean*) --

position (*int*) -- unit position in translation file

has_suggestion (*boolean*) --

has_comment (*boolean*) --

has_failing_check (*boolean*) --

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*) -- [redacted]

web_url (*string*) -- URL where the unit can be edited

source_unit (*string*) -- Source unit link; see *GET /api/units/(int:id)/*

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

[redacted] 4.3 [redacted].

[redacted]

Parameters

id (*int*) -- Unit ID

Request JSON Object

state (*int*) -- unit state, 0 - not translated, 10 - needs editing, 20 - translated, 30 - approved, 100 - read only

target (*array*) -- target string

explanation (*string*) -- String explanation, available on source units, see *Additional info on source strings*

extra_flags (*string*) -- [redacted]

PUT /api/units/(int: id) /

[redacted] 4.3 [redacted].

[redacted]

Parameters

id (*int*) -- Unit ID

Request JSON Object

state (*int*) -- unit state, 0 - not translated, 10 - needs editing, 20 - translated, 30 - approved, 100 - read only

target (*array*) -- target string

explanation (*string*) -- String explanation, available on source units, see *Additional info on source strings*

extra_flags (*string*) -- [redacted]

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

[redacted] 4.3 [redacted].

[redacted]

Parameters

id (*int*) -- Unit ID

[redacted]

[redacted] 2.10 [redacted].

GET /api/changes/

[redacted] 4.1 [redacted]: [redacted] 4.1 [redacted]

Returns a list of translation changes.

[redacted]:

Change object attributes are documented at *GET /api/changes/(int:id)/*.

Query Parameters

user (*string*) -- Username of user to filters

action (*int*) -- Action to filter, can be used several times

timestamp_after (*timestamp*) -- ISO 8601 formatted timestamp to list changes after

timestamp_before (*timestamp*) -- ISO 8601 formatted timestamp to list changes before

GET /api/changes/(int: id) /

Returns information about translation change.

Parameters

id (*int*) -- Change ID

Response JSON Object

unit (*string*) -- URL of a related unit object

translation (*string*) -- URL of a related translation object

component (*string*) -- URL of a related component object

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

user (*string*) -- URL of a related user object
author (*string*) -- URL of a related author object
timestamp (*timestamp*) -- event timestamp
action (*int*) -- numeric identification of action
action_name (*string*) -- text description of action
target (*string*) -- event changed text or detail
id (*int*) -- change identifier

????????

???? 2.14 ??

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

??:

Screenshot object attributes are documented at [GET /api/screenshots/\(int:id\)/](#).

GET /api/screenshots/(int: id)/
Returns information about screenshot information.
Parameters

id (*int*) -- Screenshot ID
Response JSON Object

name (*string*) -- name of a screenshot

component (*string*) -- URL of a related component object

file_url (*string*) -- URL to download a file; see [GET /api/screenshots/\(int:id\)/file/](#)

units (*array*) -- link to associated source string information; see [GET /api/units/\(int:id\)/](#)

GET /api/screenshots/(int: id)/file/
Download the screenshot image.
Parameters

id (*int*) -- Screenshot ID

POST /api/screenshots/(int: id)/file/
Replace screenshot image.
Parameters

id (*int*) -- Screenshot ID
Form Parameters

file image -- Uploaded file

CURL example:

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

POST /api/screenshots/(int: id)/units/
Associate source string with screenshot.
Parameters

id (*int*) -- Screenshot ID
Form Parameters

string unit_id -- Unit ID
Response JSON Object

name (*string*) -- name of a screenshot

translation (*string*) -- URL of a related translation object

file_url (*string*) -- URL to download a file; see [GET /api/screenshots/\(int:id\)/file/](#)

units (*array*) -- link to associated source string information; see [GET /api/units/\(int:id\)/](#)

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

Parameters

id (*int*) -- Screenshot ID

unit_id -- ID

POST /api/screenshots/

Creates a new screenshot.

Form Parameters

file image -- Uploaded file

string name --

string project_slug -- Project slug

string component_slug -- Component slug

string language_code --

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) /

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) /

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) /

Parameters

id (*int*) -- Screenshot ID

4.0

GET /api/component-lists/

??:

GET /api/component-lists/(str:slug) /

GET /api/component-lists/(str: slug) /

Returns information about component list.

Parameters

slug (*string*) -- Component list slug

Response JSON Object

name (*string*) -- name of a component list

slug (*string*) -- slug of a component list

show_dashboard (*boolean*) -- whether to show it on a dashboard

components (*array*) -- link to associated components; see `GET /api/components/(string:project)/(string:component)/`

auto_assign (*array*) -- automatic assignment rules

PUT /api/component-lists/(str: slug) /
Changes the component list parameters.
Parameters

slug (*string*) -- Component list slug
Request JSON Object

name (*string*) -- name of a component list

slug (*string*) -- slug of a component list

show_dashboard (*boolean*) -- whether to show it on a dashboard

PATCH /api/component-lists/(str: slug) /
Changes the component list parameters.
Parameters

slug (*string*) -- Component list slug
Request JSON Object

name (*string*) -- name of a component list

slug (*string*) -- slug of a component list

show_dashboard (*boolean*) -- whether to show it on a dashboard

DELETE /api/component-lists/(str: slug) /
Deletes the component list.
Parameters

slug (*string*) -- Component list slug

POST /api/component-lists/(str: slug) /components/
Associate component with a component list.
Parameters

slug (*string*) -- Component list slug
Form Parameters

string component_id -- Component ID

DELETE /api/component-lists/(str: slug) /components/str: component_slug Disassociate a component from the component list.
Parameters

slug (*string*) -- Component list slug

component_slug (*string*) -- Component slug

???

GET /api/glossary/
Returns a list of all glossaries which are associated with a project that user has access to.

??:

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

GET /api/glossary/(int: id) /
????????????????
Parameters

id (*int*) -- ??? id
Response JSON Object

name (*string*) -- ?????

color (*string*) -- ???????

source_language (*object*) -- Object of language plural information

projects (*array*) -- link to associated projects; see `GET /api/projects/(string:project)/`

Example JSON data:

```

{
  "name": "Hello",
  "id": 1,
  "color": "silver",
  "source_language": {
    "code": "en",
    "name": "English",
    "plural": {
      "id": 75,
      "source": 0,
      "number": 2,
      "formula": "n != 1",
      "type": 1
    },
    "aliases": [
      "english",
      "en_en",
      "base",
      "source",
      "eng"
    ],
    "direction": "ltr",
    "web_url": "http://example.com/languages/en/",
    "url": "http://example.com/api/languages/en/",
    "statistics_url": "http://example.com/api/languages/en/statistics/"
  },
  "project": {
    "name": "Hello",
    "slug": "hello",
    "id": 1,
    "source_language": {
      "code": "en",
      "name": "English",
      "plural": {
        "id": 75,
        "source": 0,
        "number": 2,
        "formula": "n != 1",
        "type": 1
      },
      "aliases": [
        "english",
        "en_en",
        "base",
        "source",
        "eng"
      ],
      "direction": "ltr",
      "web_url": "http://example.com/languages/en/",
      "url": "http://example.com/api/languages/en/",
      "statistics_url": "http://example.com/api/languages/en/
↵statistics/"
    },
    "web_url": "http://example.com/projects/demo1/",
    "url": "http://example.com/api/projects/demo1/",
    "components_list_url": "http://example.com/api/projects/demo1/
↵components/",
    "repository_url": "http://example.com/api/projects/demo1/
↵repository/",
    "statistics_url": "http://example.com/api/projects/demo1/
↵statistics/",
    "changes_list_url": "http://example.com/api/projects/demo1/changes/
↵",
    "languages_url": "http://example.com/api/projects/demo1/languages/"
  },
  "projects_url": "http://example.com/api/glossary/7/projects/",
  "terms_url": "http://example.com/api/glossary/7/terms/",
  "url": "http://example.com/api/glossary/7/"
}

```

```
}

```

PUT /api/glossary/ (int: id) /

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

Parameters

id (*int*) -- XXX id
Request JSON Object

name (*string*) -- XXX

color (*string*) -- Language direction

source_language (*object*) -- Language plural details

PATCH /api/glossary/ (int: id) /

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

Parameters

id (*int*) -- XXX id
Request JSON Object

name (*string*) -- XXX

color (*string*) -- Language direction

source_language (*object*) -- Language plural details

DELETE /api/glossary/ (int: id) /

XXXXXXXXXXXXXXXXXXXX

Parameters

id (*int*) -- XXX id

GET /api/glossary/ (int: id) /projects/

Returns projects linked with a glossary.

Parameters

id (*int*) -- XXX id
Response JSON Object

projects (*array*) -- XXXXXXXXXXXXXXX; XXX GET /api/projects/ (string:project) /

POST /api/glossary/ (int: id) /projects/

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

Parameters

id (*int*) -- XXX id
Form Parameters

string project_slug -- Project slug

DELETE /api/glossary/ (int: id) /projects/

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

Parameters

id (*int*) -- XXX id
Form Parameters

string project_slug -- Project slug

GET /api/glossary/ (int: id) /terms/

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

Parameters

id (*int*) -- XXX id

POST /api/glossary/ (int: id) /terms/

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

Parameters

id (*int*) -- XXX id
Request JSON Object

language (*object*) -- XXXXXX

source (*string*) -- XXXXXX

target (*string*) -- XXXXXXXXXXXXXXX

GET /api/glossary/ (int: id) /terms/

int: term_id / XXXXXXXXXXXXXXXXXXXXXXX

Parameters

id (*int*) -- `int` id

term_id (*int*) -- ID of term

PUT /api/glossary/(int: id)/terms/

int: *term_id* / `int`

Parameters

id (*int*) -- `int` id

term_id (*int*) -- ID of term

Request JSON Object

language (*object*) -- `object`

source (*string*) -- `string`

target (*string*) -- `string`

PATCH /api/glossary/(int: id)/terms/

int: *term_id* / `int`

Parameters

id (*int*) -- `int` id

term_id (*int*) -- ID of term

Request JSON Object

language (*object*) -- `object`

source (*string*) -- `string`

target (*string*) -- `string`

DELETE /api/glossary/(int: id)/terms/

int: *term_id* / Delete a term associated with a glossary.

Parameters

id (*int*) -- `int` id

term_id (*int*) -- ID of term

Webhooks

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* / `int` 2.6 `int`: 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) /

`int` 2.6 `int`: 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.

int: 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.

int:

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.

🔗:

Automatically receiving changes from GitLab

For instruction on setting up GitLab integration

<https://docs.gitlab.com/ce/user/project/integrations/webhooks.html>

Generic information about GitLab Webhooks

[ENABLE_HOOKS](#)

For enabling hooks for whole Weblate

POST /hooks/bitbucket/

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

🔗:

Automatically receiving changes from Bitbucket

For instruction on setting up Bitbucket integration

<https://confluence.atlassian.com/bitbucket/manage-webhooks-735643732.html>

Generic information about Bitbucket Webhooks

[ENABLE_HOOKS](#)

For enabling hooks for whole Weblate

POST /hooks/pagure/

🔗🔗🔗 3.3 🔗🔗.

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

🔗:

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 🔗🔗.

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

🔗:

Automatically receiving changes from Azure Repos

For instruction on setting up Azure integration

<https://docs.microsoft.com/en-us/azure/devops/service-hooks/services/webhooks>

Generic information about Azure Repos Web Hooks

[ENABLE_HOOKS](#)

For enabling hooks for whole Weblate

POST /hooks/gitea/

🔗🔗🔗 3.9 🔗🔗.

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

🔗:

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 🔗🔗.

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

🔗:

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

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: Please use `GET /api/components/(string:project)/(string:component)/statistics/` and `GET /api/translations/(string:project)/(string:component)/(string:language)/statistics/` instead; it allows access to ACL controlled projects as well.

Retrieves statistics for given component in given format.

Example request:

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

Example response:

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

[
  {
    "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/
↪master/cs/"
  },
  {
    "code": "nl",
    "failing": 21,
    "failing_percent": 4.8,
    "fuzzy": 11,
    "fuzzy_percent": 2.5,
    "last_author": null,
    "last_change": null,
    "name": "Dutch",
    "total": 436,
    "total_words": 15271,
    "translated": 319,
    "translated_percent": 73.2,
    "translated_words": 3201,
    "url": "http://hosted.weblate.org/engage/weblate/nl/",
    "url_translate": "http://hosted.weblate.org/projects/weblate/
↪master/nl/"
  },
  {
    "code": "el",
    "failing": 11,
    "failing_percent": 2.5,
    "fuzzy": 21,
    "fuzzy_percent": 4.8,
```



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

????

wlc [link](#) ~/.config/weblate [link](#) [link](#) for other locations [link](#):

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

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

[link](#) [link](#)

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

?:
[link](#)

??

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

[link](#)

??

Weblate [link](#) Python [link](#)API [link](#) Weblate [link](#)
[link](#)wlc [link](#)wlc [link](#)

??

```
link Weblate link
--format {csv,json,text,html}
link
--url URL
API link URL link linkURL link /api/ link https://hosted.weblate.org/api/link
--key KEY
link API link link link Weblate link
--config PATH
link
--config-section SECTION
link
```

????

????????????????:

version

????????????????

list-languages

Weblate ??????????????????

list-projects

Weblate ??????????????????

list-components

Weblate ??????????????????

list-translations

Weblate ??????????????

show

Weblate ???

ls

Weblate ???

commit

Weblate ???)??

pull

????????????????? Weblate ???

push

Weblate ?????????????????? ???

reset

?????? 0.7 ????: wlc 0.7 ?????

Weblate ???

cleanup

?????? 0.9 ????: wlc 0.9 ?????

????? ??? Weblate ???

repo

????? Weblate ???

statistics

????? Weblate ???

lock-status

?????? 0.5 ????: wlc 0.5 ?????

????????????????

lock

?????? 0.5 ????: wlc 0.5 ?????

Weblate ?????????????????????????????????????

unlock

?????? 0.5 ????: wlc 0.5 ?????

Weblate ?????????????????????????????

changes

?????? 0.7 ????: wlc 0.7 ??? Weblate 2.10 ?????

????????????????????

download

?????? 0.7 ????: wlc 0.7 ?????

????????????????

--convert

??

--output

??

upload

?????? 0.9 ????: wlc 0.9 ?????


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

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

```
????????????????????????????????????????????????????????????:
```

```
$ wlc commit
```

Weblate's Python API



The Python API is shipped separately, you need to install the *Weblate* : (wlc) to have it.

```
pip install wlc
```

wlc

WeblateException

exception wlc.WeblateException
Base class for all exceptions.

Weblate

class wlc.Weblate (key="", url=None, config=None)

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)

path (str) -- Request path

object

Performs a single API GET call.

post (path, **kwargs)

path (str) -- Request path

object

Performs a single API GET call.

wlc.config

WeblateConfig

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

section (str) -- Configuration section to use

Configuration file parser following XDG specification.

load (path=None)

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`).

wlc.main

wlc.main.main (settings=None, stdout=None, args=None)

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.

Configuration instructions

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 all 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.

??: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

??

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
```

??: 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 ??: 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 ??: 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.

??:

Invoking management commands

Docker container with HTTPS support

Please see **??** for generic deployment instructions, this section only mentions differences compared to it.

Using own SSL certificates

3.8-3

In case you have own SSL certificate you want to use, simply place the files into the Weblate data volume (see *Docker container volumes*):

ssl/fullchain.pem containing the certificate including any needed CA certificates

ssl/privkey.pem containing the private key

Both of these files must be owned by the same user as the one starting the docker container and have file mask set to 600 (readable and writable only by the owning user).

Additionally, Weblate container will now accept SSL connections on port 4443, you will want to include the port forwarding for HTTPS in docker compose override:

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

If you already host other sites on the same server, it is likely ports 80 and 443 are used by a reverse proxy, such as NGINX. To pass the HTTPS connection from NGINX to the docker container, you can use the following configuration:

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

    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 additional Docker container, `https-portal` will be used for that. This is made use of in the `docker-compose-https.yml` file. Then create a `docker-compose-https.override.yml` file with your settings:

```
version: '3'
services:
  weblate:
    environment:
      WEBLATE_EMAIL_HOST: smtp.example.com
      WEBLATE_EMAIL_HOST_USER: user
      WEBLATE_EMAIL_HOST_PASSWORD: pass
      WEBLATE_SITE_DOMAIN: weblate.example.com
      WEBLATE_ADMIN_PASSWORD: password for admin user
  https-portal:
    environment:
      DOMAINS: 'weblate.example.com -> http://weblate:8080'
```

Whenever invoking `docker-compose` you need to pass both files to it, and then do:

```
docker-compose -f docker-compose-https.yml -f docker-compose-https.  
↳override.yml build  
docker-compose -f docker-compose-https.yml -f docker-compose-https.  
↳override.yml up
```

Upgrading the Docker container

Usually it is good idea to only update the Weblate container and keep the PostgreSQL container at the version you have, as upgrading PostgreSQL is quite painful and in most cases does not bring many benefits.

You can do this by sticking with the existing docker-compose and just pull the latest images and then restart:

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

The Weblate database should be automatically migrated on first startup, and there should be no need for additional manual actions.

??? Upgrades across 3.0 are not supported by Weblate. If you are on 2.x series and want to upgrade to 3.x, first upgrade to the latest 3.0.1-x (at time of writing this it is the 3.0.1-7) image, which will do the migration and then continue upgrading to newer versions.

You might also want to update the `docker-compose` repository, though it's not needed in most case. Please beware of PostgreSQL version changes in this case as it's not straightforward to upgrade the database, see [GitHub issue](#) for more info.

Admin login

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.

???:

```
WEBLATE_ADMIN_PASSWORD?WEBLATE_ADMIN_NAME?WEBLATE_ADMIN_EMAIL
```

Docker environment variables

Many of Weblate's **???** 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
```

???:

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

```
????????????????
```

???: In case it is not set, the first item from `WEBLATE_ALLOWED_HOSTS` is used.

??:

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
```

??:

Admin login? 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`.

??: 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.

??:

Admin login? `WEBLATE_ADMIN_PASSWORD`? `WEBLATE_ADMIN_NAME`? `WEBLATE_ADMIN_EMAIL`

WEBLATE_SERVER_EMAIL

WEBLATE_DEFAULT_FROM_EMAIL

Configures the address for outgoing e-mails.

??:

Configure e-mail sending

WEBLATE_ALLOWED_HOSTS

Configures allowed HTTP hostnames using `ALLOWED_HOSTS`.

Defaults to `*` which allows all hostnames.

Example:

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

??:

`ALLOWED_HOSTS`? Allowed hosts setup? Set correct site domain

WEBLATE_REGISTRATION_OPEN

Configures whether registrations are open by toggling `REGISTRATION_OPEN`.

Example:

```
environment:
  WEBLATE_REGISTRATION_OPEN: 0
```

WEBLATE_REGISTRATION_ALLOW_BACKENDS

Configure which authentication methods can be used to create new account via `REGISTRATION_ALLOW_BACKENDS`.

Example:

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

WEBLATE_TIME_ZONE

Configures the used time zone in Weblate, see [TIME_ZONE](#).

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

Example:

```
environment:  
  WEBLATE_TIME_ZONE: Europe/Prague
```

WEBLATE_ENABLE_HTTPS

Makes Weblate assume it is operated behind a reverse HTTPS proxy, it makes Weblate use HTTPS in e-mail and API links or set secure flags on cookies.

??: 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
```

??:

Set correct site domain

WEBLATE_IP_PROXY_HEADER

Lets Weblate fetch the IP address from any given HTTP header. Use this when using a reverse proxy in front of the Weblate container.

Enables `IP_BEHIND_REVERSE_PROXY` and sets `IP_PROXY_HEADER`.

??: 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
```

??:

`SECURE_PROXY_SSL_HEADER`

WEBLATE_REQUIRE_LOGIN

Configures login required for the whole of the Weblate installation using [LOGIN_REQUIRED_URLS](#).

Example:

```
environment:  
  WEBLATE_REQUIRE_LOGIN: 1
```

WEBLATE_LOGIN_REQUIRED_URLS_EXCEPTIONS

WEBLATE_ADD_LOGIN_REQUIRED_URLS_EXCEPTIONS

WEBLATE_REMOVE_LOGIN_REQUIRED_URLS_EXCEPTIONS

Adds URL exceptions for login required for the whole Weblate installation using `LOGIN_REQUIRED_URLS_EXCEPTIONS`.

You can either replace whole settings, or modify default value using ADD and REMOVE variables.

WEBLATE_GOOGLE_ANALYTICS_ID

Configures ID for Google Analytics by changing `GOOGLE_ANALYTICS_ID`.

WEBLATE_GITHUB_USERNAME

Configures GitHub username for GitHub pull-requests by changing `GITHUB_USERNAME`.

??:

GitHub

WEBLATE_GITHUB_TOKEN

4.3

Configures GitHub personal access token for GitHub pull-requests via API by changing `GITHUB_TOKEN`.

??:

GitHub

WEBLATE_GITLAB_USERNAME

Configures GitLab username for GitLab merge-requests by changing `GITLAB_USERNAME`

??:

GitLab

WEBLATE_GITLAB_TOKEN

Configures GitLab personal access token for GitLab merge-requests via API by changing `GITLAB_TOKEN`

??:

GitLab

WEBLATE_SIMPLIFY_LANGUAGES

Configures the language simplification policy, see `SIMPLIFY_LANGUAGES`.

WEBLATE_DEFAULT_ACCESS_CONTROL

Configures the default `?` 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 `?` for new components, see `DEFAULT_TRANSLATION_PROPAGATION`.

WEBLATE_DEFAULT_COMMITER_EMAIL

Configures `DEFAULT_COMMITER_EMAIL`.

WEBLATE_DEFAULT_COMMITER_NAME

Configures `DEFAULT_COMMITER_NAME`.

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`.

??:

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.

??:

Content security policy?CSP_SCRIPT_SRC?CSP_IMG_SRC?CSP_CONNECT_SRC?CSP_STYLE_SRC?CSP_FONT_SRC

WEBLATE_LICENSE_FILTER

LICENSE_FILTER ?????

WEBLATE_HIDE_VERSION

Configures *HIDE_VERSION*.

Machine translation settings

WEBLATE_MT_APERTIUM_APY

Enables *Apertium* machine translation and sets *MT_APERTIUM_APY*

WEBLATE_MT_AWS_REGION

WEBLATE_MT_AWS_ACCESS_KEY_ID

WEBLATE_MT_AWS_SECRET_ACCESS_KEY

Configures *AWS* machine translation.

environment:

```
WEBLATE_MT_AWS_REGION: us-east-1
```

```
WEBLATE_MT_AWS_ACCESS_KEY_ID: AKIAIOSFODNN7EXAMPLE
```

```
WEBLATE_MT_AWS_SECRET_ACCESS_KEY: wJalrXUtnFEMI/K7MDENG/
```

```
↪bPxrFicYEXAMPLEKEY
```

WEBLATE_MT_DEEPL_KEY

Enables *DeepL* machine translation and sets *MT_DEEPL_KEY*

WEBLATE_MT_DEEPL_API_VERSION

???? *DeepL* API ?????? *MT_DEEPL_API_VERSION*?

WEBLATE_MT_GOOGLE_KEY

Enables *Google Translate* and sets *MT_GOOGLE_KEY*

WEBLATE_MT_MICROSOFT_COGNITIVE_KEY

Enables *Microsoft Cognitive Services Translator* and sets *MT_MICROSOFT_COGNITIVE_KEY*

WEBLATE_MT_MICROSOFT_ENDPOINT_URL

Sets *MT_MICROSOFT_ENDPOINT_URL*, please note this is supposed to contain domain name only.

WEBLATE_MT_MICROSOFT_REGION

Sets *MT_MICROSOFT_REGION*

WEBLATE_MT_MICROSOFT_BASE_URL

Sets *MT_MICROSOFT_BASE_URL*

WEBLATE_MT_MODERNMT_KEY

Enables *ModernMT* and sets *MT_MODERNMT_KEY*.

WEBLATE_MT_MYMEMORY_ENABLED

Enables *MyMemory* machine translation and sets *MT_MYMEMORY_EMAIL* to *WEBLATE_ADMIN_EMAIL*.

Example:

environment:

```
WEBLATE_MT_MYMEMORY_ENABLED: 1
```

WEBLATE_MT_GLOSBE_ENABLED

Enables *Glosbe* machine translation.

environment:

```
WEBLATE_MT_GLOSBE_ENABLED: 1
```

WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED

Enables *Microsoft Terminology Service* machine translation.

environment:

```
WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED: 1
```

WEBLATE_MT_SAP_BASE_URL

WEBLATE_MT_SAP_SANDBOX_APIKEY

WEBLATE_MT_SAP_USERNAME

WEBLATE_MT_SAP_PASSWORD

WEBLATE_MT_SAP_USE_MT

Configures *SAP Translation Hub* machine translation.

```
environment:
  WEBLATE_MT_SAP_BASE_URL: "https://example.hana.ondemand.com/
↳translationhub/api/v1/"
  WEBLATE_MT_SAP_USERNAME: "user"
  WEBLATE_MT_SAP_PASSWORD: "password"
  WEBLATE_MT_SAP_USE_MT: 1
```

Authentication settings

LDAP

WEBLATE_AUTH_LDAP_SERVER_URI

WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE

WEBLATE_AUTH_LDAP_USER_ATTR_MAP

WEBLATE_AUTH_LDAP_BIND_DN

WEBLATE_AUTH_LDAP_BIND_PASSWORD

WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS

WEBLATE_AUTH_LDAP_USER_SEARCH

WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER

WEBLATE_AUTH_LDAP_USER_SEARCH_UNION

WEBLATE_AUTH_LDAP_USER_SEARCH_UNION_DELIMITER

LDAP authentication configuration.

Example for direct bind:

```
environment:
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE: uid=%(user)s,ou=People,dc=example,
↳dc=net
  # map weblate 'full_name' to ldap 'name' and weblate 'email' attribute_
↳to 'mail' ldap attribute.
  # another example that can be used with OpenLDAP: 'full_name:cn,
↳email:mail'
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
```

Example for search and bind:

```
environment:
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_BIND_DN: CN=ldap,CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_BIND_PASSWORD: password
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
  WEBLATE_AUTH_LDAP_USER_SEARCH: CN=Users,DC=example,DC=com
```

Example for union search and bind:

```
environment:
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_BIND_DN: CN=ldap,CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_BIND_PASSWORD: password
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
  WEBLATE_AUTH_LDAP_USER_SEARCH_UNION: ou=users,dc=example,
↳dc=com|ou=otherusers,dc=example,dc=com
```

Example with search and bind against Active Directory:

environment :

```
WEBLATE_AUTH_LDAP_BIND_DN: CN=ldap,CN=Users,DC=example,DC=com
WEBLATE_AUTH_LDAP_BIND_PASSWORD: password
WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS: 0
WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
WEBLATE_AUTH_LDAP_USER_SEARCH: CN=Users,DC=example,DC=com
WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER: (sAMAccountName=%(user)s)
```

🔗:

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.

??:

Database setup for Weblate

POSTGRES_PASSWORD

PostgreSQL password.

POSTGRES_USER

PostgreSQL username.

POSTGRES_DATABASE

PostgreSQL database name.

POSTGRES_HOST

PostgreSQL server hostname or IP address. Defaults to `database`.

POSTGRES_PORT

PostgreSQL server port. Defaults to none (uses the default value).

POSTGRES_SSL_MODE

Configure how PostgreSQL handles SSL in connection to the server, for possible choices see [SSL Mode Descriptions](#)

POSTGRES_ALTER_ROLE

Configures name of role to alter during migrations, see *Configuring Weblate to use PostgreSQL*.

Database backup settings

??:

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.

??:

Enable caching

REDIS_HOST

The Redis server hostname or IP address. Defaults to `cache`.

REDIS_PORT

The Redis server port. Defaults to `6379`.

REDIS_DB

The Redis database number, defaults to `1`.

REDIS_PASSWORD

The Redis server password, not used by default.

REDIS_TLS

Enables using SSL for Redis connection.

REDIS_VERIFY_SSL

Can be used to disable SSL certificate verification for Redis connection.

Email server setup

To make outgoing e-mail work, you need to provide a mail server.

Example TLS configuration:

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

Example SSL configuration:

```
environment :
  WEBLATE_EMAIL_HOST: smtp.example.com
  WEBLATE_EMAIL_PORT: 465
  WEBLATE_EMAIL_HOST_USER: user
  WEBLATE_EMAIL_HOST_PASSWORD: pass
  WEBLATE_EMAIL_USE_TLS: 0
  WEBLATE_EMAIL_USE_SSL: 1
```

??:

Configuring outgoing e-mail

WEBLATE_EMAIL_HOST

Mail server hostname or IP address.

??:

WEBLATE_EMAIL_PORT?*WEBLATE_EMAIL_USE_SSL*?*WEBLATE_EMAIL_USE_TLS*?*EMAIL_HOST*

WEBLATE_EMAIL_PORT

Mail server port, defaults to 25.

??:

EMAIL_PORT

WEBLATE_EMAIL_HOST_USER

????????????

??:

EMAIL_HOST_USER

WEBLATE_EMAIL_HOST_PASSWORD

????????????

??:

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*.

??:

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*.

??:

WEBLATE_EMAIL_PORT?*WEBLATE_EMAIL_USE_SSL*?*EMAIL_USE_TLS*

WEBLATE_EMAIL_BACKEND

Configures Django back-end to use for sending e-mails.

??:

Configure e-mail sending?*EMAIL_BACKEND*

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).

CDN [\[?\]\[?\]\[?\]\[?\]](#)

WEBLATE_LOCALIZE_CDN_URL

WEBLATE_LOCALIZE_CDN_PATH

[\[?\]\[?\]\[?\]\[?\] 4.2.1 \[?\]\[?\]](#).

Configuration for *JavaScript [\[?\]\[?\]\[?\]\[?\] CDN](#)*.

The `WEBLATE_LOCALIZE_CDN_PATH` is path within the container. It should be stored on the persistent volume and not in the transient storage.

One of possibilities is storing that inside the Weblate data dir:

```
environment:
  WEBLATE_LOCALIZE_CDN_URL: https://cdn.example.com/
  WEBLATE_LOCALIZE_CDN_PATH: /app/data/l10n-cdn
```

[\[?\]\[?\]](#): You are responsible for setting up serving of the files generated by Weblate, it only does stores the files in configured location.

[\[?\]\[?\]](#):

Translating HTML and JavaScript using Weblate [CDN](#)`WEBLATE_LOCALIZE_CDN_URL``WEBLATE_LOCALIZE_CDN_PATH`

Changing enabled apps, checks, addons or autofixes

[\[?\]\[?\]\[?\]\[?\] 3.8-5 \[?\]\[?\]](#).

The built-in configuration of enabled checks, addons or autofixes can be adjusted by the following variables:

WEBLATE_ADD_APPS

WEBLATE_REMOVE_APPS

WEBLATE_ADD_CHECK

WEBLATE_REMOVE_CHECK

WEBLATE_ADD_AUTOFIX

WEBLATE_REMOVE_AUTOFIX

WEBLATE_ADD_ADDONS

WEBLATE_REMOVE_ADDONS

Example:

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

[\[?\]\[?\]](#):

`CHECK_LIST``AUTOFIX_LIST``WEBLATE_ADDONS``INSTALLED_APPS`

??????

CELERY_MAIN_OPTIONS

CELERY_NOTIFY_OPTIONS

CELERY_MEMORY_OPTIONS

CELERY_TRANSLATE_OPTIONS

CELERY_BACKUP_OPTIONS

CELERY_BEAT_OPTIONS

These variables allow you to adjust Celery worker options. It can be useful to adjust concurrency (`--concurrency 16`) or use different pool implementation (`--pool=gevent`).

By default, the number of concurrent workers matches the number of processors (except the backup worker, which is supposed to run only once).

Example:

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

??:

Celery worker options [Background tasks using Celery](#)

UWSGI_WORKERS

Configure how many uWSGI workers should be executed.

It defaults to number of processors + 1.

Example:

```
environment:
  UWSGI_WORKERS: 32
```

In case you have lot of CPU cores and hit out of memory issues, try reducing number of workers:

```
environment:
  UWSGI_WORKERS: 4
  CELERY_MAIN_OPTIONS: --concurrency 2
  CELERY_NOTIFY_OPTIONS: --concurrency 1
  CELERY_TRANSLATE_OPTIONS: --concurrency 1
```

Docker container volumes

There is single data volume 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/`. In the container it is mounted as `/app/data`.

??:

[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 after all environment settings are loaded, so it gets completely set up, and can be used to customize anything.

Replacing logo and other static files

3.8-5

The static files coming with Weblate can be overridden by placing into `/app/data/python/customize/static` (see *Docker container volumes*). For example creating `/app/data/python/customize/static/favicon.ico` will replace the favicon.

Tip: The files are copied to the corresponding location upon container startup, so a restart of Weblate is needed after changing the content of the volume.

Alternatively you can also include own module (see *Customizing Weblate*) and add it as separate volume to the Docker container, for example:

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

Adding own Python modules

3.8-5

You can place own Python modules in `/app/data/python/` (see *Docker container volumes*) and they can be then loaded by Weblate, most likely by using *Custom configuration files*.

Tip:

Customizing Weblate

Select your machine - local or cloud providers

With Docker Machine you can create your Weblate deployment either on your local machine, or on any large number of cloud-based deployments on e.g. Amazon AWS, Greenhost, and many other providers.

Installing on Debian and Ubuntu

Hardware requirements

Weblate should run on all 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.

???: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

???

System requirements

Install the dependencies needed to build the Python modules (see *Software requirements*):

```
apt install \
  libxml2-dev libxslt-dev libfreetype6-dev libjpeg-dev libz-dev libyaml-
↪dev \
  libcairo-dev gir1.2-pango-1.0 libgirepository1.0-dev libacl1-dev libssl-
↪dev \
  build-essential python3-gdbm python3-dev python3-pip python3-virtualenv ↪
↪virtualenv git
```

Install wanted optional dependencies depending on features you intend to use (see *Optional dependencies*):

```
apt install tesseract-ocr libtesseract-dev libleptonica-dev
```

Optionally install software for running production server, see *Running server*, *Database setup for Weblate*, *Background tasks using Celery*. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
apt install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with ``mod_wsgi``
apt install apache2 libapache2-mod-wsgi

# Caching backend: Redis
apt install redis-server

# Database server: PostgreSQL
apt install postgresql postgresql-contrib

# SMTP server
apt install exim4
```

Python modules

???: 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 tesseractocr
```

Configuring Weblate

⚠️: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **weblate** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`.
2. Adjust the values in the new `settings.py` file to your liking. You can stick with shipped example for testing purposes, but you will want changes for production setup, see *Adjusting configuration*.
3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check *Database setup for Weblate* for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Running server* and *Serving static files*):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see *Compressing client assets*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Background tasks using Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see *Running server* for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

You can now access Weblate on `http://localhost:8000/`.

Login with admin credentials obtained during installation or register with new users.

You can now run Weblate commands using **weblate** command when Weblate virtualenv is active, see *Management commands*.

You can stop the test server with `Ctrl+C`.

Adding translation

1. Open the admin interface (<http://localhost:8000/create/project/>) and create the project you want to translate. See *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 all 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.

??: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

??

System requirements

Install the dependencies needed to build the Python modules (see *Software requirements*):

```
zypper install \
  libxslt-devel libxml2-devel freetype-devel libjpeg-devel zlib-devel
↪ libyaml-devel \
  cairo-devel typelib-1_0-Pango-1_0 gobject-introspection-devel libacl-
↪ devel \
  python3-pip python3-virtualenv python3-devel git
```

Install wanted optional dependencies depending on features you intend to use (see *Optional dependencies*):

```
zypper install tesseract-ocr tesseract-devel leptonica-devel
```

Optionally install software for running production server, see *Running server*, *Database setup for Weblate*, *Background tasks using Celery*. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
zypper install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with ``mod_wsgi``
zypper install apache2 apache2-mod_wsgi

# Caching backend: Redis
zypper install redis-server
```

(?????????)

```
# Database server: PostgreSQL
zypper install postgresql postgresql-contrib

# SMTP server
zypper install postfix
```

Python modules

Tip: 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 aedon boto3 zeep chardet tesseract
```

Configuring Weblate

Tip: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **weblate** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`.
2. Adjust the values in the new `settings.py` file to your liking. You can stick with shipped example for testing purposes, but you will want changes for production setup, see [Adjusting configuration](#).
3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check [Database setup for Weblate](#) for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see [Running server](#) and [Serving static files](#)):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see [Compressing client assets](#)):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See [Background tasks using Celery](#) for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see *Running server* for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

You can now access Weblate on `http://localhost:8000/`.

Login with admin credentials obtained during installation or register with new users.

You can now run Weblate commands using **weblate** command when Weblate virtualenv is active, see *Management commands*.

You can stop the test server with Ctrl+C.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *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 all 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.

?: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.



System requirements

Install the dependencies needed to build the Python modules (see *Software requirements*):

```
dnf install \
  libxslt-devel libxml2-devel freetype-devel libjpeg-devel zlib-devel \
  ↳ libyaml-devel \
  cairo-devel pango-devel gobject-introspection-devel libacl-devel \
  python3-pip python3-virtualenv python3-devel git
```

Install wanted optional dependencies depending on features you intend to use (see *Optional dependencies*):

```
dnf install tesseract-langpack-eng tesseract-devel leptonica-devel
```

Optionally install software for running production server, see *Running server*, *Database setup for Weblate*, *Background tasks using Celery*. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
dnf install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with ``mod_wsgi``
dnf install apache2 apache2-mod_wsgi

# Caching backend: Redis
dnf install redis

# Database server: PostgreSQL
dnf install postgresql postgresql-contrib

# SMTP server
dnf install postfix
```

Python modules

!!!: 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 aedon boto3 zeep chardet tesseractocr
```

Configuring Weblate

❗: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **weblate** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`.
2. Adjust the values in the new `settings.py` file to your liking. You can stick with shipped example for testing purposes, but you will want changes for production setup, see [Adjusting configuration](#).
3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check [Database setup for Weblate](#) for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see [Running server](#) and [Serving static files](#)):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see [Compressing client assets](#)):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See [Background tasks using Celery](#) for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see [Running server](#) for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

You can now access Weblate on `http://localhost:8000/`.

Login with admin credentials obtained during installation or register with new users.

You can now run Weblate commands using **weblate** command when Weblate virtualenv is active, see [Management commands](#).

You can stop the test server with Ctrl+C.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See [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 all 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.

??: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

??

System requirements

Install the dependencies needed to build the Python modules (see *Software requirements*):

```
brew install pango libjpeg python git libyaml gobject-introspection
pip3 install virtualenv
```

Make sure pip will be able to find the libffi version provided by homebrew — this will be needed during the installation build step.

```
export PKG_CONFIG_PATH="/usr/local/opt/libffi/lib/pkgconfig"
```

Install wanted optional dependencies depending on features you intend to use (see *Optional dependencies*):

```
brew install tesseract
```

Optionally install software for running production server, see *Running server*, *Database setup for Weblate*, *Background tasks using Celery*. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
brew install nginx uwsgi

# Web server option 2: Apache with ``mod_wsgi``
brew install httpd

# Caching backend: Redis
brew install redis

# Database server: PostgreSQL
brew install postgresql
```

Python modules

!!!: 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 aidon boto3 zeep chardet tesseract
```

Configuring Weblate

!!!: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **weblate** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`.

2. Adjust the values in the new `settings.py` file to your liking. You can stick with shipped example for testing purposes, but you will want changes for production setup, see [Adjusting configuration](#).

3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check [Database setup for Weblate](#) for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see [Running server](#) and [Serving static files](#)):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see [Compressing client assets](#)):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See [Background tasks using Celery](#) for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see [Running server](#) for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

You can now access Weblate on `http://localhost:8000/`.

Login with admin credentials obtained during installation or register with new users.

You can now run Weblate commands using **weblate** command when Weblate virtualenv is active, see *Management commands*.

You can stop the test server with Ctrl+C.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *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 can stick with shipped example for testing purposes, but you will want changes for production setup, see *Adjusting configuration*.
6. Create the database used by Weblate, see *Database setup for Weblate*.
7. Build Django tables, static files and initial data (see *Filling up the database* and *Serving static files*):

```
weblate migrate
weblate collectstatic
weblate compress
weblate compilemessages
```

 This step should be repeated whenever you update the repository.

Installing on OpenShift

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

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

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.

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:

<https://www.djangoproject.com/>

<https://docs.celeryproject.org/>

<https://toolkit.translatehouse.org/>

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

<https://python-social-auth.readthedocs.io/>

<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`.

<https://www.mercurial-scm.org/>

<https://github.com/viraptor/phply>

<https://github.com/sirfz/tesseract>

<https://github.com/ubernostrum/akismet>

<https://pypi.org/project/ruamel.yaml/>

<https://docs.python-zeep.org/>

<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:

<https://git-scm.com/>

<https://cairographics.org/> <https://pango.gnome.org/> *Pango and Cairo*

<https://pypi.org/project/git-review/>

<https://git-scm.com/docs/git-svn>

<https://github.com/tesseract-ocr/tesseract>

<https://github.com/licensee/licensee>

Compile time dependencies

To compile 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`.

Weblate uses Pango and Cairo for rendering bitmap widgets (see *Promoting the translation*) and rendering checks (see `?`). 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.

Verifying release signatures

Weblate release are cryptographically signed by the releasing developer. Currently this is Michal Čihař. Fingerprint of his PGP key is:

```
63CB 1DF1 EF12 CF2A C0EE 5A32 9C27 B313 42B7 511D
```

and you can get more identification information from <https://keybase.io/nijel>.

You should verify that the signature matches the archive you have downloaded. This way you can be sure that you are using the same code that was released. You should also verify the date of the signature to make sure that you downloaded the latest version.

Each archive is accompanied with `.asc` files which contain the PGP signature for it. Once you have both of them in the same folder, you can verify the signature:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Ne 3. března 2019, 16:43:15 CET
gpg:          using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: Can't check signature: public key not found
```

As you can see GPG complains that it does not know the public key. At this point you should do one of the following steps:

Use *wkd* to download the key:

```
$ gpg --auto-key-locate wkd --locate-keys michal@cihar.com
pub  rsa4096 2009-06-17 [SC]
    63CB1DF1EF12CF2AC0EE5A329C27B31342B7511D
uid          [ultimate] Michal Čihař <michal@cihar.com>
uid          [ultimate] Michal Čihař <nijel@debian.org>
uid          [ultimate] [jpeg image of size 8848]
uid          [ultimate] Michal Čihař (Braiiins) <michal.cihar@braiiins.cz>
sub  rsa4096 2009-06-17 [E]
sub  rsa4096 2015-09-09 [S]
```

Download the keyring from [Michal's server](#), then import it with:

```
$ gpg --import wmxth3chu9jfxdxywj1skpmhsj311mzm
```

Download and import the key from one of the key servers:

```
$ gpg --keyserver hkp://pgp.mit.edu --recv-keys 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
↪87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: key 9C27B31342B7511D: "Michal Čihař <michal@cihar.com>" imported
gpg: Total number processed: 1
gpg:          unchanged: 1
```

This will improve the situation a bit - at this point you can verify that the signature from the given key is correct but you still can not trust the name used in the key:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Ne 3. března 2019, 16:43:15 CET
gpg:          using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: Good signature from "Michal Čihař <michal@cihar.com>" [ultimate]
gpg:          aka "Michal Čihař <nijel@debian.org>" [ultimate]
gpg:          aka "[jpeg image of size 8848]" [ultimate]
gpg:          aka "Michal Čihař (Braiiins) <michal.cihar@braiiins.cz>
↪" [ultimate]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
↪owner.
Primary key fingerprint: 63CB 1DF1 EF12 CF2A C0EE 5A32 9C27 B313 42B7 511D
```

The problem here is that anybody could issue the key with this name. You need to ensure that the key is actually owned by the mentioned person. The GNU Privacy Handbook covers this topic in the chapter [Validating other keys on your public keyring](#). The most reliable method is to meet the developer in person and exchange key fingerprints, however you can also rely on the web of trust. This way you can trust the key transitively 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
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Sun Mar  3 16:43:15 2019 CET
gpg:          using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: Good signature from "Michal Čihař <michal@cihar.com>" [ultimate]
gpg:          aka "Michal Čihař <nijel@debian.org>" [ultimate]
gpg:          aka "[jpeg image of size 8848]" [ultimate]
gpg:          aka "Michal Čihař (Braiiins) <michal.cihar@braiiins.cz>
↪" [ultimate]
```

Should the signature be invalid (the archive has been changed), you would get a clear error regardless of the fact that the key is trusted or not:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: Signature made Sun Mar  3 16:43:15 2019 CET
gpg:          using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: BAD signature from "Michal Čihař <michal@cihar.com>" [ultimate]
```

Filesystem permissions

The Weblate process needs to be able to read and write to the directory where it keeps data - `DATA_DIR`. All files within this directory should be owned and writable by the user running Weblate.

The default configuration places them in the same tree as the Weblate sources, however you might prefer to move these to a better location such as: `/var/lib/weblate`.

Weblate tries to create these directories automatically, but it will fail when it does not have permissions to do so.

You should also take care when running *Management commands*, as they should be ran under the same user as Weblate itself is running, otherwise permissions on some files might be wrong.

In the Docker container, all files in the `/app/data` volume have to be owned by weblate user inside the container (UID 1000).

??:

Serving static files

Database setup for Weblate

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

??:

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.

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

??:

PostgreSQL notes

Creating a database in PostgreSQL

It is usually a good idea to run Weblate in a separate database, and separate user account:

```
# If PostgreSQL was not installed before, set the main password
sudo -u postgres psql postgres -c "\password postgres"

# Create a database user called "weblate"
sudo -u postgres createuser --superuser --pwprompt weblate

# Create the database "weblate" owned by "weblate"
sudo -u postgres createdb -O weblate weblate
```

???: If you don't want to make the Weblate user a superuser in PostgreSQL, you can omit that. In that case you will have to perform some of the migration steps manually as a PostgreSQL superuser in schema Weblate will use:

```
CREATE EXTENSION IF NOT EXISTS pg_trgm WITH SCHEMA weblate;
```

Configuring Weblate to use PostgreSQL

The settings.py snippet for PostgreSQL:

```
DATABASES = {
    "default": {
        # Database engine
        "ENGINE": "django.db.backends.postgresql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Name of role to alter to set parameters in PostgreSQL,
        # use in case role name is different than user used for
        # authentication.
        "ALTER_ROLE": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "database.example.com",
        # Set to empty string for default
        "PORT": "",
    }
}
```

The migration code assumes that the role name matches username used while authenticating, in case it is not, please setting ALTER_ROLE. Otherwise you get PostgreSQL error about not existing role during the database migration (psycopg2.errors.UndefinedObject: role "weblate@hostname" does not exist).

MySQL and MariaDB

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

⚠️: Some Weblate features will perform better with *PostgreSQL*. This includes searching and translation memory, which both utilize full-text features in the database and PostgreSQL implementation is superior.

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

Following configuration is recommended for Weblate:

Use the `utf8mb4` charset to allow representation of higher Unicode planes (for example emojis).

Configure the server with `InnoDB_large_prefix` to allow longer indices on text fields.

Set the isolation level to `READ COMMITTED`.

The SQL mode should be set to `STRICT_TRANS_TABLES`.

Other configurations

Configuring outgoing e-mail

Weblate sends out e-mails on various occasions - for account activation and on various notifications configured by users. For this it needs access to an SMTP server.

The mail server setup is configured using these settings: `EMAIL_HOST`, `EMAIL_HOST_PASSWORD`, `EMAIL_HOST_USER` and `EMAIL_PORT`. Their names are quite self-explanatory, but you can find more info in the Django documentation.

⚠️: You can verify whether outgoing e-mail is working correctly by using the `sendtestemail` management command (see *Invoking management commands* for instructions on how to invoke it in different environments).

Running behind reverse proxy

Several features in Weblate rely on being able to get client IP address. This includes [IP address](#), [Spam protection](#) or [IP address](#).

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.

Tip:

`Spam protection` `IP_BEHIND_REVERSE_PROXY` `IP_PROXY_HEADER` `IP_PROXY_OFFSET` `SECURE_PROXY`

HTTP proxy

Weblate does execute VCS commands and those accept proxy configuration from environment. The recommended approach is to define proxy settings in `settings.py`:

```
import os
os.environ['http_proxy'] = "http://proxy.example.com:8080"
os.environ['HTTPS_PROXY'] = "http://proxy.example.com:8080"
```

Tip:

[Proxy Environment Variables](#)

Adjusting configuration

Tip:

[Copy settings](#)

Copy `weblate/settings_example.py` to `weblate/settings.py` and adjust it to match your setup. You will probably want to adjust the following options: `ADMINS`

List of site administrators to receive notifications when something goes wrong, for example notifications on failed merges, or Django errors.

Tip:

`ADMINS`

`ALLOWED_HOSTS`

You need to set this to list the hosts your site is supposed to serve. For example:

```
ALLOWED_HOSTS = ['demo.weblate.org']
```

Alternatively you can include wildcard:

```
ALLOWED_HOSTS = ['*']
```

Tip:

`ALLOWED_HOSTS` `WEBLATE_ALLOWED_HOSTS` [Allowed hosts setup](#)

`SESSION_ENGINE`

Configure how your sessions will be stored. In case you keep the default database backend engine, you should schedule: `weblate clearsessions` to remove stale session data from the database.

If you are using Redis as cache (see [Enable caching](#)) it is recommended to use it for sessions as well:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
```

Tip:

[Configuring the session engine](#) `SESSION_ENGINE`

`DATABASES`

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

??:

[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.

??:

DEBUG

DEFAULT_FROM_EMAIL

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

??:

DEFAULT_FROM_EMAIL

SECRET_KEY

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

??:

SECRET_KEY

SERVER_EMAIL

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

??:

SERVER_EMAIL

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.

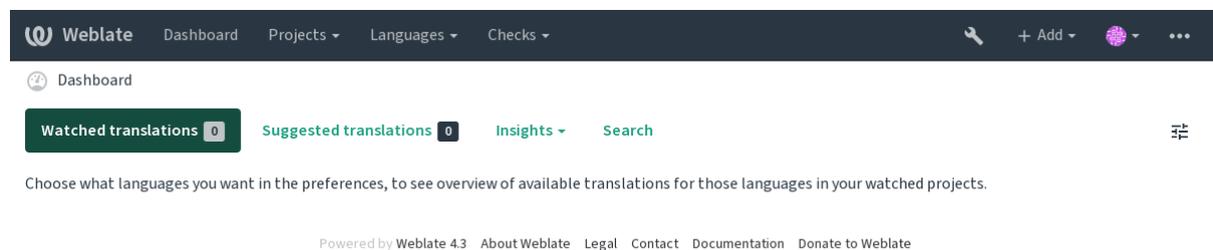
Once you are done, you should also check the *Performance report* in the admin interface, which will give you hints of potential non optimal configuration on your site.

??:

??????????

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
```

??:

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.

??:

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'),  
)
```

??:

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 **???**: Prior to the 4.2 release the Django sites framework was used instead, please see [The “sites” framework](#).

??:

Allowed hosts setup **?** *Correctly configure HTTPS* **?** *SITE_DOMAIN* **?** *WEBLATE_SITE_DOMAIN* **?** *ENABLE_HTTPS*

Correctly configure HTTPS

It is strongly recommended to run Weblate using the encrypted HTTPS protocol. After enabling it, you should set *ENABLE_HTTPS* in the settings:

```
ENABLE_HTTPS = True
```

???: You might want to set up HSTS as well, see [SSL/HTTPS](#) for more details.

??:

ENABLE_HTTPS **?** *Allowed hosts setup* **?** *Set correct site domain*

Set properly SECURE_HSTS_SECONDS

If your site is served over SSL, you have to consider setting a value for *SECURE_HSTS_SECONDS* in the *settings.py* to enable HTTP Strict Transport Security. By default it's set to 0 as shown below.

```
SECURE_HSTS_SECONDS = 0
```

If set to a non-zero integer value, the `django.middleware.security.SecurityMiddleware` sets the HTTP Strict Transport Security header on all responses that do not already have it.

??: 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.

??:

[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',
        }
    }
}
```

??:

[Django's cache framework](#)

[Django's cache framework](#)

In addition to caching of Django, Weblate performs caching of avatars. It is recommended to use a separate, file-backed cache for this purpose:

```
CACHES = {
    'default': {
        # Default caching backend setup, see above
        'BACKEND': 'django_redis.cache.RedisCache',
        'LOCATION': 'unix:///var/run/redis/redis.sock?db=0',
        'OPTIONS': {
            'CLIENT_CLASS': 'django_redis.client.DefaultClient',
            'PARSER_CLASS': 'redis.connection.HiredisParser',
        }
    },
    'avatar': {
        'BACKEND': 'django.core.cache.backends.filebased.FileBasedCache',
        'LOCATION': os.path.join(DATA_DIR, 'avatar-cache'),
        'TIMEOUT': 604800,
        'OPTIONS': {
            'MAX_ENTRIES': 1000,
        }
    },
}
```

??:

[ENABLE_AVATARS](#) [AVATAR_URL_PREFIX](#) [Avatars](#) [Enable caching](#) [Django's cache framework](#)

Configure e-mail sending

Weblate needs to send out e-mails on several occasions, and these e-mails should have a correct sender address, please configure `SERVER_EMAIL` and `DEFAULT_FROM_EMAIL` to match your environment, for example:

```
SERVER_EMAIL = 'admin@example.org'
DEFAULT_FROM_EMAIL = 'weblate@example.org'
```

Tip: 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.

Tip:

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`.

Tip: On Docker container, this is available as `WEBLATE_ALLOWED_HOSTS`.

Tip:

`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.

Tip:

`SECRET_KEY`

Home directory

Tip: 2.1 **Tip:** This is no longer required, Weblate now stores all its data in `DATA_DIR`.

The home directory for the user running Weblate should exist and be writable by this user. This is especially needed if you want to use SSH to access private repositories, but Git might need to access this directory as well (depending on the Git version you use).

You can change the directory used by Weblate in `settings.py`, for example to set it to configuration directory under the Weblate tree:

```
os.environ['HOME'] = os.path.join(BASE_DIR, 'configuration')
```

Tip: 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.

Tip:

[XXXXXXXXXX](#)

Template loading

It is recommended to use a cached template loader for Django. It caches parsed templates and avoids the need to do parsing with every single request. You can configure it using the following snippet (the `loaders` setting is important here):

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            os.path.join(BASE_DIR, 'templates'),
        ],
        'OPTIONS': {
            'context_processors': [
                'django.contrib.auth.context_processors.auth',
                'django.template.context_processors.debug',
                'django.template.context_processors.i18n',
                'django.template.context_processors.request',
                'django.template.context_processors.csrf',
                'django.contrib.messages.context_processors.messages',
                'weblate.trans.context_processors.weblate_context',
            ],
            'loaders': [
                ('django.template.loaders.cached.Loader', [
                    'django.template.loaders.filesystem.Loader',
                    'django.template.loaders.app_directories.Loader',
                ]),
            ],
        },
    ],
]
```

:

`django.template.loaders.cached.Loader`

Running maintenance tasks

For optimal performance, it is good idea to run some maintenance tasks in the background. This is now automatically done by *Background tasks using Celery* and covers following tasks:

Configuration health check (hourly).

Committing pending changes (hourly), see *Lazy commits* and *commit_pending*.

Updating component alerts (daily).

Update remote branches (nightly), see *AUTO_UPDATE*.

Translation memory backup to JSON (daily), see *dump_memory*.

Fulltext and database maintenance tasks (daily and weekly tasks), see *cleanuptrans*.

 3.2 : Since version 3.2, the default way of executing these tasks is using Celery and Weblate already comes with proper configuration, see *Background tasks using Celery*.

System locales and encoding

The system locales should be configured to UTF-8 capable ones. On most Linux distributions this is the default setting. In case it is not the case on your system, please change locales to UTF-8 variant.

For example by editing `/etc/default/locale` and setting there `LANG="C.UTF-8"`.

In some cases the individual services have separate configuration for locales. For example when using Apache you might want to set it in `/etc/apache2/envvars`:

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

Using custom certificate authority

Weblate does verify SSL certificates during HTTP requests. In case you are using custom certificate authority which is not trusted in default bundles, you will have to add its certificate as trusted.

The preferred approach is to do this at system level, please check your distro documentation for more details (for example on debian this can be done by placing the CA certificate into `/usr/local/share/ca-certificates/` and running `update-ca-certificates`).

Once this is done, system tools will trust the certificate and this includes Git.

For Python code, you will need to configure requests to use system CA bundle instead of the one shipped with it. This can be achieved by placing following snippet to `settings.py` (the path is Debian specific):

```
import os
os.environ["REQUESTS_CA_BUNDLE"] = "/etc/ssl/certs/ca-certificates.crt"
```

Compressing client assets

Weblate comes with a bunch of JavaScript and CSS files. For performance reasons it is good to compress them before sending to a client. In default configuration this is done on the fly at cost of little overhead. On big installations, it is recommended to enable offline compression mode. This needs to be done in the configuration and the compression has to be triggered on every Weblate upgrade.

The configuration switch is simple by enabling `django.conf.settings.COMPRESS_OFFLINE` and configuring `django.conf.settings.COMPRESS_OFFLINE_CONTEXT` (the latter is already included in the example configuration):

```
COMPRESS_OFFLINE = True
```

On each deploy you need to compress the files to match current version:

```
weblate compress
```

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

🔗:

Common Deployment Scenarios [🔗](#) *Serving static files*

Running server

You will need several services to run Weblate, the recommended setup consists of:

Database server (see [Database setup for Weblate](#))

Cache server (see [Enable caching](#))

Frontend web server for static files and SSL termination (see [Serving static files](#))

Wsgi server for dynamic content (see [Sample configuration for NGINX and uWSGI](#))

Celery for executing background tasks (see [Background tasks using Celery](#))

🔗: There are some dependencies between the services, for example cache and database should be running when starting up Celery or uwsgi processes.

In most cases, you will run all services on single (virtual) server, but in case your installation is heavy loaded, you can split up the services. The only limitation on this is that Celery and Wsgi servers need access to `DATA_DIR`.

Running web server

Running Weblate is not different from running any other Django based program. Django is usually executed as uWSGI or fcgi (see examples for different webservers below).

For testing purposes, you can use the built-in web server in Django:

```
weblate runserver
```

⚠️: 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`.

⚠️⚠️: 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

⚠️⚠️⚠️ 2.4 **⚠️⚠️**: Prior to version 2.4, Weblate didn't properly use the Django static files framework and the setup was more complex.

Django needs to collect its static files in a single directory. To do so, execute `weblate collectstatic --noinput`. This will copy the static files into a directory specified by the `STATIC_ROOT` setting (this defaults to a static directory inside `DATA_DIR`).

It is recommended to serve static files directly from your web server, you should use that for the following paths:

Serves static files for Weblate and the admin interface (from defined by `STATIC_ROOT`).

Used for user media uploads (e.g. screenshots).

Should be rewritten to rewrite a rule to serve `/static/favicon.ico`.

⚠️:

[Compressing client assets](#)[Deploying Django](#)[Deploying static files](#)

Content security policy

The default Weblate configuration enables `weblate.middleware.SecurityMiddleware` middleware which sets security related HTTP headers like `Content-Security-Policy` or `X-XSS-Protection`. These are by default set up to work with Weblate and its configuration, but this might need customization for your environment.

⚠️:

`CSP_SCRIPT_SRC`[CSP_IMG_SRC](#)[CSP_CONNECT_SRC](#)[CSP_STYLE_SRC](#)[CSP_FONT_SRC](#)

Sample configuration for NGINX and uWSGI

To run production webserver, use the wsgi wrapper installed with Weblate (in virtual env case it is installed as `~/weblate-env/lib/python3.7/site-packages/weblate/wsgi.py`). Don't forget to set the Python search path to your virtualenv as well (for example using `virtualenv = /home/user/weblate-env` in uWSGI).

The following configuration runs Weblate as uWSGI under the NGINX webserver.

Configuration for NGINX (also available as `weblate/examples/weblate.nginx.conf`):

```
# This example assumes Weblate is installed in virtualenv in /home/weblate/  
↪weblate-env  
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match_  
↪your setup.  
server {  
    listen 80;  
    server_name weblate;
```

([⚠️⚠️⚠️⚠️⚠️](#))

```

# Not used
root /var/www/html;

location ~ ^/favicon.ico$ {
    # DATA_DIR/static/favicon.ico
    alias /home/weblate/data/static/favicon.ico;
    expires 30d;
}

location /static/ {
    # DATA_DIR/static/
    alias /home/weblate/data/static/;
    expires 30d;
}

location /media/ {
    # DATA_DIR/media/
    alias /home/weblate/data/media/;
    expires 30d;
}

location / {
    include uwsgi_params;
    # Needed for long running operations in admin interface
    uwsgi_read_timeout 3600;
    # Adjust based to uwsgi configuration:
    uwsgi_pass unix:///run/uwsgi/app/weblate/socket;
    # uwsgi_pass 127.0.0.1:8080;
}
}

```

Configuration for uWSGI (also available as weblate/examples/weblate.uwsgi.ini):

```

# This example assumes Weblate is installed in virtualenv in /home/weblate/
↪weblate-env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match
↪your setup.
[uwsgi]
plugins      = python3
master       = true
protocol     = uwsgi
socket       = 127.0.0.1:8080
wsgi-file    = /home/weblate/weblate-env/lib/python3.7/site-packages/
↪weblate/wsgi.py

# Add path to Weblate checkout if you did not install
# Weblate by pip
# python-path = /path/to/weblate

# In case you're using virtualenv uncomment this:
virtualenv = /home/weblate/weblate-env

# Needed for OAuth/OpenID
buffer-size  = 8192

# Reload when consuming too much of memory
reload-on-rss = 250

# Increase number of workers for heavily loaded sites
workers      = 8

# Enable threads for Sentry error submission
enable-threads = true

# Child processes do not need file descriptors
close-on-exec = true

```

```
# 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
```

??:

How to use Django with uWSGI

Sample configuration for Apache

The following configuration runs Weblate as WSGI, you need to have enabled mod_wsgi (available as weblate/examples/apache.conf):

```
#
# VirtualHost for Weblate
#
# This example assumes Weblate is installed in virtualenv in /home/weblate/
↪weblate-env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match
↪your setup.
#
<VirtualHost *:80>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>

    # DATA_DIR/media/
    Alias /media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    # Path to your Weblate virtualenv
    WSGIDaemonProcess weblate python-home=/home/weblate/weblate-env
    WSGIProcessGroup weblate
    WSGIApplicationGroup %{GLOBAL}

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

    <Directory /home/weblate/weblate-env/lib/python3.7/site-packages/
↪weblate/>
```

```

    <Files wsgi.py>
    Require all granted
    </Files>
</Directory>

</VirtualHost>

```

❗: 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`.

❗:

System locales and encoding [How to use Django with Apache and mod_wsgi](#)

Sample configuration for Apache and Gunicorn

The following configuration runs Weblate in Gunicorn and Apache 2.4 (available as `weblate/examples/apache.gunicorn.conf`):

```

#
# VirtualHost for Weblate using gunicorn on localhost:8000
#
# This example assumes Weblate is installed in virtualenv in /home/weblate/
↪weblate-env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match_
↪your setup.
#
<VirtualHost *:443>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>

    # DATA_DIR/media/
    Alias /media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/https_cert.cert
    SSLCertificateKeyFile /etc/apache2/ssl/https_key.pem
    SSLProxyEngine On

    ProxyPass /favicon.ico !
    ProxyPass /static/ !
    ProxyPass /media/ !

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

```

❗:

How to use Django with Gunicorn

Running Weblate under path

1.3: This is supported since Weblate 1.3.

A sample Apache configuration to serve Weblate under /weblate. Again using mod_wsgi (also available as weblate/examples/apache-path.conf):

```
#
# VirtualHost for Weblate, running under /weblate path
#
# This example assumes Weblate is installed in virtualenv in /home/weblate/
↪weblate-env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match
↪your setup.
#
<VirtualHost *:80>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /weblate/favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /weblate/static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>

    # DATA_DIR/media/
    Alias /weblate/media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    # Path to your Weblate virtualenv
    WSGIDaemonProcess weblate python-home=/home/weblate/weblate-env
    WSGIProcessGroup weblate
    WSGIApplicationGroup %{GLOBAL}

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

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

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

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

Background tasks using Celery

3.2

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

A typical setup using Redis as a backend looks like this:

```
CELERY_TASK_ALWAYS_EAGER = False
CELERY_BROKER_URL = 'redis://localhost:6379'
CELERY_RESULT_BACKEND = CELERY_BROKER_URL
```

You should also start the Celery worker to process the tasks and start scheduled tasks, this can be done directly on the command line (which is mostly useful when debugging or developing):

```
./weblate/examples/celery start
./weblate/examples/celery stop
```

Running Celery as system service

Most likely you will want to run Celery as a daemon and that is covered by [Daemonization](#). For the most common Linux setup using systemd, you can use the example files shipped in the `examples` folder listed below.

Systemd unit to be placed as `/etc/systemd/system/celery-weblate.service`:

```
[Unit]
Description=Celery Service (Weblate)
After=network.target

[Service]
Type=forking
User=weblate
Group=weblate
EnvironmentFile=/etc/default/celery-weblate
WorkingDirectory=/home/weblate
RuntimeDirectory=celery
RuntimeDirectoryPreserve=restart
LogsDirectory=celery
ExecStart=/bin/sh -c '${CELERY_BIN} multi start ${CELERYD_NODES} \
  -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \
  --logfile=${CELERYD_LOG_FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_
  ↳OPTS}'
ExecStop=/bin/sh -c '${CELERY_BIN} multi stopwait ${CELERYD_NODES} \
  --pidfile=${CELERYD_PID_FILE}'
ExecReload=/bin/sh -c '${CELERY_BIN} multi restart ${CELERYD_NODES} \
  -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \
  --logfile=${CELERYD_LOG_FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_
  ↳OPTS}'

[Install]
WantedBy=multi-user.target
```

Environment configuration to be placed as `/etc/default/celery-weblate`:

```
# Name of nodes to start
CELERYD_NODES="celery notify 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"

# Extra command-line arguments to the worker,
# increase concurrency if you get weblate.E019
CELERYD_OPTS="--beat:celery --queues:celery=celery --prefetch-
  ↳multiplier:celery=4 \
```

(XXXXXXXXXX)

```
--queues:notify=notify --prefetch-multiplier:notify=10 \  
--queues:memory=memory --prefetch-multiplier:memory=10 \  
--queues:translate=translate --prefetch-multiplier:translate=4 \  
--concurrency:backup=1 --queues:backup=backup --prefetch-  
↪multiplier:backup=2"  
  
# Logging configuration  
# - %n will be replaced with the first part of the nodename.  
# - %I will be replaced with the current child process index  
# and is important when using the prefork pool to avoid race conditions.  
CELERYD_PID_FILE="/var/run/celery/weblate-%n.pid"  
CELERYD_LOG_FILE="/var/log/celery/weblate-%n%I.log"  
CELERYD_LOG_LEVEL="INFO"  
  
# Internal Weblate variable to indicate we're running inside Celery  
CELERY_WORKER_RUNNING="1"
```

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

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

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

Periodic tasks using Celery beat

Weblate comes with built-in setup for scheduled tasks. You can however define additional tasks in `settings.py`, for example see *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.

⚠: 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*.

⚠:

Configuration and defaults Workers Guide Daemonization Monitoring and Management
Guide `celery_queues`

Monitoring Weblate

Weblate provides the `/healthz/` URL to be used in simple health checks, for example using Kubernetes.

Collecting error reports

Weblate, as any other software, can fail. In order to collect useful failure states we recommend to use third party services to collect such information. This is especially useful in case of failing Celery tasks, which would otherwise only report error to the logs and you won't get notified on them. Weblate has support for the following services:

Sentry

Weblate has built-in support for [Sentry](#). To use it, it's enough to set `SENTRY_DSN` in the `settings.py`:

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

Rollbar

Weblate has built-in support for [Rollbar](#). To use it, it's enough to follow instructions for [Rollbar notifier for Python](#).

In short, you need to adjust `settings.py`:

```
# Add rollbar as last middleware:
MIDDLEWARE = [
    # ... other middleware classes ...
    'rollbar.contrib.django.middleware.RollbarNotifierMiddleware',
]

# Configure client access
ROLLBAR = {
    'access_token': 'POST_SERVER_ITEM_ACCESS_TOKEN',
    'client_token': 'POST_CLIENT_ITEM_ACCESS_TOKEN',
    'environment': 'development' if DEBUG else 'production',
    'branch': 'master',
    'root': '/absolute/path/to/code/root',
}
```

Everything else is integrated automatically, you will now collect both server and client side errors.

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.

Weblate deployments

Weblate can be easily installed in your cloud. Please find detailed guide for your platform:

Installing using Docker

Installing on OpenShift

Helm Chart

You can install Weblate on Kubernetes using Helm. See <https://github.com/WeblateOrg/helm/tree/master/charts/weblate> for the detailed instructions.

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 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
```

Upgrading Weblate

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.

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.

⚠️: 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:

```
cd weblate-src
git pull
# Update Weblate inside your virtualenv
. ~/weblate-env/bin/pip install -e .
# Install dependencies directly when not using virtualenv
pip install --upgrade -r requirements.txt
```

3. Upgrade configuration file, refer to `settings_example.py` or *Version specific instructions* for needed steps.

4. Upgrade database structure:

```
weblate migrate --noinput
```

5. Collect updated static files (see *Running server* and *Serving static files*):

```
weblate collectstatic --noinput
```

6. Compress JavaScript and CSS files (optional, see *Compressing client assets*):

```
weblate compress
```

7. If you are running version from Git, you should also regenerate locale files every time you are upgrading. You can do this by invoking:

```
weblate compilemessages
```

8. Verify that your setup is sane (see also *Production setup*):

```
weblate check --deploy
```

9. Restart celery worker (see *Background tasks using Celery*).

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.

⚠️:

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.

🔗:

[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`.

`DEFAULT_THROTTLE_CLASSES` `API`

There are some new and updated requirements.

There is a change in `INSTALLED_APPS`.

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

🔗:

[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.

🔗:

[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* `weblate` 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`.

Changed in 4.3.1: The Celery configuration was changed to add `memory` queue. Please adjust your startup scripts and `CELERY_TASK_ROUTES` setting.

??:

Generic upgrade instructions

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.

Migrating from other databases to PostgreSQL

If you are running Weblate on other database than PostgreSQL, you should migrate to PostgreSQL as that will be the only supported database backend in the 4.0 release. The following steps will guide you in migrating your data between the databases. Please remember to stop both web and Celery servers prior to the migration, otherwise you might end up with inconsistent data.

Creating a database in PostgreSQL

It is usually a good idea to run Weblate in a separate database, and separate user account:

```
# If PostgreSQL was not installed before, set the main password
sudo -u postgres psql postgres -c "\password postgres"

# Create a database user called "weblate"
sudo -u postgres createuser -D -P weblate

# Create the database "weblate" owned by "weblate"
sudo -u postgres createdb -O weblate weblate
```

Migrating using Django JSON dumps

The simplest approach for migration is to utilize Django JSON dumps. This works well for smaller installations. On bigger sites you might want to use *pgloader* instead, see *Migrating to 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',
```

(2)(2)(2)(2)(2)(2)

```

# Database user
'USER': 'weblate',
# Database password
'PASSWORD': 'password',
# Set to empty string for localhost
'HOST': 'database.example.com',
# Set to empty string for default
'PORT': '',
# Additional database options
'OPTIONS': {
    # In case of using an older MySQL server, which has MyISAM as
    ↪a default storage
    # 'init_command': 'SET storage_engine=INNODB',
    # Uncomment for MySQL older than 5.7:
    # 'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
    # If your server supports it, see the Unicode issues above
    'charset': 'utf8mb4',
    ↪error:
    # Change connection timeout in case you get MySQL gone away.
    'connect_timeout': 28800,
}
},
'postgresql': {
    # Database engine
    'ENGINE': 'django.db.backends.postgresql',
    # Database name
    'NAME': 'weblate',
    # Database user
    'USER': 'weblate',
    # Database password
    'PASSWORD': 'password',
    # Set to empty string for localhost
    'HOST': 'database.example.com',
    # Set to empty string for default
    'PORT': '',
}
}

```

2.Run migrations and drop any data inserted into the tables:

```

weblate migrate --database=postgresql
weblate sqlflush --database=postgresql | weblate dbshell --
↪database=postgresql

```

3.Dump legacy database and import to PostgreSQL

```

weblate dumpdata --all --output weblate.json
weblate loaddata weblate.json --database=postgresql

```

4.Adjust `DATABASES` to use just PostgreSQL database as default, remove legacy connection.

Weblate should be now ready to run from the PostgreSQL database.

Migrating to PostgreSQL using pgloader

The `pgloader` is a generic migration tool to migrate data to PostgreSQL. You can use it to migrate Weblate database.

1.Adjust your `settings.py` to use PostgreSQL as a database.

2.Migrate the schema in the PostgreSQL database:

```

weblate migrate
weblate sqlflush | weblate dbshell

```

3.Run the `pgloader` to transfer the data. The following script can be used to migrate the database, but you might want to learn more about `pgloader` to understand what it does and tweak it to match your setup:

```
LOAD DATABASE
FROM      mysql://weblate:password@localhost/weblate
INTO      postgresql://weblate:password@localhost/weblate

WITH include no drop, truncate, create no tables, create no indexes, no_
↔foreign keys, disable triggers, reset sequences, data only

ALTER SCHEMA 'weblate' RENAME TO 'public'
;
```

Migrating from Pootle

As Weblate was originally written as replacement from Pootle, it is supported to migrate user accounts from Pootle. You can dump the users from Pootle and import them using *importusers*.

Backing up and moving Weblate

Automated backup using BorgBackup

3.9

Weblate 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 on the *Backups* tab.

Note: Only PostgreSQL database is included in the automated backups. Other database engines have to be backed up manually. You are recommended to migrate to PostgreSQL, see *Database setup for Weblate* and *Migrating from other databases to PostgreSQL*.

The backups using Borg are incremental and Weblate is configured to keep following backups:

14 daily backups

8 weekly backups

6 monthly backups

Backup process triggered

Weblate status Backups Translation memory Performance report SSH keys Alerts Repositories Users Tools

Backup service: /tmp/tmpvyevrwjlweblate

Backup service credentials	Oct. 15, 2020
Backup repository	/tmp/tmpvyevrwjlweblate
Passphrase	Xm(0zW2&LZ1@I6eCG0vZgIYj jgvI) sV&ubz7r0Wrx77Tcvo4@a 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	Oct. 15, 2020
Backup performed	Oct. 15, 2020
Repository initialization	Oct. 15, 2020

Turn off Perform backup Delete

Activate support package

The support packages include priority e-mail support, or cloud backups of your Weblate installation.

Activation token

Please enter the activation token obtained when making the subscription.

Activate Purchase support package

Add backup service

Backup repository URL

Use /path/to/repo for local backups or user@host:/path/to/repo for remote SSH backups.

Add

Borg encryption key

BorgBackup creates encrypted backups and without a passphrase you will not be able to restore the backup. The passphrase is generated when adding new backup service and you should copy it and keep it in a secure place.

In case you are using Weblate provisioned backup storage, please backup your private SSH key as well — it is used to access your backups.

🔑:

borg init

Weblate provisioned backup storage

The easiest approach to backup your Weblate instance is to purchase [backup service at weblate.org](https://weblate.org/support/#backup). The process of activating can be performed in few steps:

1. Purchase backup service on <https://weblate.org/support/#backup>.
2. Enter obtained key in the management interface, see [Integrating support](#).
3. Weblate will connect to the cloud service and obtain access information for the backups.
4. Turn on the new backup configuration on the *Backups* tab.
5. Backup Borg credentials in order to be able to restore the backups, see [Borg encryption key](#).

!!!: The manual step of turning on is there for your safety. Without your consent no data is sent to the backup repository obtained through the registration process.

Using custom backup storage

You can also use your own storage for the backups. SSH can be used to store backups on the remote destination, the target server needs to have [BorgBackup](#) installed.

!:

[General](#) in the Borg documentation

Local filesystem

It is recommended to specify absolute path for the local backup, for example */path/to/backup*. The directory has to be writable by user running Weblate (see [Filesystem permissions](#)). In case it doesn't exist, Weblate will attempt to create it, but it needs permissions to do so.

!!!: When running Weblate in Docker, please make sure that the backup location is exposed as a volume from the Weblate container. Otherwise the backups would be discarded by Docker on container restart.

One option is to place backups in existing volume. For example choose */app/data/borgbackup*. This is existing volume in the container.

You can also add new container for the backups in the Docker compose file and use for example */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 will not be able to write the backups there.

!!!! ?????

Remote backups using SSH are supported. The SSH server needs to have [BorgBackup](#) installed. Weblate connects to the server using SSH key, please make sure the Weblate SSH key is accepted by the server (see [Weblate SSH !!](#)).

!!!: *Weblate provisioned backup storage* provides you automated remote backups.

Restoring from BorgBackup

1. Restore access to your backup repository and prepare your backup passphrase.
2. List backup existing on the server using `borg list REPOSITORY`.
3. Restore the desired backup to 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 Weblate configuration (`backups/settings.py`, see *Dumped data for backups*) to the correct location, see *Adjusting configuration*.
6. Copy the whole restored data dir to location configured by `DATA_DIR`.

The Borg session might look like:

```
$ 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:
```

❏:

borg list❏borg extract

Manual backup

Depending on what you want to save, back up the type data Weblate stores in each respective place.

❏❏❏: In case you are doing manual backups, you might want to silent Weblate warning about lack of backups by adding `weblate.I028` to `SILENCED_SYSTEM_CHECKS` in `settings.py` or `WEBLATE_SILENCED_SYSTEM_CHECKS` for Docker.

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

Database

The actual storage location depends on your database setup.

The database is the most important storage. Set up regular backups of your database, without it all your translation setup will be gone.

Native database backup

The recommended approach is to do dump of the database using database native tools such as `pg_dump` or `mysql-dump`. It usually performs better than Django backup and restores complete tables with all data.

You can restore this backup in newer Weblate release, it will perform any necessary migrations when running in `migrate`. Please consult *Upgrading Weblate* on more detailed information how to perform upgrade between versions.

Django database backup

Alternatively you can backup database using Django's `dumpdata` command. That way the backup is database agnostic and can be used in case you want to change database backend.

Prior to restoring you need to be running exactly same Weblate version as was used when doing backups. 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`.

Once this is done, some entries will be already 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 management shell (see *Invoking management commands*):

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



If you have enough backup space, simply backup the whole `DATA_DIR`. This is safe bet even if it includes some files you don't want. The following sections describe in detail what you should back up and what you can skip.

Dumped data for backups

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`.

The database backups are by default saved as plain text, but they can also be compressed or entirely skipped by using `DATABASE_BACKUP`.

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 push on commit enabled for all your translation components, all Weblate changes are included upstream and you do not have to backup the repositories on the Weblate side. They can be cloned again from the upstream locations 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 user uploaded files (e.g. *Visual context for strings*).

Celery tasks

The Celery tasks queue might contain some info, but is usually not needed for a backup. At most you will lose updates that have not yet been processed to translation memory. It is recommended to perform the fulltext or repository updates upon restoring anyhow, so there is no problem in losing these.

??:

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 instance:

```
$ 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 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. For instance, to avoid saving the translation memory (in backups folder), you could 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
```

Restoring manual backup

1. Restore all data you have backed up.
2. Update all repositories using `updategit`.

```
weblate updategit --all
```

Moving a Weblate installation

Relocate your installation to a different system by following the backup and restore instructions above.

??:

Upgrading from Python 2 to Python 3 [Migrating from other databases to PostgreSQL](#)

??

??????

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 [??????](#).

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.

??:

Authentication settings describes how to configure authentication in the official Docker image.

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](#).

??: 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:

```
SOCIAL_AUTH_OPENSUSE_FORCE_EMAIL_VALIDATION = True
```

??:

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.

??:

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:

```
# 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',
)
```

??:

OpenID

GitHub authentication

You need to register an 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/`.

??: 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](#).

??:

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
```

??: 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*.

??:

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/`

```
# 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'
```

??: 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*.

??:

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/`

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.facebook.FacebookOAuth2',
    'social_core.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
```

(???)

(XXXXXXXXXX)

```
# Social auth backends setup
SOCIAL_AUTH_FACEBOOK_KEY = 'key'
SOCIAL_AUTH_FACEBOOK_SECRET = 'secret'
SOCIAL_AUTH_FACEBOOK_SCOPE = ['email', 'public_profile']
```

??: 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*.

??:

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.

```
# 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/'
```

??: 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*.

??:

GitLab

Microsoft Azure Active Directory

Weblate can be configured to use common or specific tenants for authentication.

The redirect URL is `https://WEBLATE_SERVER/accounts/complete/azuread-oauth2/` for common and `https://WEBLATE_SERVER/accounts/complete/azuread-tenant-oauth2/` for tenant-specific authentication.

```
# Azure AD common

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.azuread.AzureADOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# OAuth2 keys
SOCIAL_AUTH_AZUREAD_OAUTH2_KEY = ""
SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET = ""
```

```
# Azure AD Tenant

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.azuread_tenant.AzureADTenantOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# OAuth2 keys
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY = ""
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECRET = ""
# Tenant ID
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENANT_ID = ""
```

??: 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*.

??:

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/`.

```
# 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 = ''
```

??: 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*.

??:

Slack

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.

????: 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:

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

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.

SAML authentication

4.1.1

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:

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

# Social auth backends setup
SOCIAL_AUTH_SAML_SP_PUBLIC_CERT = "-----BEGIN CERTIFICATE-----"
SOCIAL_AUTH_SAML_SP_PRIVATE_KEY = "-----BEGIN PRIVATE KEY-----"
SOCIAL_AUTH_SAML_ENABLED_IDPS = {
    "weblate": {
        "entity_id": "https://idp.testshib.org/idp/shibboleth",
        "url": "https://idp.testshib.org/idp/profile/SAML2/Redirect/SSO",
        "x509cert": "MIIEDjCCAvagAwIBAgIBADA ... 8Bbnl+ev0peYzxFyF5sQA==",
        "attr_name": "full_name",
        "attr_username": "username",
        "attr_email": "email",
    }
}
```

??:

Configuring SAML in Docker SAML

LDAP authentication

LDAP authentication can be best achieved using the `django-auth-ldap` package. You can install it via usual means:

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

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

??: With django-auth-ldap older than 1.3.0 the `django_auth_ldap.config.LDAPSearch` will not work properly for newly created users.

??: 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:

```
# Add LDAP backed, keep Django one if you want to be able to login
# even without LDAP for admin account
AUTHENTICATION_BACKENDS = (
    'django_auth_ldap.backend.LDAPBackend',
    'weblate.accounts.auth.WeblateUserBackend',
)

# LDAP server address
AUTH_LDAP_SERVER_URI = 'ldaps://ldap.example.net'

# DN to use for authentication
AUTH_LDAP_USER_DN_TEMPLATE = 'cn=%(user)s,o=Example'
# Depending on your LDAP server, you might use a different DN
# like:
# AUTH_LDAP_USER_DN_TEMPLATE = 'ou=users,dc=example,dc=com'

# List of attributes to import from LDAP upon login
# Weblate stores full name of the user in the full_name attribute
AUTH_LDAP_USER_ATTR_MAP = {
    'full_name': 'name',
    # Use the following if your LDAP server does not have full name
    # Weblate will merge them later
    # 'first_name': 'givenName',
    # 'last_name': 'sn',
    # Email is required for Weblate (used in VCS commits)
    'email': 'mail',
}

# Hide the registration form
REGISTRATION_OPEN = False
```

??: 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 can not use direct bind for authentication, you will need to use search, and provide a user to bind for the search. For example:

```
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

```
import ldap
from django_auth_ldap.config import LDAPSearch, \
↳NestedActiveDirectoryGroupType

AUTH_LDAP_BIND_DN = "CN=ldap,CN=Users,DC=example,DC=com"
AUTH_LDAP_BIND_PASSWORD = "password"

# User and group search objects and types
AUTH_LDAP_USER_SEARCH = LDAPSearch("CN=Users,DC=example,DC=com", ldap.
↳SCOPE_SUBTREE, "(sAMAccountName=%(user)s)")

# Make selected group a superuser in Weblate
AUTH_LDAP_USER_FLAGS_BY_GROUP = {
    # is_superuser means user has all permissions
    "is_superuser": "CN=weblate_AdminUsers,OU=Groups,DC=example,DC=com",
}

# Map groups from AD to Weblate
AUTH_LDAP_GROUP_SEARCH = LDAPSearch("OU=Groups,DC=example,DC=com", ldap.
↳SCOPE_SUBTREE, "(objectClass=group)")
AUTH_LDAP_GROUP_TYPE = NestedActiveDirectoryGroupType()
AUTH_LDAP_FIND_GROUP_PERMS = True

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

🔗:

Django Authentication Using LDAP Authentication

CAS authentication

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:

```
# 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)

```

from django_cas_ng.signals import cas_user_authenticated
from django.dispatch import receiver
@receiver(cas_user_authenticated)
def update_user_email_address(sender, user=None, attributes=None,
↪**kwargs):
    # If your CAS server does not always include the email attribute
    # you can wrap the next two lines of code in a try/catch block.
    user.email = attributes['email']
    user.save()

```

??:

Django CAS NG

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.

??:

LDAP authentication CAS authentication

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:

```

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

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

```

??????

3.0 ??: Weblate 3.0 Django Weblate
Weblate
Weblate
Weblate
Weblate

???

Weblate ???

Weblate REQUIRE_LOGIN REGISTRATION_OPEN

acl

Restricted access

django

Restricted access

acl

django/admin/ URL

acl

acl

acl



WeblateOrg / Manage users

Users												
Username	Full name	E-mail	Last login	Administration	Billing	Glossary	Languages	Memory	Screenshots	Template	Translate	VCS
testuser	Weblate Test	weblate@example.org	14 seconds ago	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>						

Once all its permissions are removed, the user will be removed from the project.

Add a user

User to add

Please type in an existing Weblate account name or e-mail address.

Add

Invite new user

E-mail

Username

Username may only contain letters, numbers or the following characters: @ . + - _

Full name

Invite

Administration
Administration, Manage screenshots
Administration, Manage screenshots
Administration, Manage screenshots
Administration, Edit source
Administration
Administration, Edit source, Power user, Review strings, Translate
Administration, Edit source, Power user, Review strings, Translate
Administration, Review strings
Administration, Review strings
Administration, Edit source, Power user
Administration, Edit source, Power user, Review strings, Translate
Add suggestion, Administration, Edit source, Power user, Review strings, Translate
Administration
Administration, Edit source, Power user, Review strings, Translate
Administration, Manage languages, Power user
Administration, Manage languages
Administration, Manage languages
Administration, Manage languages
Administration
Administration, Edit source, Power user, Review strings, Translate
Administration, Edit source, Power user, Review strings, Translate
Access repository, Administration, Manage repository, Power user
Administration, Manage repository
Administration, Manage repository
Administration, Manage repository
upstream Administration, Administration, Manage repository, Power user
Administration, Manage repository

Administration
Administration

Webplate

Translations

```
setupgroups ANONYMOUS_USER_NAME
Add suggestion Access repository
none
Power user
Review strings
Administration
```

Note: Weblate *URL*.

Translations

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 the 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*.

Note: You can share cloned VCS repositories using *Weblate URL*. Using this feature is highly recommended when you have many components sharing the same VCS. It improves performance and decreases required disk space.

Adding translation projects and components

3.2 An interface for adding projects and components is included, and you no longer have to use *The Django admin interface*.

3.4 The process of adding components is now multi staged, with automated discovery of most parameters.

Based on your permissions, new translation projects and components can be created. It is always permitted for users with the *Add new projects* permission, and if your instance uses billing (e.g. like <https://hosted.weblate.org/> see **2.2**), 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:

Webate Dashboard Projects Languages Checks + Add

Your profile / Billing

Billing plan	
Current plan	Basic plan (Active)
Monthly price	19 EUR
Yearly price	199 EUR
Strings limit	Used 0 <input checked="" type="checkbox"/>
Languages limit	Used 0 <input checked="" type="checkbox"/>
Last invoice	2020-10-14 - 2020-10-16
Projects limit	Used 0 of 1 <input type="checkbox"/>
Projects	<div style="background-color: #fff9c4; padding: 5px; text-align: center;">No projects currently assigned!</div> <div style="text-align: center; margin-top: 10px;"> Add new translation project </div>
Terminate billing plan	

Invoices		
Invoice period	Invoice amount	Download invoice
10/14/2020 - 10/16/2020	19.0 EUR	Not available

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:

Add new translation project

Project name ⓘ

 Display name

URL slug ⓘ

 Name used in URLs and filenames.

Project website ⓘ

 Main website of translated project.

Mailing list ⓘ

 Mailing list for translators.

Translation instructions ⓘ

 You can use Markdown and mention users by @username.

Billing ⓘ

Save

Powered by Weblate 4.3 [About Weblate](#) [Legal](#) [Contact](#) [Documentation](#) [Donate to Weblate](#)

After creating the project, you are taken directly to the project page:

WeblateOrg translated 100%

Components Languages Info Search Glossaries Insights Files Tools Manage Share Watching

Nothing to list here.

Add new translation component

Powered by Weblate 4.3 [About Weblate](#) [Legal](#) [Contact](#) [Documentation](#) [Donate to Weblate](#)

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:

Creates component from remote version control repository.

Creates additional component to existing one by choosing different files.

Creates additional component to existing one, just for different branch.

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 [API](#).

Upload single document and translate that.

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:

The screenshot shows the Weblate web interface. At the top is a dark navigation bar with the Weblate logo, 'Dashboard', 'Projects', 'Languages', and 'Checks' menus, and a '+ Add' button. Below the navigation bar is a 'Create component' section with four tabs: 'From version control' (active), 'Upload translations files', 'Translate document', and 'Start from scratch'. The main content area is titled 'Create a new translation component from remote version control system repository.' It contains several form fields: 'Component name' (with a help icon) containing 'Language names', 'Display name' (empty), 'URL slug' (with a help icon) containing 'language-names', 'Project' (dropdown menu) containing 'WeblateOrg', 'Source language' (dropdown menu) containing 'English', 'Version control system' (dropdown menu) containing 'Git', 'Source code repository' (text input) containing 'https://github.com/WeblateOrg/demo.git', and 'Repository branch' (text input) which is empty. A 'Continue' button is at the bottom of the form. At the very bottom of the page is a footer: 'Powered by Weblate 4.3 About Weblate Legal Contact Documentation Donate to Weblate'.

On the next page, you are presented with a list of discovered translatable resources:

The screenshot shows a modal dialog titled 'Add new translation component' with a close icon. The dialog has a section 'Choose translation files to import' with a help icon. Below this section are four radio button options: 'Specify configuration manually', 'File format Android String Resource, Filemask app/src/main/res/values-*/strings.xml', 'File format gettext PO file, Filemask weblate/langdata/locale/*/LC_MESSAGES/django.po', and 'File format gettext PO file, Filemask weblate/locale/*/LC_MESSAGES/django.po'. A 'Continue' button is at the bottom of the dialog. At the very bottom of the page is a footer: 'Powered by Weblate 4.3 About Weblate Legal Contact Documentation Donate to Weblate'.

As a last step, you review the translation component info and fill in optional details:

Create component

Detected license as MIT, please check whether it is correct.

Add new translation component

Project
WeblateOrg

Component name
Language names
Display name

URL slug
language-names
Name used in URLs and filenames.

Version control system
Git
Version control system to use to access your repository containing translations. You can also choose additional integration with third party providers to submit merge requests.

Source code repository
https://github.com/WeblateOrg/demo.git
URL of a repository, use weblate://project/component to share it with other component.

Repository branch

Repository branch to translate

Repository push URL

URL of a push repository, pushing is turned off if empty.

Push branch

Branch for pushing changes, leave empty to use repository branch

Repository browser
https://github.com/WeblateOrg/demo/blob/{branch}/{filename}#L{line}
Link to repository browser, use {branch} for branch, {filename} and {line} as filename and line placeholders.

File format
gettext PO file

Filemask
weblate/langdata/locale/*/LC_MESSAGES/django.po
Path of files to translate relative to repository root, use * instead of language code, for example: po/*po or locale/*/LC_MESSAGES/django.po.

Monolingual base language file

Filename of translation base file, containing all strings and their source; it is recommended for monolingual translation formats.

Edit base file
Whether users will be able to edit the base file for monolingual translations.

Intermediate language file

Filename of intermediate translation file. In most cases this is a translation file provided by developers and is used when creating actual source strings.

Template for new translations
weblate/langdata/locale/django.pot
Filename of file used for creating new translations. For gettext choose .pot file.

Translation license
GNU General Public License v3.0 or later

Adding new translation
Create new language file
How to handle requests for creating new translations.

Language code style
Default based on the file format
Customize language code used to generate the filename for translations created by Weblate.

Language filter
^(cs|he|hu)\$
Regular expression used to filter translation when scanning for filemask.

Source language
English
Language used for source strings in all components

You will be able to edit more options in the component settings after creating it.

Save

??:

The Django admin interface [Project configuration](#) [Component configuration](#)

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 [Memory Management](#).

These basic attributes set up and inform translators of a project:

????????

Verbose project name, used to display the project name.

Project slug

Project name suitable for URLs.

???????? Web ???

URL where translators can find more info about the project.

????????

Mailing list where translators can discuss or comment translations.

????????

URL to more site with more detailed instructions for translators.

Set Language-Team header

Whether Weblate should manage the `Language-Team` header (this is a *GNU gettext* only feature right now).

????????

Whether to use shared translation memory, see [????????](#) for more details.

????????

Whether to contribute to shared translation memory, see [????????](#) for more details.

???????

Configure per project access control, see [????????](#) for more details.

Default value can be changed by `DEFAULT_ACCESS_CONTROL`.

enable_review_workflow_translations

Enable review workflow for translations, see [enable_review_workflow_translations](#).

enable_review_workflow_source_strings

Enable review workflow for source strings, see [enable_review_workflow_source_strings](#).

unauthenticated_repositories

Whether unauthenticated [repositories](#) are to be used for this repository.

??:

[Bilingual and monolingual formats](#) [Language definitions](#)

language_codes_mapping

Define language codes mapping when importing translations into Weblate. Use this when language codes are inconsistent in your repositories and you want to get a consistent view in Weblate.

The typical use case might be mapping American English to English: `en_US:en`

Multiple mappings to be separated by comma: `en_GB:en, en_US:en`

???: 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.

??:

Parsing language codes

Component configuration

A component is a grouping of something for translation. You enter a VCS repository location and file mask for which files you want translated, and Weblate automatically fetches from this VCS, and finds all matching translatable files.

You can find some examples of typical configurations in the [Supported file formats](#).

??: 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:

verbose_component_name

Verbose component name, used to display the component name.

Component slug

Component name suitable for URLs.

Component project

Project configuration where the component belongs.

`weblate_vcs`

VCS to use, see `weblate_vcs` for details.

`weblate_vcs_repo`

VCS repository used to pull changes.

`weblate_vcs_urls`:

See `weblate_vcs_urls` for more details on specifying URLs.

`weblate_vcs_urls`: This can either be a real VCS URL or `weblate://project/component` indicating that the repository should be shared with another component. See *Weblate URL* for more details.

`weblate_vcs_push_url` URL

Repository URL used for pushing. This setting is used only for *Git* and *Mercurial* and push support is turned off for these when this is empty.

`weblate_vcs_push_urls`:

See `weblate_vcs_push_urls` for more details on how to specify a repository URL and *Pushing changes from Weblate* for more details on pushing changes from Weblate.

`weblate_vcs_repo_browser`

URL of repository browser used to display source files (location of used messages). When empty, no such links will be generated. You can use *Template markup*.

For example on GitHub, use something like: `https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename}}#L{{line}}`

In case your paths are relative to different folder, you might want to strip leading directory by `parent-dir` filter (see *Template markup*): `https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename|parentdir}}#L{{line}}`

`weblate_vcs_export_url` URL

URL where changes made by Weblate are exported. This is important when *Web* `weblate_vcs_exporter` is not used, or when there is a need to manually merge changes. You can use *Git exporter* to automate this for Git repositories.

????????

Which branch to checkout from the VCS, and where to look for translations.

????? push

Branch for pushing changes, leave empty to use [????????](#).

?: This is currently only supported for Git, GitLab and GitHub, it is ignored for other VCS integrations.

File mask

Mask of files to translate, including path. It should include one "*" replacing language code (see [Language definitions](#) for info on how this is processed). In case your repository contains more than one translation file (e.g. more gettext domains), you need to create a component for each of them.

For example `po/* .po` or `locale/*/LC_MESSAGES/django.po`.

In case your filename contains special characters such as `[,]`, these need to be escaped as `[[]` or `[]]`.

?:

[Bilingual and monolingual formats](#) What does mean "There are more files for the single language (en)"?

????????

Base file containing string definitions for *Monolingual components*.

?:

[Bilingual and monolingual formats](#) What does mean "There are more files for the single language (en)"?

????????

Whether to allow editing the base file for *Monolingual components*.

????????

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 translation is based on this file, but all others are based on [????????](#). In case the string is not translated in source translation, translating to other languages is prohibited. This provides [????????](#).

?:

[????????](#) [Bilingual and monolingual formats](#) What does mean "There are more files for the single language (en)"?

????????

Base file used to generate new translations, e.g. `.pot` file with gettext.

?: 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.

?:

[Adding new translations](#) [????????](#) [Bilingual and monolingual formats](#) What does mean "There are more files for the single language (en)"?

??????

Translation file format, see also *Supported file formats*.

????????????????

Email address used for reporting upstream bugs. This address will also receive notification about any source string comments made in Weblate.

????????????

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`.

??????

Whether translation suggestions are accepted for this component.

??????

Turns on votecasting for suggestions, see [??????](#).

????????

Automatically accept voted suggestions, see [??????](#).

??????

???????? Weblate [????????](#) [????????](#)

??????

List of checks which can not be ignored, see [??????](#).

????????

License of the translation (does not need to be the same as the source code license).

????????

????????

????????

How to handle requests for creation of new languages. Available options:

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.

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).

User can select language and Weblate automatically creates the file for it and translation can begin.

There will be no option for user to start new translation.

??:

Adding new translations.

????? ?????

Customize language code used to generate the filename for translations created by Weblate, see *Adding new translations* for more details.

????????

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_ADD_MESSAGE*?*DEFAULT_ADDON_MESSAGE*?*DEFAULT_COMMIT_MESSAGE*?*DEFAULT_*

????????

Name of the committer used for Weblate commits, the author will always be the real translator. On some VCSs this might be not supported.

Default value can be changed by *DEFAULT_COMMITTER_NAME*.

????????????????

Email of committer used for Weblate commits, the author will always be the real translator. On some VCSs this might be not supported. The default value can be changed in *DEFAULT_COMMITTER_EMAIL*.

????????????????

Whether committed changes should be automatically pushed to the upstream repository. When enabled, the push is initiated once Weblate commits changes to its internal repository (see *Lazy commits*). To actually enable pushing *Repository push URL* has to be configured as well.

COMMIT_PENDING_HOURS

Sets how old changes (in hours) are to get before they are committed by background task or `commit_pending` management command. All changes in a component are committed once there is at least one older than this period.

Default value can be changed by `COMMIT_PENDING_HOURS`.

LOCK_ON_REPO_ERROR

Enables locking the component on repository error (failed pull, push or merge). Locking in this situation avoids adding another conflict which would have to be resolved manually.

The component will be automatically unlocked once there are no repository errors left.

LANGUAGE

Language used for source strings. Change this if you are translating from something else than English.

NOTE: In case you are translating bilingual files from English, but want to be able to do fixes in the English translation as well, you might want to choose *English (Developer)* as a source language. To avoid conflict between name of the source language and existing translation.

For monolingual translations, you can use intermediate translation in this case, see [intermediate translation](#).

LANG_FILTER

Regular expression used to filter the translation when scanning for filemask. This can be used to limit the list of languages managed by Weblate.

NOTE: You need to list language codes as they appear in the filename.

Some examples of filtering:

Filter description	Filter
Selected languages only	<code>^(cs de es)\$</code>
Exclude languages	<code>^(?! (it fr) \$) .+\$</code>
Exclude non language files	<code>^(?! (blank) \$) .+\$</code>
Include all files (default)	<code>^[^.] +\$</code>

STRING_VARIANTS

Regular expression used to determine the variants of a string, see *String variants*.

NOTE: Most of the fields can be edited by project owners or managers, in the Weblate interface.

NOTE:

Does Weblate support other VCSes than Git and Mercurial? [Translation component alerts](#)

????

??

Restricted access

By default the component is visible to anybody who has access to the project, even if the person can not perform any changes in the component. This makes it easier to keep translation consistency within the project.

Enable this in case you want to grant access to this component explicitly - the project level permissions will not apply and you will have to specify component or component list level permission in order to grant access.

Default value can be changed by `DEFAULT_RESTRICTED_COMPONENT`.

!!!: This applies to project managers as well - please make sure you will not loose access to the component after toggling the status.

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

????????????

????????????

Executing scripts from addon

There following variables are available in the component templates:

??????

????

????????????

Component slug

??????????

Project slug

Translation URL

??????????

Translation stats, this has further attributes, examples below.

Total strings count

Count of strings needing review

Percent of strings needing review

Translated strings count

Translated strings percent

Number of strings with failing checks

Percent of strings with failing checks

Author of current commit, available only in the commit scope.

Name of currently executed addon, available only in the addon commit message.

The following variables are available in the repository browser or editor templates:

current branch

line in file

filename, you can also strip leading parts using the `parentdir` filter, for example `{{ filename|parentdir }}`

You can combine them with filters:

```
{{ component|title }}
```

You can use conditions:

```
{% if stats.translated_percent > 80 %}Well translated!{% endif %}
```

There is additional tag available for replacing characters:

```
{% replace component "-" " " %}
```

You can combine it with filters:

```
{% replace component|capfirst "-" " " %}
```

There are also additional filter to manipulate with filenames:

```
Directory of a file: {{ filename|dirname }}  
File without extension: {{ filename|stripext }}  
File in parent dir: {{ filename|parentdir }}  
It can be used multiple times: {{ filename|parentdir|parentdir }}
```

...and other Django template features.

Importing speed

Fetching VCS repository and importing translations to Weblate can be a lengthy process, depending on size of your translations. Here are some tips:

Optimize configuration

The default configuration is useful for testing and debugging Weblate, while for a production setup, you should do some adjustments. Many of them have quite a big impact on performance. Please check *Production setup* for more details, especially:

Configure Celery for executing background tasks (see *Background tasks using Celery*)

Enable caching

Use a powerful database engine

Disable debug mode

Check resource limits

If you are importing huge translations or repositories, you might be hit by resource limitations of your server.

Check the amount of free memory, having translation files cached by the operating system will greatly improve performance.

Disk operations might be bottleneck if there is a lot of strings to process—the disk is pushed by both Weblate and the database.

Additional CPU cores might help improve performance of background tasks (see *Background tasks using Celery*).

Disable unneeded checks

Some quality checks can be quite expensive, and if not needed, can save you some time during import if omitted. See *CHECK_LIST* for info on configuration.

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 [gettext](#) 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`.

??:

Management commands [gettext](#)

Language definitions

To present different translations properly, info about language name, text direction, plural definitions and language code is needed. Definitions for about 350 languages are included.

Parsing language codes

While parsing translations, Weblate attempts to map language code (usually the ISO 639-1 one) to any existing language object.

You can further adjust this mapping at project level by [gettext](#).

If no exact match can be found, an attempt will be made to best fit it into an existing language (e.g. ignoring the default country code for a given language—choosing `cs` instead of `cs_CZ`).

Should that also fail, a new language definition will be created using the defaults (left to right text direction, one plural) and naming of the language as `xx_XX` (*generated*). You might want to change this in the admin interface later, (see [Changing language definitions](#)) and report it to the issue tracker (see [Weblate](#) [gettext](#)).

???: In case you see something unwanted as a language, you might want to adjust [gettext](#) to ignore such file when parsing translations.

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.

Language definitions

Each language consists of following fields:

?????

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.

??:

Parsing language codes

???

Visible name of the language. The language names included in Weblate are also being localized depending on user interface language.

??????

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.

??????

Gettext compatible plural formula used to determine which plural form is used for given count.

?:

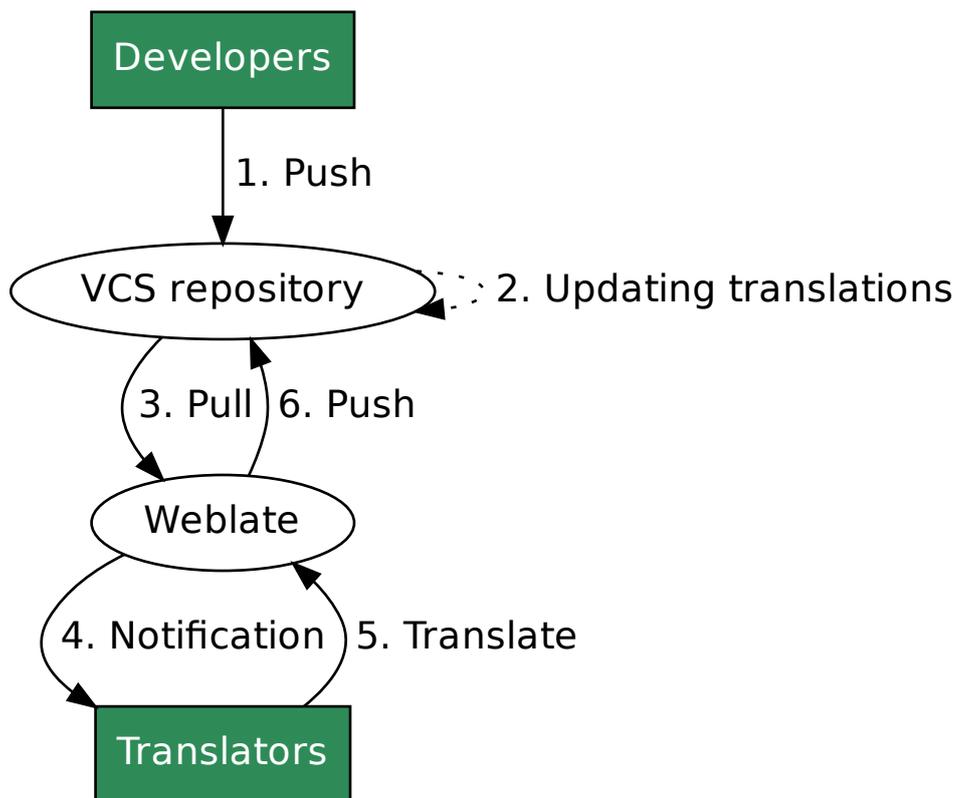
???, GNU gettext utilities: Plural forms, Language Plural Rules by the Unicode Consortium

Web ??????????

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.

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*).



Updating repositories

You should set up some way of updating backend repositories from their source.

Use [XXXXXXXXXX](#) to integrate with most of common code hosting services

Manually trigger update either in the repository management or using [API](#) or [Weblate XXXXXXXXXX](#)

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 [XXXXXXXXXX](#).

Avoiding merge conflicts

The merge conflicts from Weblate arise when same file was changed both in Weblate and outside it. There are two approaches to deal with that - avoid edits outside Weblate or integrate Weblate into your updating process, so that it flushes changes prior to updating the files outside Weblate.

The first approach is easy with monolingual files - you can add new strings within Weblate and leave whole editing of the files there. For bilingual files, there is usually some kind of message extraction process to generate translatable files from the source code. In some cases this can be split into two parts - one for the extraction generates template (for example gettext POT is generated using `xgettext`) and then further process merges it into actual translations (the gettext PO files are updated using `msgmerge`). You can perform the second step within Weblate and it will make sure that all pending changes are included prior to this operation.

The second approach can be achieved by using [API](#) to force Weblate to push all pending changes and lock the translation while you are doing changes on your side.

The script for doing updates can look like this:

```
# Lock Weblate translation
wlc lock
# Push changes from Weblate to upstream repository
wlc push
# Pull changes from upstream repository to your local copy
git pull
# Update translation files, this example is for Django
./manage.py makemessages --keep-pot -a
git commit -m 'Locale updates' -- locale
# Push changes to upstream repository
git push
# Tell Weblate to pull changes (not needed if Weblate follows your repo
# automatically)
wlc pull
# Unlock translations
wlc unlock
```

If you have multiple components sharing same repository, you need to lock them all separately:

```
wlc lock foo/bar
wlc lock foo/baz
wlc lock foo/baj
```

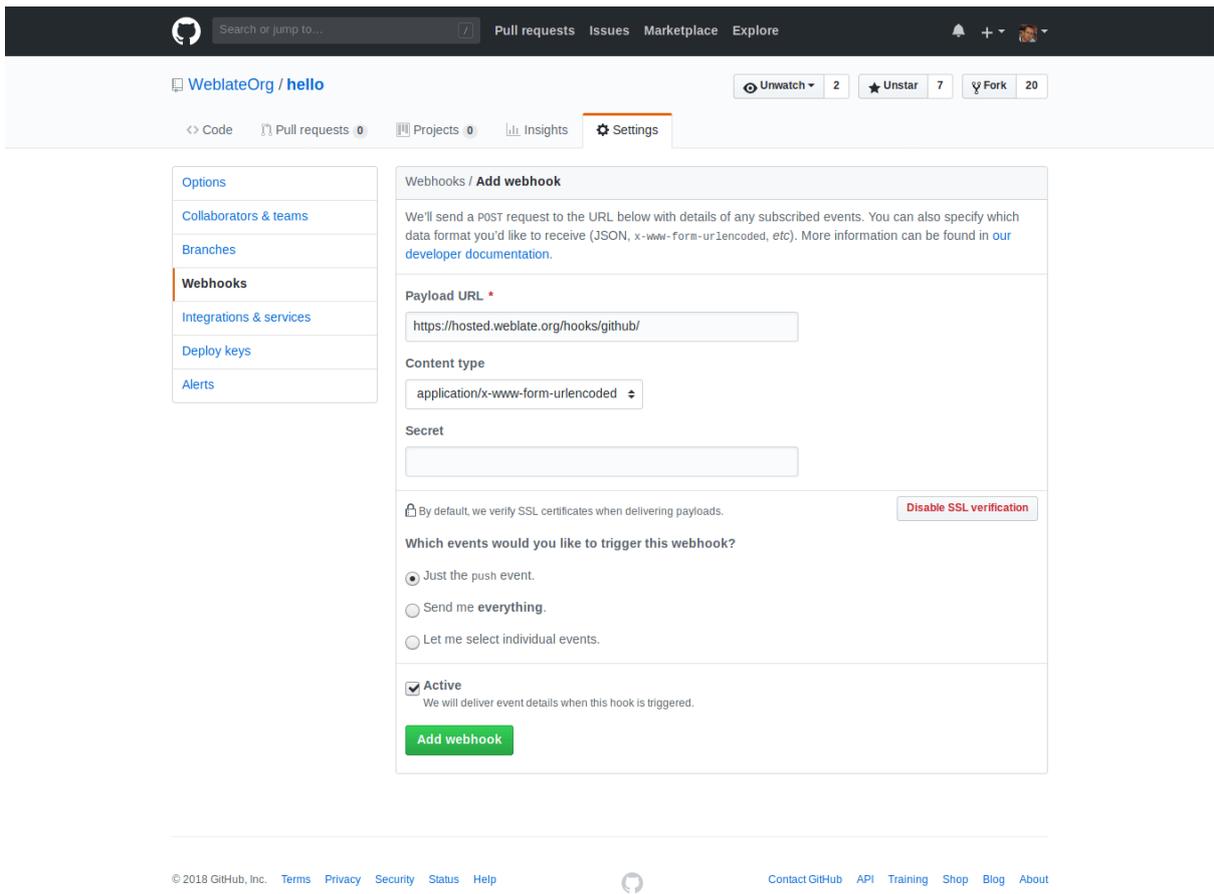
 The example uses [Weblate `weblate`](#), which needs configuration (API keys) to be able to control Weblate remotely. You can also achieve this using any HTTP client instead of `wlc`, e.g. `curl`, see [API](#).

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/`.

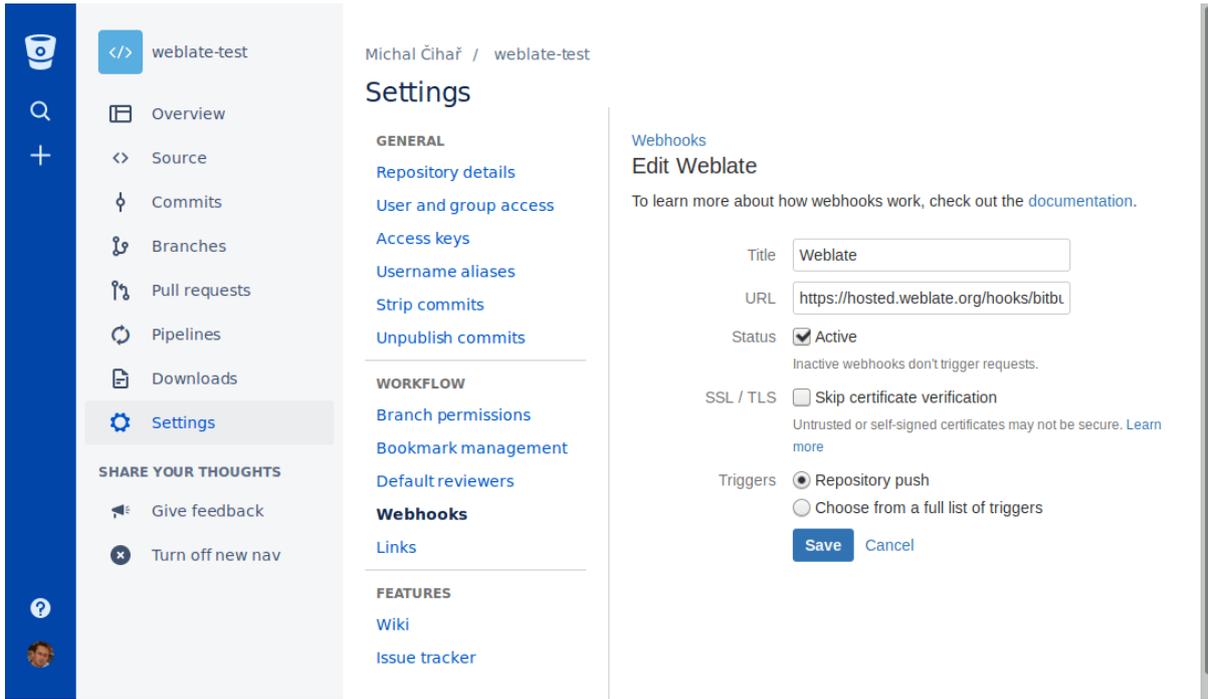
You can leave other values at default settings (Weblate can handle both content types and consumes just the *push* event).

??:

`POST /hooks/github/ 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/`).



🔗:

`POST /hooks/bitbucket/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/`).

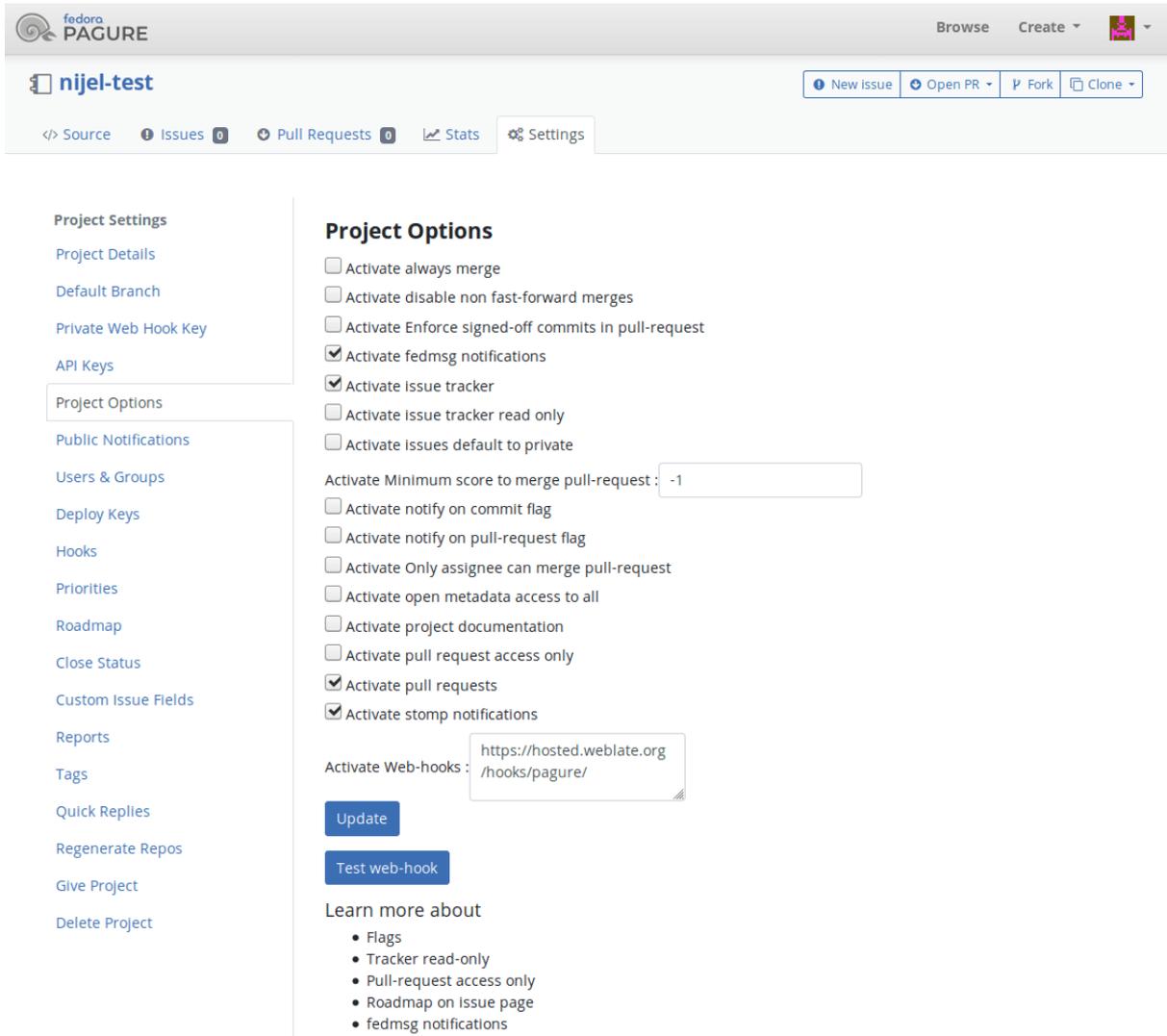
🔗:

`POST /hooks/gitlab/Hosted Weblate`

Automatically receiving changes from Pagine

3.3

Weblate has support for Pagine hooks, add a webhook with destination to `/hooks/pagine/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/pagine/`). This can be done in *Activate Web-hooks* under *Project options*:



🔗:

`POST /hooks/pagure/🔗Hosted Weblate 🔗🔗🔗🔗🔗🔗🔗🔗🔗`

Automatically receiving changes from Azure Repos

🔗🔗🔗🔗 3.8 🔗🔗🔗.

Weblate has support for Azure Repos web hooks, add a webhook for *Code pushed* event with destination to `/hooks/azure/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/azure/`). This can be done in *Service hooks* under *Project settings*.

🔗🔗:

Web hooks in Azure DevOps manual 🔗`POST /hooks/azure/🔗Hosted Weblate 🔗🔗🔗🔗🔗🔗🔗🔗`

Automatically receiving changes from Gitea Repos

3.9

Weblate has support for Gitea webhooks, add a *Gitea Webhook* for *Push events* event with destination to `/hooks/gitea/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/gitea/`). This can be done in *Webhooks* under repository *Settings*.

Note:

Webhooks in Gitea manual `POST /hooks/gitea/` Hosted Weblate

Automatically receiving changes from Gitee Repos

3.9

Weblate has support for Gitee webhooks, add a *WebHook* for *Push* event with destination to `/hooks/gitee/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/gitee/`). This can be done in *WebHooks* under repository *Management*.

Note:

Webhooks in Gitee manual `POST /hooks/gitee/` 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`.

Pushing changes from Weblate

Each translation component can have a push URL set up (see `URL`), 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 `push`). 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 `push` used, more details are found in that chapter.

In case you do not want direct pushes by Weblate, there is support for *GitHub*, *GitLab* pull requests or *Gerrit* reviews, you can activate these by choosing *GitHub*, *GitLab* or *Gerrit* as `push` in *Component configuration*.

Overall, following options are available with Git, GitHub and GitLab:

Desired setup	<code>push</code>	<code>URL</code>	<code>push</code>
No push	<i>Git</i>	<i>empty</i>	<i>empty</i>
Push directly	<i>Git</i>	SSH URL	<i>empty</i>
<code>push</code>	<i>Git</i>	SSH URL	Branch name
GitHub pull request from fork	<i>GitHub</i>	<i>empty</i>	<i>empty</i>
GitHub pull request from branch	<i>GitHub</i>	SSH URL ¹	Branch name
GitLab merge request from fork	<i>GitLab</i>	<i>empty</i>	<i>empty</i>
GitLab merge request from branch	<i>GitLab</i>	SSH URL ¹	Branch name

Note: You can also enable automatic pushing of changes after Weblate commits, this can be done in `push`.

Note:

See `push` for setting up SSH keys, and *Lazy commits* for info about when Weblate decides to commit changes.

Can be empty in case `push` supports pushing.

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:

Require pull request reviews before merging

When enabled, all commits must be made to a non-protected branch and submitted via a pull request with the required number of approving reviews and no changes requested before it can be merged into a branch that matches this rule.

Required approving reviews: 1 ▾

Dismiss stale pull request approvals when new commits are pushed

New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

Require review from Code Owners

Require an approved review in pull requests including files with a designated code owner.

Restrict who can dismiss pull request reviews

Specify people or teams allowed to dismiss pull request reviews.

🔍 Search for people or teams

People and teams that can dismiss reviews.



Organization and repository administrators

These members can always dismiss.



weblate
Weblate push user



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.

⚠️: Rebasing can cause you trouble in case of complicated merges, so carefully consider whether or not you want to enable them.

Interacting with others

Weblate makes it easy to interact with others using its API.

🔗:

[API](#)

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 [\[?\]\[?\]\[?\]\[?\]](#)).

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 *Age of changes to commit* on *Component configuration*.

[?][?]: Commits are created for every component. So in case you have many components you will still see lot of commits. You might utilize *Git [?][?][?][?][?][?][?]* addon in that case.

If you want to commit changes more frequently and without checking of age, you can schedule a regular task to perform a commit:

```
CELERY_BEAT_SCHEDULE = {
    # Unconditionally commit all changes every 2 minutes
    "commit": {
        "task": "weblate.trans.tasks.commit_pending",
        # Ommiting hours will honor per component settings,
        # otherwise components with no changes older than this
        # won't be committed
        "kwargs": {"hours": 0},
        # How frequently to execute the job in seconds
        "schedule": 120,
    }
}
```

Processing repository with scripts

The way to customize how Weblate interacts with the repository is [\[?\]\[?\]\[?\]](#). Consult *Executing scripts from addon* for info on how to execute external scripts through addons.

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 translation propagation 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.

[?]: 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 [consistency check](#) fires whenever the strings are different. You can utilize this to review such differences manually and choose the right translation.

Automatic translation

Automatic translation based on different components can be way to synchronize the translations across components. You can either trigger it manually (see [manual trigger](#)) or make it run automatically on repository update using add-on (see [automatic translation](#)).

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.

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.

Contributor license agreement

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:

Language	Translated	Untranslated	Untranslated words	Checks	Suggestions	Comments
Czech 🇨🇪 GPL-3.0	✓					
Hebrew 🇮🇱 GPL-3.0	✓					
Hungarian 🇭🇺 GPL-3.0	81%	4	5			
English 🇬🇧 GPL-3.0	✓					

The entered text is formatted into paragraphs and external links can be included. HTML markup can not be used.

User licenses

Any user can review all translation licenses of all public projects on the instance from their profile:

The screenshot shows the Weblate user interface. At the top is a dark navigation bar with the Weblate logo, 'Dashboard', 'Projects', 'Languages', and 'Checks' menus. On the right are icons for a wrench, '+ Add', a user profile, and a menu. Below the navigation bar is a section for 'Your profile' with tabs for 'Languages', 'Preferences', 'Notifications', 'Account', 'Profile', 'Licenses' (which is active), 'Audit log', and 'API access'. The 'Licenses' section has a title 'Licenses' and contains the following text: '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:'. A bulleted list follows: '• WeblateOrg/Language names'. Below this is a section titled 'Licenses for individual translations' which lists: 'GNU General Public License v3.0 or later' with a link to 'GPL-3.0', 'WeblateOrg/Djangojs', 'WeblateOrg/Django', and 'WeblateOrg/Language names'; and 'MIT License' with a link to 'MIT' and 'WeblateOrg/Android'.

Powered by Weblate 4.3 [About Weblate](#) [Legal](#) [Contact](#) [Documentation](#) [Donate to Weblate](#)

??????

???????

Everyone can add suggestions by default, to be accepted by signed in users. Suggestion voting can be used to make use of a string when more than signed in user agrees, by setting up the *Component configuration* configuration with *Suggestion voting* to turn on voting, and *Autoaccept suggestions* to set a threshold for accepted suggestions (this includes a vote from the user making the suggestion if it is cast).

???: 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* privilege (see [???????](#)).

You can combine these with [???????](#) 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.

Additional info on source strings

Enhance the translation process with info available in the translation files. This includes explanation, string priority, check flags, or providing visual context. All these features can be set in the *Reviewing strings*:

The screenshot shows the Weblate interface with a modal dialog titled "Edit additional string info". The dialog contains the following sections:

- Explanation**: A text input field with a placeholder "Additional explanation to clarify meaning or usage of the string."
- Labels**: Two checkboxes, "Current sprint" (checked) and "Next sprint" (unchecked). Below them is the text "Additional labels can be defined in the project settings."
- Translation flags**: A text input field with a placeholder "Additional comma-separated flags to influence quality checks. Possible values can be found in the documentation."

The background interface shows the translation details for the key "dow_monday" in English, with a "Save" button and a "Suggest" button. A table of nearby strings is visible below the dialog:

Key	Translation	Status
dow_monday	Monday	✓
dow_monday_min	M	✓
dow_monday_short	Mon	✓

Access this directly from the translation interface by clicking the "Edit" icon next to *Screenshot context* or *Flags*.

Translation

Explanation
Help text for automatic translation tool

English
Automatic translation via machine translation uses active machine translation engines to get the best possible translations and applies them in this project.

Czech
Automatický překlad prostřednictvím strojového překladu používá aktivní enginy strojového překladu pro získání nejlepších možných překladů a použije je na tento projekt.

Needs editing (169/1570)

Save Suggest Skip

Glossary

English	Czech
machine translation	strojový překlad
project	projekt

+ Add term to glossary

Source information

Screenshot context
No screenshot currently associated.

Explanation
Help text for automatic translation tool

Labels
No labels currently set.

Flags
No flags currently set.

Source string location
weblate/templates/translation.html:212

String age
a second ago

Source string age
a second ago

Translation file
weblate/locale/cs/LC_MESSAGES/django.po, string 11

Nearby strings 26 Comments Automatic suggestions Other languages History

Context	English	Czech	State
Files		Soubory	✓
Automatic translation		Automatický překlad	✓
Add new translation string		Add new translation string	✓
Translation status		Stav překladu	✓
%(count)s word		%(count)s slovo	✓
Other components		Další součásti	✓
Translation file		Soubor s překladem	✓
Download		Stáhnout	✓
Browse all translation changes		Procházet všechny změny v překladu.	✓
	Automatic translation takes existing translations in this project and applies them to the current component. It can be used to push translations to a different branch, to fix inconsistent translations or to translate a new component using translation memory.	Automatický překlad použije stávající překlady v projektu na tuto součást. Může být užitečný pro sloučení překladů z jiné větve, opravu nekonzistentních překladů nebo překlad nové součásti pomocí překladové paměti.	✓
	Automatic translation via machine translation uses active machine translation engines to get the best possible translations and applies them in this project.	Automatický překlad prostřednictvím strojového překladu používá aktivní enginy strojového překladu pro získání nejlepších možných překladů a použije je na tento projekt.	✓
	You can add new translation string here, it will automatically appear in all translations.	Zde můžete přidat nový řetězec k překladu, automaticky se objeví ve všech jazycích.	✓
	The uploaded file will be merged with the current translation. In case you want to overwrite already translated strings, don't forget to enable it.	Nahráný soubor bude sloučen se stávajícími překlady. Pokud chcete přepsat již přeložené řetězce, nepamenejte to povolit.	✓
	The uploaded file will be merged with the current translation.	Nahráný soubor bude sloučen se stávajícími překlady.	✓
	The fulltext search might not work properly as the fulltext index for this translation is not yet up to date.	Fulltextové vyhledávání nemusí fungovat správně, protože fulltextový index pro tento překlad ještě není plně zpracován.	✓
	Review	Kontrola	✓
	Review translations touched by other users.	Zkontrolovat překlady od ostatních uživatelů.	✓
	Start review	Začít kontrolu	✓
	Percent	Procenta	✓
	Total	Celkem	✓
	Failing check	Neúspěšných kontrol	✓
	Last activity	Poslední aktivita	✓
	Last change	Poslední změna	✓
	Last author	Poslední autor	✓
Question for a mathematics-based CAPTCHA, the %s is an arithmetic problem	What is %s?	Kolik to je?	✓
	The string uses three dots (...) instead of an ellipsis character (...)		ⓘ

Strings prioritization

2.0

String priority can be changed to offer higher priority strings for translation earlier by using the `priority` flag

This can be used to order the flow of translation in a logical manner.

:

2.4

3.3: Previously called *Quality checks flags*, it no longer configures only checks.

The default set of translation flags is determined by the translation *Component configuration* and the translation file. However, you might want to use it to customize this per source string.

:

4.1: In previous version this has been called extra context.

Use the explanation to clarify scope or usage of the translation. You can use Markdown to include links and other markup.

Visual context for strings

2.9

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:

Powered by Weblate 4.3 [About Weblate](#) [Legal](#) [Contact](#) [Documentation](#) [Donate to Weblate](#)

In addition to *Reviewing strings*, screenshots have a separate management interface under the *Tools* menu. Upload screenshots, assign them to source strings manually, or use optical character recognition to do so.

Once a screenshot is uploaded, this interface handles management and source string association:

Screenshot has been uploaded, you can now assign it to source strings.

Assigned source strings

Source string	Context	Location	Assigned screenshots	Actions
No source strings are currently assigned!				
Screenshot is shown to add visual context for all listed source strings.				

Assign source strings

Source string	Context	Location	Assigned screenshots	Actions
No new matching source strings found.				

Source string search

Image

Source string

Hello, world!

One
Orangutan has %d banana.

Other
Orangutan has %d bananas.

Try Weblate at <http://demo.weblate.org/>!

Thank you for using Weblate.

Screenshot is shown to add visual context for all listed source strings.

Edit screenshot

Screenshot name

Image
Currently: [screenshots/screenshot.png](#)
Change:
 No file chosen
Upload JPEG or PNG images up to 2000x2000 pixels.

Screenshot details

Created	now
Uploaded by	testuser
Language	English

Delete screenshot

Deleting screenshot will remove it from all associated source strings.

????
 ????
 ????
 ????
 ????
 ????
 ????
 ????
 ????
 ????
 ????
 ????
 ????
 ???? HTML? ????
 ?URL? ????
 ?XML? ????
 ?XML? ????
 ????
 ????
 ????
 ????
 ????
 ????
 ????
 ????
 ????

??: ???? ignore-* ????

 ???? *Component configuration* ???? GNU gettext?

????

???? 3.11 ??

Component configuration ? [????](#) ????
[???](#) ???? *Translation states*?

??????

???? 3.7 ??

???? [????](#) ???? [Weblate](#)
 ???? *Manage* ???? *Fonts* ????
 TrueType ??? OpenType ???? ????
 ???? ??:

Font group		
Name	default-font	
Default font	Source Sans Pro Bold	
Japanese	language override Droid Sans Fallback Regular	Remove
Korean	language override Droid Sans Fallback Regular	Remove
Delete		

Add language override

Language

Font

Save

Edit font group

Font group name

Identifier you will use in checks to select this font group. Avoid whitespaces and special characters.

Default font

Default font is used unless per language override matches.

Save

???? ?????:

Group name	Default font	Language overrides	
default-font	Source Sans Pro Bold	Japanese: Droid Sans Fallback Regular Korean: Droid Sans Fallback Regular	Edit

Add font group

Font group name

Identifier you will use in checks to select this font group. Avoid whitespaces and special characters.

Default font

.....
▼

Default font is used unless per language override matches.

Save

???? ??:

Font	
Font family	Droid Sans Fallback
Font style	Regular
File size	3939852
Created	now
Uploaded by	 testuser
Used in groups	
Delete	

Weblate ??:


```

"""Simple quality check example."""
from django.utils.translation import gettext_lazy as _
from weblate.checks.base import TargetCheck

class FooCheck (TargetCheck) :

    # Used as identifier for check, should be unique
    # Has to be shorter than 50 characters
    check_id = "foo"

    # Short name used to display failing check
    name = _("Foo check")

    # Description for failing check
    description = _("Your translation is foo")

    # Real check code
    def check_single(self, source, target, unit):
        return "foo" in target

```

??

???????????????????????? 2 ?????????????????????????????????

```

#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.
#
"""Quality check example for Czech plurals."""
from django.utils.translation import gettext_lazy as _
from weblate.checks.base import TargetCheck

class PluralCzechCheck (TargetCheck) :

    # 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

```

????

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*.

amaGama

Special installation of *tmserver* run by the authors of Virtaal.

Turn on this service by adding `weblate.machinery.tmserver.AmagamaTranslation` to `MT_SERVICES`.

??:

Installing amaGama?Amagama?amaGama Translation Memory

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`.

??:

`MT_APERTIUM_APY`, Apertium website, Apertium APY documentation

AWS

????? 3.1 ???.

Amazon Translate is a neural machine translation service for translating text to and from English across a breadth of supported languages.

1. Turn on this service by adding `weblate.machinery.aws.AWSTranslation` to `MT_SERVICES`.
2. Install the *boto3* module.
3. Configure Weblate.

??:

`MT_AWS_REGION`?`MT_AWS_ACCESS_KEY_ID`?`MT_AWS_SECRET_ACCESS_KEY` Amazon Translate Documentation

Baidu API machine translation

3.2

Machine translation service provided by Baidu.

This service uses an API and you need to obtain an ID and API key from Baidu to use it.

Turn on this service by adding `weblate.machinery.baidu.BaiduTranslation` to `MT_SERVICES` and set `MT_BAIDU_ID` and `MT_BAIDU_SECRET`.

:

`MT_BAIDU_ID` `MT_BAIDU_SECRET` [Baidu Translate API](#)

DeepL

2.20

DeepL is paid service providing good machine translation for a few languages. You need to purchase *DeepL API* subscription or you can use legacy *DeepL Pro (classic)* plan.

Turn on this service by adding `weblate.machinery.deepl.DeepLTranslation` to `MT_SERVICES` and set `MT_DEEPL_KEY`.

: In case you have subscription for CAT tools, you are supposed to use "v1 API" instead of default "v2" used by Weblate (it is not really an API version in this case). You can toggle this by `MT_DEEPL_API_VERSION`.

:

`MT_DEEPL_KEY` `MT_DEEPL_API_VERSION` [DeepL website](#) [DeepL pricing](#) [DeepL API documentation](#)

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`.

:

[Glosbe website](#)

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`.

:

`MT_GOOGLE_KEY` [Google translate documentation](#)

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.googlelev3.GoogleV3Translation` to `MT_SERVICES` and set

`MT_GOOGLE_CREDENTIALS`

`MT_GOOGLE_PROJECT`

If `location` fails, you may also need to specify `MT_GOOGLE_LOCATION`.

🔗:

`MT_GOOGLE_CREDENTIALS` `MT_GOOGLE_PROJECT` `MT_GOOGLE_LOCATION` [Google translate documentation](#)

Microsoft Cognitive Services Translator

🔗🔗🔗🔗 2.10 🔗🔗.

Machine translation service provided by Microsoft in Azure portal as a one of Cognitive Services.

Weblate implements Translator API V3.

To enable this service, add `weblate.machinery.microsoft.MicrosoftCognitiveTranslation` to `MT_SERVICES` and set `MT_MICROSOFT_COGNITIVE_KEY`.

Translator Text API V2

The key you use with Translator API V2 can be used with API 3.

Translator Text API V3

You need to register at Azure portal and use the key you obtain there. With new Azure keys, you also need to set `MT_MICROSOFT_REGION` to locale of your service.

🔗:

`MT_MICROSOFT_COGNITIVE_KEY`, `MT_MICROSOFT_REGION`, [Cognitive Services - Text Translation API, Microsoft Azure Portal](#)

Microsoft Terminology Service

🔗🔗🔗🔗 2.19 🔗🔗.

The Microsoft Terminology Service API allows you to programmatically access the terminology, definitions and user interface (UI) strings available in the Language Portal through a web service.

Turn this service on by adding `weblate.machinery.microsoftterminology.MicrosoftTerminologyService` to `MT_SERVICES`.

🔗:

[Microsoft Terminology Service API](#)

ModernMT

🔗🔗🔗🔗 4.2 🔗🔗.

Turn this service on by adding `weblate.machinery.modernmt.ModernMTTranslation` to `MT_SERVICES` and configure `MT_MODERNMT_KEY`.

🔗:

[ModernMT API](#), `MT_MODERNMT_KEY`, `MT_MODERNMT_URL`

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`.

??:

[MT_MYMEMORY_EMAIL](#) [MT_MYMEMORY_USER](#) [MT_MYMEMORY_KEY](#) [MyMemory website](#)

NetEase Sight API machine translation

[NetEase](#) 3.3 [???](#).

Machine translation service provided by NetEase.

This service uses an API, and you need to obtain key and secret from NetEase.

Turn on this service by adding `weblate.machinery.youdao.NeteaseSightTranslation` to `MT_SERVICES` and set `MT_NETEASE_KEY` and `MT_NETEASE_SECRET`.

??:

[MT_NETEASE_KEY](#) [MT_NETEASE_SECRET](#) [Netease Sight Translation Platform](#)

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/'
```

??:

[MT_TMSERVER](#) [tmserver](#) [Installing amaGama](#) [Amagama](#) [Amagama Translation Memory](#)

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`.

??:

[MT_YANDEX_KEY](#) [Yandex Translate API](#) [Powered by Yandex.Translate](#)

Youdao Zhiyun API machine translation

3.2

Machine translation service provided by Youdao.

This service uses an API, and you need to obtain an ID and an API key from Youdao.

Turn on this service by adding `weblate.machinery.youdao.YoudaoTranslation` to `MT_SERVICES` and set `MT_YOUDAO_ID` and `MT_YOUDAO_SECRET`.

Example:

```
MT_YOUDAO_ID=MT_YOUDAO_SECRET Youdao Zhiyun Natural Language Translation Service
```

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`.

Weblate Translation Memory

2.20

The `weblate.memory.machine` 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.

SAP Translation Hub

Machine translation service provided by SAP.

You need to have a SAP account (and enabled the SAP Translation Hub in the SAP Cloud Platform) to use this service.

Turn on this service by adding `weblate.machinery.saptranslationhub.SAPTranslationHub` to `MT_SERVICES` and set the appropriate access to either sandbox or the productive API.

Example: 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`.

Example:

```
MT_SAP_BASE_URL=MT_SAP_SANDBOX_APIKEY=MT_SAP_USERNAME=MT_SAP_PASSWORD=MT_SAP_USE_MT SAP Translation Hub API
```

Custom machine translation

You can also implement your own machine translation services using a few lines of Python code. This example implements machine translation in a fixed list of languages using `dictionary` Python module:

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
```

(2012-2020)

(???????????)

```
# 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, unit, user, ↵
↵search):
        """Return tuple with translations."""
        for t in dictionary.translate(text):
            yield {"text": t, "quality": 100, "service": self.name, "source
↵": text}
```

You can list own class in `MT_SERVICES` and Weblate will start using that.

????

???? 2.19 ??

Addons provide ways to customize translation workflow. They can be installed in the translation component view, and work behind the scenes. Addon management is available from the *Manage* ↓ *Addons* menu of each respective translation component for admins.

Installed addons
There are no addons currently installed.

Table with 2 columns: Addon Name and Install Button. Includes addons like Automatic translation, Language consistency, Component discovery, Bulk edit, Statistics generator, etc.

??????

????

????? 3.9 ???

??

This addon is triggered automatically when new strings appear in a component.

??:

????? Keeping translations same across components

JavaScript CDN

 4.2 .

JavaScript  HTML  CDN 

It can be used to localize static HTML pages or used to load localization in the JavaScript code.

Upon installation the addon generates unique URL for your component which you can include in the HTML documents to get them localized. See *Translating HTML and JavaScript using Weblate CDN* for more details.

Weblate CDN  [Translating HTML and JavaScript using Weblate CDN](#) Weblate CDN  [Weblate CDN !\[\]\(503b48e08530b707c81eec2a196ea5fc_img.jpg\) HTML !\[\]\(69dff6f10b26160d0a6faf8061e24e29_img.jpg\)](#)

1                

It creates empty translations in languages that have unadded components.
Missing languages are checked once every 24 hours and when a new language is added in Weblate.
Unlike most others, this addon affects the whole project.

: Auto-translate the newly added strings with .

It is triggered on every VCS update, and otherwise similar to the `import_project` management command. This way you can track multiple translation components within one VCS.
Create one main component least likely to disappear in the future, and others will employ *Weblate  URL* to it as a VCS configuration, and configure it to find all components in it.
The matching is done using regular expressions, where power is a tradeoff for complexity in configuration. Some examples for common use cases can be found in the addon 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:

Configure addon

Please review and confirm the matched components.

Component Matched files

Following components would be created

Djangojs	<ul style="list-style-type: none"> weblate/locale/he/LC_MESSAGES/djangojs.po (he) weblate/locale/hu/LC_MESSAGES/djangojs.po (hu) weblate/locale/cs/LC_MESSAGES/djangojs.po (cs)
Django	<ul style="list-style-type: none"> weblate/locale/he/LC_MESSAGES/django.po (he) weblate/locale/hu/LC_MESSAGES/django.po (hu) weblate/locale/cs/LC_MESSAGES/django.po (cs)

I confirm the above matches look correct

Regular expression to match translation files against

weblate/locale/(?P<language>[^\s]*)/LC_MESSAGES/(?P<component>[^\s]*)\.po

File format

gettext PO file

Customize the component name

{{ component|title }}

Define the monolingual base filename

Leave empty for bilingual translation files.

Define the base file for new translations

weblate/locale/{{ component }}.pot

Filename of file used for creating new translations. For gettext choose .pot file.

Language filter

^(cs|he|hu)\$

Regular expression to filter translation against when scanning for filemask.

Clone addons from the main component to the newly created ones

Remove components for inexistant files

The regular expression to match translation files has to contain two named groups to match component and language, some examples:

Regular expression	Example matched files	Description
<code>(?P<language>[^\s]*)/(?P<component>[^\s]*)\.po</code>	<ul style="list-style-type: none"> cs/application.po cs/website.po de/application.po de/website.po 	One folder per language containing translation files for components.
<code>locale/(?P<language>[^\s]*)/LC_MESSAGES/(?P<component>[^\s]*)\.po</code>	<ul style="list-style-type: none"> locale/cs/LC_MESSAGES/application.po locale/cs/LC_MESSAGES/website.po locale/de/LC_MESSAGES/application.po locale/de/LC_MESSAGES/website.po 	Usual structure for storing gettext PO files.
<code>src/locale/(?P<component>[^\s]*)\. (?P<language>[^\s]*)\.po</code>	<ul style="list-style-type: none"> src/locale/application.cs.po src/locale/website.cs.po src/locale/application.de.po src/locale/website.de.po 	Using both component and language name within filename.
<code>locale/(?P<language>[^\s]*)/(?P<component>[^\s]*)/(?P=language)\.po</code>	<ul style="list-style-type: none"> locale/cs/application/cs.po locale/cs/website/cs.po locale/de/application/de.po locale/de/website/de.po 	Using language in both path and filename.
<code>res/values-(?P<language>[^\s]*)/strings-(?P<component>[^\s]*)\.xml</code>	<ul style="list-style-type: none"> res/values-cs/strings-about.xml res/values-cs/strings-help.xml res/values-de/strings-about.xml res/values-de/strings-help.xml 	Android resource strings, split into several files.

You can use Django template markup in both component name and the monolingual base filename, for example:

`{{ component }}`

Component filename match

`{{ component|title }}`

Component filename with upper case first letter

Save

PO

PO

The PO file header will contain a list of contributors and years contributed:

```
# Michal Čihař <michal@cihar.com>, 2012, 2018, 2019, 2020.
# Pavel Borecki <pavel@example.com>, 2018, 2019.
# Filip Hron <filip@example.com>, 2018, 2019.
# anonymous <noreply@weblate.org>, 2019.
```

configure ALL_LINGUAS

Updates the ALL_LINGUAS variable in configure, configure.in or any configure.ac files, when a new translation is added.

gettext

gettext

It offers the following options:

Wrap lines at 77 characters and at newlines

no-wrap: By default gettext wraps lines at 77 characters and for newlines. With the `--no-wrap` parameter, it wraps only at newlines.

LINGUAS

LINGUAS

MO

PO MO

POT PO (msgmerge)

msgmerge POT PO

Git

Git

You can choose one of following modes:

3.4

1

3.5

Per author

4.1

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 `$(cat $(find . -name *.json) | jq -r 'select(contains("Co-authored-by:")) | del(.Co-authored-by) | tojson'`

```
$(cat $(find . -name *.json) | jq -r 'select(contains("Co-authored-by:")) | del(.Co-authored-by) | tojson'
```

Java `$(cat $(find . -name *.java) | sed -e 's/Co-authored-by: //g' | tr '\n' '\0'`

```
$(cat $(find . -name *.java) | sed -e 's/Co-authored-by: //g' | tr '\n' '\0'
```

`$(cat $(find . -name *.php) | sed -e 's/Co-authored-by: //g'`

```
$(cat $(find . -name *.php) | sed -e 's/Co-authored-by: //g'
```

```
$(cat $(find . -name *.php) | sed -e 's/Co-authored-by: //g'
```

This can be useful to remove old comments which might have become outdated. Use with care as comment being old does not mean they have lost their importance.

`$(cat $(find . -name *.php) | sed -e 's/Co-authored-by: //g'`

```
$(cat $(find . -name *.php) | sed -e 's/Co-authored-by: //g'
```

```
$(cat $(find . -name *.php) | sed -e 's/Co-authored-by: //g'
```

This can be very useful in connection with suggestion voting (see [suggestion voting](#)) to remove suggestions which don't receive enough positive votes in a given timeframe.

RESX `$(cat $(find . -name *.resx) | sed -e 's/Co-authored-by: //g'`

```
$(cat $(find . -name *.resx) | sed -e 's/Co-authored-by: //g'
```

```
$(cat $(find . -name *.resx) | sed -e 's/Co-authored-by: //g'
```

Tip: Use `$(cat $(find . -name *.resx) | sed -e 's/Co-authored-by: //g'` if you only want to remove stale translation keys.

YAML `$(cat $(find . -name *.yaml) | sed -e 's/Co-authored-by: //g'`

```
$(cat $(find . -name *.yaml) | sed -e 's/Co-authored-by: //g'
```

```
$(cat $(find . -name *.yaml) | sed -e 's/Co-authored-by: //g'
```

Customizing list of addons

The list of addons is configured by `WEBLATE_ADDONS`. To add another addon, simply include class absolute name in this setting.

Writing addon

You can write your own addons too, all you need to do is subclass `BaseAddon`, define the addon metadata and implement a callback which will do the processing.

Here is an example addon:

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.
#

from django.utils.translation import gettext_lazy as _

from weblate.addons.base import BaseAddon
from weblate.addons.events import EVENT_PRE_COMMIT

class ExampleAddon(BaseAddon):
    # Filter for compatible components, every key is
    # matched against property of component
    compat = {"file_format": {"po", "po-mono"}}
    # List of events addon should receive
    events = (EVENT_PRE_COMMIT,)
    # Addon unique identifier
    name = "weblate.example.example"
    # Verbose name shown in the user interface
    verbose = _("Example addon")
    # Detailed addon description
    description = _("This addon does nothing it is just an example.")

    # Callback to implement custom behavior
    def pre_commit(self, translation, author):
        return
```

Executing scripts from addon

Addons 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 addon.

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```

# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.
#
"""Example pre commit script."""

from django.utils.translation import gettext_lazy as _

from weblate.addons.events import EVENT_PRE_COMMIT
from weblate.addons.scripts import BaseScriptAddon

class ExamplePreAddon(BaseScriptAddon):
    # Event used to trigger the script
    events = (EVENT_PRE_COMMIT,)
    # Name of the addon, has to be unique
    name = "weblate.example.pre"
    # Verbose name and long 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 [XXXXXXXXXXXXXXXXXXXXXXXXXXXX](#).

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

XXXXXX 2.11 XXXX.

Repository branch configured in the current component.

WL_FILEMASK

Filemask for current component.

WL_TEMPLATE

Filename of template for monolingual translations (can be empty).

WL_NEW_BASE

XXXXXX 2.14 XXXX.

Filename of the file used for creating new translations (can be empty).

WL_FILE_FORMAT

Fileformat used in current component.

WL_LANGUAGE

Language of currently processed translation (not available for component level hooks).

WL_PREVIOUS_HEAD

Previous HEAD on update (available only when running post update hook).

WL_COMPONENT_SLUG

XXXXXX 3.9 XXXX.

Component slug used to construct URL.

WL_PROJECT_SLUG

XXXXXX 3.9 XXXX.

Project slug used to construct URL.

WL_COMPONENT_NAME
3.9

Component name.

WL_PROJECT_NAME
3.9

Project name.

WL_COMPONENT_URL
3.9

Component URL.

WL_ENGAGE_URL
3.9

Project engage URL.

WL:

Component configuration

Post update repository processing

Post update repository processing can be used to update translation files when the VCS upstream source changes. To achieve this, please remember that 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:

```
#!/bin/sh
gulp --gulpfile gulp-i18n-extract.js
git commit -m 'Update source strings' src/languages/en.lang.json
```

Pre commit processing of translations

Use the commit script to automatically make changes to the translation before it is committed to the repository. It is passed as a single parameter consisting of the filename of a current translation.

WL:

2.20

Weblate comes with a built-in translation memory consisting of the following:

Manually imported translation memory (see *User interface*).

Automatically stored translations performed in Weblate (depending on *Translation memory scopes*).

Automatically imported past translations.

Content in the translation memory can be applied one of two ways:

Manually, **WL** view while translating.

Automatically, by translating strings using **WL**, or **WL** addon.

For installation tips, see *Weblate Translation Memory*, which is turned on by default.

Translation memory scopes

3.2: In earlier versions translation memory could be only loaded from a file corresponding to the current imported translation memory scope.

The translation memory scopes are there to allow both privacy and sharing of translations, to suit the desired behavior.

Imported translation memory

Importing arbitrary translation memory data using the `import_memory` command makes memory content available to all users and projects.

Per user translation memory

Stores all user translations automatically in the personal translation memory of each respective user.

Per project translation memory

All translations within a project are automatically stored in a project translation memory only available for this project.

Shared translation memory

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.

Managing translation memory

User interface

3.2.

In the basic user interface you can manage per user and per project translation memories. It can be used to download, wipe or import translation memory.

3.2: Translation memory in JSON can be imported into Weblate, TMX is provided for interoperability with other tools.

3.2:

Weblate 3.2

testuser / Translation memory

Translation memory status	
Number of your entries	0 Download as JSON Download as TMX Delete
Total number of entries	0

Import translation memory

File

Choose File No file chosen

You can upload a TMX or JSON file.

Upload

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):

Exports the memory into JSON

Imports TMX or JSON files into the translation memory

??

?? settings.py ?? Django ??

?: Weblate WSGI Celery
 mod_wsgi Apache

?:

Django Django's documentation

AKISMET_API_KEY

Weblate Akismet API akismet.com

ANONYMOUS_USER_NAME

??

?:

???

AUDITLOG_EXPIRY

3.6

Webate

180

AUTH_LOCK_ATTEMPTS

2.14

:

Password resets. Prevents new e-mails from being sent, avoiding spamming users with too many password reset attempts.

Defaults to 10.

:

AUTO_UPDATE

3.2

3.11: The original on/off option was changed to differentiate which strings are accepted.

Updates all repositories on a daily basis.

: Useful if you are not using `ENABLE_AUTO_UPDATE` to update Weblate repositories automatically.

: On/off options exist in addition to string selection for backward compatibility.

Options are:

No daily updates.

Only update remotes.

Update remotes and merge working copy.

: This requires that *Background tasks using Celery* is working, and will take effect after it is restarted.

AVATAR_URL_PREFIX

Prefix for constructing avatar URLs as: `${AVATAR_URL_PREFIX}/avatar/${MAIL_HASH}?${PARAMS}`. The following services are known to work:

```
AVATAR_URL_PREFIX = 'https://www.gravatar.com/'
```

```
AVATAR_URL_PREFIX = 'https://www.libravatar.org/'
```

:

`ENABLE_AVATARS` Avatars

AUTH_TOKEN_VALID

2.14

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).

AUTH_PASSWORD_DAYS

2.15

How many days using the same password should be allowed.

2.2: Password changes made prior to Weblate 2.15 will not be accounted for in this policy.

180

AUTOFIX_LIST

List of automatic fixes to apply when saving a string.

2.2: Provide a fully-qualified path to the Python class that implementing the autofixer interface.

Available fixes:

Matches whitespace at the start and end of the string to the source.

Replaces trailing dots (...) if the source string has ellipsis (...).

Removes zero-width space characters if the source does not contain any.

Removes control characters if the source does not contain any.

Removes unsafe HTML markup from strings flagged as `safe-html` (see [HTML](#)).

You can select which ones to use:

```
AUTOFIX_LIST = (  
    'weblate.trans.autofixes.whitespace.SameBookendingWhitespace',  
    'weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis',  
)
```

2.2:

`weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis`

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.

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.

**** 2.2: ****

```
# Enable Cloudflare Javascript optimizations  
CSP_SCRIPT_SRC = ["ajax.cloudflare.com"]
```

2.2:

Content security policy, Content Security Policy (CSP)

CHECK_LIST

check Python

check Python

check Python

`weblate.checks.chars.BeginNewlineCheck`,
`weblate.checks.chars.EndNewlineCheck`,
`weblate.checks.chars.MaxLengthCheck`,

```
CHECK_LIST = ()
```

check Python

```
CHECK_LIST = (
    'weblate.checks.chars.BeginNewlineCheck',
    'weblate.checks.chars.EndNewlineCheck',
    'weblate.checks.chars.MaxLengthCheck',
)
```

check Python

check Python

[Running maintenance tasks](#)

COMMENT_CLEANUP_DAYS

3.6

Delete comments after a given number of days. Defaults to `None`, meaning no deletion at all.

COMMIT_PENDING_HOURS

2.10

Number of hours between committing pending changes by way of the background task.

check Python

[Component configuration](#)

[Running maintenance tasks](#)

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 directory used for invoking scripts.

SSH keys and configuration.

Default location for static Django files, specified by `STATIC_ROOT`.

Default location for Django media files, specified by `MEDIA_ROOT`.

Version control repositories.

Daily backup data, please check [Dumped data for backups](#) for details.

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.

??:

BASE_DIR [Backing up and moving Weblate](#)

DATABASE_BACKUP

[3.1](#).

Whether the database backups should be stored as plain text, compressed or skipped. The authorized values are:

"plain"

"compressed"

"none"

??:

Backing up and moving Weblate

DEFAULT_ACCESS_CONTROL

[3.3](#).

The default access control setting for new projects:

Public

Protected

Private

Custom

Use *Custom* if you are managing ACL manually, which means not relying on the internal Weblate management.

??:

Restricted access

DEFAULT_RESTRICTED_COMPONENT

[4.1](#).

The default value for component restriction.

??:

Restricted access

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.

??:

Template markup [Component configuration](#) *Commit, add, delete, merge and addon messages*

DEFAULT_ADDONS

Default addons to install on every created component.

??: This setting affects only newly created components.

?:

```
DEFAULT_ADDONS = {
  # Addon with no parameters
  "weblate.flags.target_edit": {},

  # Addon with parameters
  "weblate.autotranslate.autotranslate": {
    "mode": "suggest",
    "filter_type": "todo",
    "auto_source": "mt",
    "component": "",
    "engines": ["weblate-translation-memory"],
    "threshold": "80",
  }
}
```

??:

install_addon

DEFAULT_COMMITER_EMAIL

?????? 2.4 ????

Committer e-mail address for created translation components defaulting to `noreply@weblate.org`.

??:

`DEFAULT_COMMITER_NAME` *Component configuration* [????????????????]

DEFAULT_COMMITER_NAME

?????? 2.4 ????

Committer name for created translation components defaulting to `Weblate`.

??:

`DEFAULT_COMMITER_EMAIL` *Component configuration* [????????]

DEFAULT_MERGE_STYLE

?????? 3.4 ????

Merge style for any new components.

rebase - default

merge

??:

Component configuration [????????]

DEFAULT_TRANSLATION_PROPAGATION

2.5

Default setting for translation propagation, defaults to True.

:

Component configuration

DEFAULT_PULL_MESSAGE

Title for new pull requests, defaulting to 'Update from Weblate'.

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.

:

AVATAR_URL_PREFIX *Avatars*

ENABLE_HOOKS

Whether to enable anonymous remote hooks.

:

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.

:

SESSION_COOKIE_SECURE *CSRF_COOKIE_SECURE* *SECURE_SSL_REDIRECT* *Set correct site domain*

ENABLE_SHARING

Turn on/off the *Share* menu so users can share translation progress on social networks.

GITLAB_CREDENTIALS

4.3

List for credentials for GitLab servers.

: Use this in case you want Weblate to interact with more of them, for single GitLab endpoint stick with *GITLAB_USERNAME* and *GITLAB_TOKEN*.

```
GITLAB_CREDENTIALS = {
    "gitlab.com": {
        "username": "weblate",
        "token": "your-api-token",
    },
    "gitlab.example.com": {
        "username": "weblate",
        "token": "another-api-token",
    }
}
```

```
} ,
}
```

GITLAB_USERNAME

GitLab username used to send merge requests for translation updates.

??:

`GITLAB_CREDENTIALS` [GitLab](#)

GITLAB_TOKEN

?????? 4.3 ????

????????????? API ?????? GitLab ?????????? ??????

??:

`GITLAB_CREDENTIALS` [GitLab](#) [GitLab: Personal access token](#)

GITHUB_CREDENTIALS

?????? 4.3 ????

List for credentials for GitHub servers.

???: Use this in case you want Weblate to interact with more of them, for single GitHub endpoint stick with `GITHUB_USERNAME` and `GITHUB_TOKEN`.

```
GITHUB_CREDENTIALS = {
  "api.github.com": {
    "username": "weblate",
    "token": "your-api-token",
  },
  "github.example.com": {
    "username": "weblate",
    "token": "another-api-token",
  },
}
```

GITHUB_USERNAME

GitHub username used to send pull requests for translation updates.

??:

`GITHUB_CREDENTIALS` [GitHub](#)

GITHUB_TOKEN

?????? 4.3 ????

GitHub personal access token used to make API calls to send pull requests for translation updates.

??:

`GITHUB_CREDENTIALS` [GitHub](#) [Creating a personal access token](#)

GOOGLE_ANALYTICS_ID

Google Analytics ID to turn on monitoring of Weblate using Google Analytics.

HIDE_REPO_CREDENTIALS

Hide repository credentials from appearing in 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.

??: This is turned on by default.

HIDE_VERSION

????? 4.3.1 ???.

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.

??: This is turned off by default.

IP_BEHIND_REVERSE_PROXY

????? 2.14 ???.

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`.

??: Ensure you are actually using a reverse proxy and that it sets this header, otherwise users will be able to fake the IP address.

??: This is not on by default.

??:

Running behind reverse proxy ???? ? IP_PROXY_HEADER ? IP_PROXY_OFFSET

IP_PROXY_HEADER

????? 2.14 ???.

Indicates which header Weblate should obtain the IP address from when `IP_BEHIND_REVERSE_PROXY` is turned on.

Defaults to `HTTP_X_FORWARDED_FOR`.

??:

Running behind reverse proxy ???? ? SECURE_PROXY_SSL_HEADER ? IP_BEHIND_REVERSE_PROXY ? IP_PROXY_OFFSET

IP_PROXY_OFFSET

2.14

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.

Warning: Setting this affects the security of your installation, you should only configure it to use trusted proxies for determining IP address.

Defaults to 0.

Warning:

Running behind reverse proxy `SECURE_PROXY_SSL_HEADER, IP_BEHIND_REVERSE_PROXY, IP_PROXY_HEADER`

LEGAL_URL

3.5

URL where your Weblate instance shows its legal documents.

Warning: Useful if you host your legal documents outside Weblate for embedding them inside Weblate, please check [for details](#).

Warning:

```
LEGAL_URL = "https://weblate.org/terms/"
```

LICENSE_EXTRA

Additional licenses to include in the license choices.

Warning: Each license definition should be tuple of its short name, a long name and an URL.

For example:

```
LICENSE_EXTRA = [
    (
        "AGPL-3.0",
        "GNU Affero General Public License v3.0",
        "https://www.gnu.org/licenses/agpl-3.0-standalone.html",
    ),
]
```

LICENSE_FILTER

4.3 **Warning:** 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.

Warning: This filter uses the short license names.

For example:

```
LICENSE_FILTER = {"AGPL-3.0", "GPL-3.0-or-later"}
```

Following disables the license alert:

```
LICENSE_FILTER = set()
```

??:

Translation component alerts

LICENSE_REQUIRED

Defines whether the license attribute in *Component configuration* is required.

?: This is off by default.

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.

?: Set this to `False` to allow longer translations (up to 10.000 characters) irrespective of source string length.

?: Defaults to `True`.

LOCALIZE_CDN_URL and LOCALIZE_CDN_PATH

These settings configure the *JavaScript* [CDN](#) 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`.

?: On Hosted Weblate, this uses `https://weblate-cdn.com/`.

?:

JavaScript [CDN](#)

LOGIN_REQUIRED_URLS

A list of URLs you want to require logging into. (Besides the standard rules built into Weblate).

?: This allows you to password protect a whole installation using:

```
LOGIN_REQUIRED_URLS = (
    r'/(.*)$',
)
REST_FRAMEWORK["DEFAULT_PERMISSION_CLASSES"] = [
    "rest_framework.permissions.IsAuthenticated"
]
```

?: It is desirable to lock down API access as well, as shown in the above example.

LOGIN_REQUIRED_URLS_EXCEPTIONS

List of exceptions for `LOGIN_REQUIRED_URLS`. If not specified, users are allowed to access the login page.
Some of exceptions you might want to include:

```
LOGIN_REQUIRED_URLS_EXCEPTIONS = (  
    r'/accounts/(.*)$', # Required for login  
    r'/static/(.*)$', # Required for development mode  
    r'/widgets/(.*)$', # Allowing public access to widgets  
    r'/data/(.*)$', # Allowing public access to data exports  
    r'/hooks/(.*)$', # Allowing public access to notification hooks  
    r'/api/(.*)$', # Allowing access to API  
    r'/js/i18n/$', # JavaScript localization  
)
```

MATOMO_SITE_ID

ID of a site in Matomo (formerly Piwik) you want to track.

🔗: This integration does not support the Matomo Tag Manager.

🔗:

`MATOMO_URL`

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.

🔗🔗: This integration does not support the Matomo Tag Manager.

For example:

```
MATOMO_SITE_ID = 1  
MATOMO_URL = "https://example.matomo.cloud/"
```

🔗:

`MATOMO_SITE_ID`

MT_SERVICES

🔗🔗🔗🔗 3.0 🔗🔗: 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.

🔗: Many of the services need additional configuration like API keys, please check their documentation 🔗🔗🔗 for more details.

```
MT_SERVICES = (  
    'weblate.machinery.apertium.ApertiumAPYTranslation',  
    'weblate.machinery.deepl.DeepLTranslation',  
    'weblate.machinery.glosbe.GlosbeTranslation',  
    'weblate.machinery.google.GoogleTranslation',  
    'weblate.machinery.microsoft.MicrosoftCognitiveTranslation',  
    'weblate.machinery.microsoftterminology.MicrosoftTerminologyService',  
    'weblate.machinery.mymemory.MyMemoryTranslation',  
    'weblate.machinery.tmserver.AmagamaTranslation',  
    'weblate.machinery.tmserver.TMServerTranslation',  
    'weblate.machinery.yandex.YandexTranslation',  
)
```

(🔗🔗🔗🔗🔗🔗)

(XXXXXXXXXXXX)

```
'weblate.machinery.weblatetm.WeblateTranslation',  
'weblate.machinery.saptranslationhub.SAPTranslationHub',  
'weblate.memory.machine.WeblateMemory',  
)
```

??:

XXXXXXXXXX

MT_APERTIUM_APY

URL of the Apertium-APy server, <https://wiki.apertium.org/wiki/Apertium-apy>

??:

ApertiumXXXXXXXXXX

MT_AWS_ACCESS_KEY_ID

Access key ID for Amazon Translate.

??:

AWSXXXXXXXXXX

MT_AWS_SECRET_ACCESS_KEY

API secret key for Amazon Translate.

??:

AWSXXXXXXXXXX

MT_AWS_REGION

Region name to use for Amazon Translate.

??:

AWSXXXXXXXXXX

MT_Baidu_ID

Client ID for the Baidu Zhiyun API, you can register at <https://api.fanyi.baidu.com/api/trans/product/index>

??:

Baidu API machine translationXXXXXXXXXX

MT_Baidu_SECRET

Client secret for the Baidu Zhiyun API, you can register at <https://api.fanyi.baidu.com/api/trans/product/index>

??:

Baidu API machine translationXXXXXXXXXX

MT_DEEPL_API_VERSION

4.1.1

API version to use with DeepL service. The version limits scope of usage:

Is meant for CAT tools and is usable with user-based subscription.

Is meant for API usage and the subscription is usage based.

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.

Example:

`DeepL`

MT_DEEPL_KEY

API key for the DeepL API, you can register at <https://www.deepl.com/pro.html>

Example:

`DeepL`

MT_GOOGLE_KEY

API key for Google Translate API v2, you can register at <https://cloud.google.com/translate/docs>

Example:

`Google Translate`

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.

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

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

Example:

`Google Translate API V3 (Advanced)`

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`.

MT_MICROSOFT_COGNITIVE_KEY

Client key for the Microsoft Cognitive Services Translator API.

??:

Microsoft Cognitive Services Translator [Microsoft Cognitive Services - Text Translation API](#) [Microsoft Azure Portal](#)

MT_MICROSOFT_REGION

Region prefix as defined in the "Authenticating with a Multi-service resource" section.

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.

MT_MODERNMT_KEY

ModernMT [API](#)

??:

ModernMT `MT_MODERNMT_URL`

MT_MODERNMT_URL

URL of ModernMT. It defaults to `https://api.modernmt.com/` for the cloud service.

??:

ModernMT `MT_MODERNMT_KEY`

MT_MYMEMORY_EMAIL

MyMemory identification e-mail address. It permits 1000 requests per day.

??:

MyMemory [MyMemory: API technical specifications](#)

MT_MYMEMORY_KEY

MyMemory access key for private translation memory, use it with `MT_MYMEMORY_USER`.

??:

MyMemory [MyMemory: API key generator](#)

MT_MYMEMORY_USER

MyMemory user ID for private translation memory, use it with *MT_MYMEMORY_KEY*.

🔗:

MyMemory 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 *MyMemory: API key generator*

MT_NETEASE_KEY

App key for NetEase Sight API, you can register at <https://sight.netease.com/>

🔗:

NetEase Sight API machine translation 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗

MT_NETEASE_SECRET

App secret for the NetEase Sight API, you can register at <https://sight.netease.com/>

🔗:

NetEase Sight API machine translation 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗

MT_TMSERVER

URL where tmserver is running.

🔗:

tmserver 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 *tmserver*

MT_YANDEX_KEY

API key for the Yandex Translate API, you can register at <https://yandex.com/dev/translate/>

🔗:

Yandex Translate 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗

MT_YOUDAO_ID

Client ID for the Youdao Zhiyun API, you can register at <https://ai.youdao.com/product-fanyi-text.s>.

🔗:

Youdao Zhiyun API machine translation 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗

MT_YOUDAO_SECRET

Client secret for the Youdao Zhiyun API, you can register at <https://ai.youdao.com/product-fanyi-text.s>.

🔗:

Youdao Zhiyun API machine translation 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗 🔗

MT_SAP_BASE_URL

API URL to the SAP Translation Hub service.

??:

SAP Translation Hub [?][?][?][?][?][?][?][?]

MT_SAP_SANDBOX_APIKEY

API key for sandbox API usage

??:

SAP Translation Hub [?][?][?][?][?][?][?][?]

MT_SAP_USERNAME

Your SAP username

??:

SAP Translation Hub [?][?][?][?][?][?][?][?]

MT_SAP_PASSWORD

Your SAP password

??:

SAP Translation Hub [?][?][?][?][?][?][?][?]

MT_SAP_USE_MT

Whether to also use machine translation services, in addition to the term database. Possible values: True or False

??:

SAP Translation Hub [?][?][?][?][?][?][?][?]

NEARBY_MESSAGES

How many strings to show around the currently translated string. This is just a default value, users can adjust this in [?][?][?][?][?].

RATELIMIT_ATTEMPTS

[?][?][?][?] 3.2 [?][?].

Maximum number of authentication attempts before rate limiting is applied.

Defaults to 5.

??:

[?][?][?][?][?]RATELIMIT_WINDOW [?][?][?][?][?]RATELIMIT_LOCKOUT

RATELIMIT_WINDOW

3.2

How long authentication is accepted after rate limiting applies.

An amount of seconds defaulting to 300 (5 minutes).

:

`RATELIMIT_ATTEMPTS` `RATELIMIT_LOCKOUT`

RATELIMIT_LOCKOUT

3.2

How long authentication is locked after rate limiting applies.

An amount of seconds defaulting to 600 (10 minutes).

:

`RATELIMIT_ATTEMPTS` `RATELIMIT_WINDOW`

REGISTRATION_ALLOW_BACKENDS

4.1

List of authentication backends to allow registration from. This only limits new registrations, users can still authenticate and add authentication using all configured authentication backends.

It is recommended to keep `REGISTRATION_OPEN` enabled while limiting registration backends, otherwise users will be able to register, but Weblate will not show links to register in the user interface.

:

```
REGISTRATION_ALLOW_BACKENDS = ["azuread-oauth2", "azuread-tenant-oauth2"]
```

: The backend names match names used in URL for authentication.

:

`REGISTRATION_OPEN`

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.

REGISTRATION_EMAIL_MATCH

2.17

Allows you to filter which e-mail addresses can register.

Defaults to `*`, which allows any e-mail address to be registered.

You can use it to restrict registration to a single e-mail domain:

```
REGISTRATION_EMAIL_MATCH = r'^.*@weblate\.org$'
```

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`).

Note: 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.

Note:

`REGISTRATION_ALLOW_BACKENDS` `REGISTRATION_EMAIL_MATCH`

REPOSITORY_ALERT_THRESHOLD

4.0.2

Threshold for triggering an alert for outdated repositories, or ones that contain too many changes. Defaults to 25.

Note:

Translation component alerts

REQUIRE_LOGIN

4.1

This enables `LOGIN_REQUIRED_URLS` and configures REST framework to require login for all API endpoints.

Note: This is implemented in the `weblate`. For Docker, use `WEBLATE_REQUIRE_LOGIN`.

SENTRY_DSN

3.9

Sentry DSN to use for *Collecting error reports*.

Note:

Django integration for Sentry

SESSION_COOKIE_AGE_AUTHENTICATED

4.3

Set session expiry for authenticated users. This complements `SESSION_COOKIE_AGE` which is used for unauthenticated users.

Example:

```
SESSION_COOKIE_AGE
```

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.

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"
```

Example: 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`.

Example: On a Docker container, the site domain is configured through `WEBLATE_ALLOWED_HOSTS`.

Example:

```
Set correct site domain Allowed hosts setup Correctly configure HTTPS WE-
BLATE_SITE_DOMAIN ENABLE_HTTPS
```

SITE_TITLE

Site title to be used for the website and sent e-mails.

SPECIAL_CHARS

Additional characters to include in the visual keyboard, .

The default value is:

```
SPECIAL_CHARS = ('\t', '\n', '...')
```

SINGLE_PROJECT

3.8

Redirects users directly to a project or component instead of showing the dashboard. You can either set it to `True` and in this case it only works in case there is actually only single project in Weblate. Alternatively set the project slug, and it will redirect unconditionally to this project.

3.11: The setting now also accepts a project slug, to force displaying that single project.

:

```
SINGLE_PROJECT = "test"
```

STATUS_URL

The URL where your Weblate instance reports its status.

SUGGESTION_CLEANUP_DAYS

3.2.1

Automatically deletes suggestions after a given number of days. Defaults to `None`, meaning no deletions.

URL_PREFIX

This setting allows you to run Weblate under some path (otherwise it relies on being run from the webserver root).

: 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`.

: The prefix should start with a `/`.

:

```
URL_PREFIX = '/translations'
```

: This setting does not work with Django's built-in server, you would have to adjust `urls.py` to contain this prefix.

VCS_BACKENDS

Configuration of available VCS backends.

: Weblate tries to use all supported back-ends you have the tools for.

: You can limit choices or add custom VCS back-ends by using this.

```
VCS_BACKENDS = (  
    'weblate.vcs.git.GitRepository',  
)
```

:

VCS_CLONE_DEPTH

3.10.2

Configures how deep cloning of repositories Weblate should do.

Note: 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 `weblate`), you might want to increase the depth or turn off shallow clones completely by setting this to 0.

Note: 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
```

WEBLATE_ADDONS

List of addons available for use. To use them, they have to be enabled for a given translation component. By default this includes all built-in addons, when extending the list you will probably want to keep existing ones enabled, for example:

```
WEBLATE_ADDONS = (  
    # Built-in addons  
    "weblate.addons.gettext.GenerateMoAddon",  
    "weblate.addons.gettext.UpdateLinguasAddon",  
    "weblate.addons.gettext.UpdateConfigureAddon",  
    "weblate.addons.gettext.MsgmergeAddon",  
    "weblate.addons.gettext.GettextCustomizeAddon",  
    "weblate.addons.gettext.GettextAuthorComments",  
    "weblate.addons.cleanup.CleanupAddon",  
    "weblate.addons.consistency.LanguaugeConsistencyAddon",  
    "weblate.addons.discovery.DiscoveryAddon",  
    "weblate.addons.flags.SourceEditAddon",  
    "weblate.addons.flags.TargetEditAddon",  
    "weblate.addons.flags.SameEditAddon",  
    "weblate.addons.flags.BulkEditAddon",  
    "weblate.addons.generate.GenerateFileAddon",  
    "weblate.addons.json.JSONCustomizeAddon",  
    "weblate.addons.properties.PropertiesSortAddon",  
    "weblate.addons.git.GitSquashAddon",  
    "weblate.addons.removal.RemoveComments",  
    "weblate.addons.removal.RemoveSuggestions",  
    "weblate.addons.resx.ResxUpdateAddon",  
    "weblate.addons.autotranslate.AutoTranslateAddon",  
    "weblate.addons.yaml.YAMLCustomizeAddon",  
    "weblate.addons.cdn.CDNJSAddon",  
  
    # Addon you want to include  
    "weblate.addons.example.ExampleAddon",  
)
```

Note:

`weblate`

WEBLATE_EXPORTERS

4.2

List of a available exporters offering downloading translations or glossaries in various file formats.

:

Supported file formats

WEBLATE_FORMATS

3.0

List of file formats available for use.

: The default list already has the common formats.

:

Supported file formats

WEBLATE_GPG_IDENTITY

3.1

Identity used by Weblate to sign Git commits, for example:

```
WEBLATE_GPG_IDENTITY = 'Weblate <weblate@example.com>'
```

The Weblate GPG keyring is searched for a matching key (home/ .gnupg under `DATA_DIR`). If not found, a key is generated, please check *Signing Git commits with GnuPG* for more details.

:

Signing Git commits with GnuPG

weblate/settings_example.py

Weblate `weblate/settings_example.py`:

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.
#

import os
import platform
from logging.handlers import SysLogHandler

#
# Django settings for Weblate project.
```

(2020)

```

#
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
            # "connect_timeout": 28800,
        },
    },
}

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Data directory
DATA_DIR = os.path.join(BASE_DIR, "data")

# Local time zone for this installation. Choices can be found here:
# http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
# although not all choices may be available on all operating systems.
# In a Windows environment this must be set to your system time zone.
TIME_ZONE = "UTC"

# Language code for this installation. All choices can be found here:
# http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = "en-us"

LANGUAGES = (
    ("ar", "العربية"),
    ("az", "Azərbaycan"),
    ("be", "Беларуская"),
    ("be@latin", "Biełaruskaja"),
    ("bg", "Български"),
    ("br", "Brezhoneg"),
    ("ca", "Català"),

```

```

("cs", "Čeština"),
("da", "Dansk"),
("de", "Deutsch"),
("en", "English"),
("el", "Ελληνικά"),
("en-gb", "English (United Kingdom)"),
("es", "Español"),
("fi", "Suomi"),
("fr", "Français"),
("gl", "Galego"),
("he", "עברית"),
("hu", "Magyar"),
("hr", "Hrvatski"),
("id", "Indonesia"),
("is", "Íslenska"),
("it", "Italiano"),
("ja", "????"),
("kab", "Taqbaylit"),
("kk", "Қазақ тілі"),
("ko", "????"),
("nb", "Norsk bokmål"),
("nl", "Nederlands"),
("pl", "Polski"),
("pt", "Português"),
("pt-br", "Português brasileiro"),
("ru", "Русский"),
("sk", "Slovenčina"),
("sl", "Slovenščina"),
("sq", "Shqip"),
("sr", "Српски"),
("sv", "Svenska"),
("tr", "Türkçe"),
("uk", "Українська"),
("zh-hans", "????"),
("zh-hant", "????"),
)

SITE_ID = 1

# If you set this to False, Django will make some optimizations so as not
# to load the internationalization machinery.
USE_I18N = True

# If you set this to False, Django will not format dates, numbers and
# calendars according to the current locale.
USE_L10N = True

# If you set this to False, Django will not use timezone-aware datetimes.
USE_TZ = True

# 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.i18n",
                "django.template.context_processors.request",
                "django.template.context_processors.csrf",
                "django.contrib.messages.context_processors.messages",
                "weblate.trans.context_processors.weblate_context",
            ],
            "loaders": _TEMPLATE_LOADERS,
        },
    },
]

# GitHub username for sending pull requests.
# Please see the documentation for more details.
GITHUB_USERNAME = None
GITHUB_TOKEN = None

# GitLab username for sending merge requests.
# Please see the documentation for more details.
GITLAB_USERNAME = None
GITLAB_TOKEN = None

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.email.EmailAuth",
    # "social_core.backends.google.GoogleOAuth2",
    # "social_core.backends.github.GithubOAuth2",
    # "social_core.backends.bitbucket.BitbucketOAuth",
    # "social_core.backends.suse.OpenSUSEOpenId",
    # "social_core.backends.ubuntu.UbuntuOpenId",
    # "social_core.backends.fedora.FedoraOpenId",
    # "social_core.backends.facebook.FacebookOAuth2",

```

```

        "weblate.accounts.auth.WeblateUserBackend",
    )

    # Custom user model
    AUTH_USER_MODEL = "weblate_auth.User"

    # Social auth backends setup
    SOCIAL_AUTH_GITHUB_KEY = ""
    SOCIAL_AUTH_GITHUB_SECRET = ""
    SOCIAL_AUTH_GITHUB_SCOPE = ["user:email"]

    SOCIAL_AUTH_BITBUCKET_KEY = ""
    SOCIAL_AUTH_BITBUCKET_SECRET = ""
    SOCIAL_AUTH_BITBUCKET_VERIFIED_EMAILS_ONLY = True

    SOCIAL_AUTH_FACEBOOK_KEY = ""
    SOCIAL_AUTH_FACEBOOK_SECRET = ""
    SOCIAL_AUTH_FACEBOOK_SCOPE = ["email", "public_profile"]
    SOCIAL_AUTH_FACEBOOK_PROFILE_EXTRA_PARAMS = {"fields": "id,name,email"}

    SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = ""
    SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = ""

    # Social auth settings
    SOCIAL_AUTH_PIPELINE = (
        "social_core.pipeline.social_auth.social_details",
        "social_core.pipeline.social_auth.social_uid",
        "social_core.pipeline.social_auth.auth_allowed",
        "social_core.pipeline.social_auth.social_user",
        "weblate.accounts.pipeline.store_params",
        "weblate.accounts.pipeline.verify_open",
        "social_core.pipeline.user.get_username",
        "weblate.accounts.pipeline.require_email",
        "social_core.pipeline.mail.mail_validation",
        "weblate.accounts.pipeline.revoke_mail_code",
        "weblate.accounts.pipeline.ensure_valid",
        "weblate.accounts.pipeline.remove_account",
        "social_core.pipeline.social_auth.associate_by_email",
        "weblate.accounts.pipeline.reauthenticate",
        "weblate.accounts.pipeline.verify_username",
        "social_core.pipeline.user.create_user",
        "social_core.pipeline.social_auth.associate_user",
        "social_core.pipeline.social_auth.load_extra_data",
        "weblate.accounts.pipeline.cleanup_next",
        "weblate.accounts.pipeline.user_full_name",
        "weblate.accounts.pipeline.store_email",
        "weblate.accounts.pipeline.notify_connect",
        "weblate.accounts.pipeline.password_reset",
    )
    SOCIAL_AUTH_DISCONNECT_PIPELINE = (
        "social_core.pipeline.disconnect.allowed_to_disconnect",
        "social_core.pipeline.disconnect.get_entries",
        "social_core.pipeline.disconnect.revoke_tokens",
        "weblate.accounts.pipeline.cycle_session",
        "weblate.accounts.pipeline.adjust_primary_mail",
        "weblate.accounts.pipeline.notify_disconnect",
        "social_core.pipeline.disconnect.disconnect",
        "weblate.accounts.pipeline.cleanup_next",
    )

    # Custom authentication strategy
    SOCIAL_AUTH_STRATEGY = "weblate.accounts.strategy.WeblateStrategy"

    # Raise exceptions so that we can handle them later
    SOCIAL_AUTH_RAISE_EXCEPTIONS = True

    SOCIAL_AUTH_EMAIL_VALIDATION_FUNCTION = "weblate.accounts.pipeline.send_
    <-validation"

```

```

SOCIAL_AUTH_EMAIL_VALIDATION_URL = "{0}/accounts/email-sent/".format(URL_
↪PREFIX)
SOCIAL_AUTH_LOGIN_ERROR_URL = "{0}/accounts/login/".format(URL_PREFIX)
SOCIAL_AUTH_EMAIL_FORM_URL = "{0}/accounts/email/".format(URL_PREFIX)
SOCIAL_AUTH_NEW_ASSOCIATION_REDIRECT_URL = "{0}/accounts/profile/#account".
↪format(
    URL_PREFIX
)
SOCIAL_AUTH_PROTECTED_USER_FIELDS = ("email",)
SOCIAL_AUTH_SLUGIFY_USERNAMES = True
SOCIAL_AUTH_SLUGIFY_FUNCTION = "weblate.accounts.pipeline.slugify_username"

# Password validation configuration
AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME": "django.contrib.auth.password_validation.
↪UserAttributeSimilarityValidator" # noqa: E501, pylint: disable=line-
↪too-long
    },
    {
        "NAME": "django.contrib.auth.password_validation.
↪MinimumLengthValidator",
        "OPTIONS": {"min_length": 10},
    },
    {"NAME": "django.contrib.auth.password_validation.
↪CommonPasswordValidator"},
    {"NAME": "django.contrib.auth.password_validation.
↪NumericPasswordValidator"},
    {"NAME": "weblate.accounts.password_validation.CharsPasswordValidator"}
↪,
    {"NAME": "weblate.accounts.password_validation.PastPasswordsValidator"}
↪,
    # Optional password strength validation by django-zxcvbn-password
    # {
    #     "NAME": "zxcvbn_password.ZXCVCBNValidator",
    #     "OPTIONS": {
    #         "min_score": 3,
    #         "user_attributes": ("username", "email", "full_name")
    #     }
    # },
]

# 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.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "weblate.accounts.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
    "social_django.middleware.SocialAuthExceptionMiddleware",
    "weblate.accounts.middleware.RequireLoginMiddleware",
    "weblate.api.middleware.ThrottlingMiddleware",
    "weblate.middleware.SecurityMiddleware",
]

ROOT_URLCONF = "weblate.urls"

```

```

# Django and Weblate apps
INSTALLED_APPS = [
    # Weblate apps on top to override Django locales and templates
    "weblate.addons",
    "weblate.auth",
    "weblate.checks",
    "weblate.formats",
    "weblate.glossary",
    "weblate.machinery",
    "weblate.trans",
    "weblate.lang",
    "weblate_language_data",
    "weblate.memory",
    "weblate.screenshots",
    "weblate.fonts",
    "weblate.accounts",
    "weblate.utils",
    "weblate.vcs",
    "weblate.wladmin",
    "weblate",
    # Optional: Git exporter
    "weblate.gitexport",
    # Standard Django modules
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "django.contrib.admin.apps.SimpleAdminConfig",
    "django.contrib.admindocs",
    "django.contrib.sitemaps",
    "django.contrib.humanize",
    # Third party Django modules
    "social_django",
    "crispy_forms",
    "compressor",
    "rest_framework",
    "rest_framework.authtoken",
    "django_filters",
]

# Custom exception reporter to include some details
DEFAULT_EXCEPTION_REPORTER_FILTER = "weblate.trans.debug.
↪WeblateExceptionReporterFilter"

# Default logging of Weblate messages
# - to syslog in production (if available)
# - otherwise to console
# - you can also choose "logfile" to log into separate file
#   after configuring it below

# Detect if we can connect to syslog
HAVE_SYSLOG = False
if platform.system() != "Windows":
    try:
        handler = SysLogHandler(address="/dev/log", facility=SysLogHandler.
↪LOG_LOCAL2)
        handler.close()
        HAVE_SYSLOG = True
    except IOError:
        HAVE_SYSLOG = False

if DEBUG or not HAVE_SYSLOG:
    DEFAULT_LOG = "console"
else:
    DEFAULT_LOG = "syslog"
DEFAULT_LOGLEVEL = "DEBUG" if DEBUG else "INFO"

```

```

# A sample logging configuration. The only tangible logging
# performed by this configuration is to send an email to
# the site admins on every HTTP 500 error when DEBUG=False.
# See http://docs.djangoproject.com/en/stable/topics/logging for
# more details on how to customize your logging configuration.
LOGGING = {
    "version": 1,
    "disable_existing_loggers": True,
    "filters": {"require_debug_false": {"()": "django.utils.log.
↪RequireDebugFalse"}},
    "formatters": {
        "syslog": {"format": "weblate[%(process)d]: %(levelname)s
↪%(message)s"},
        "simple": {"format": "%(levelname)s %(message)s"},
        "logfile": {"format": "%(asctime)s %(levelname)s %(message)s"},
        "django.server": {
            "()": "django.utils.log.ServerFormatter",
            "format": "[%(server_time)s] %(message)s",
        },
    },
    "handlers": {
        "mail_admins": {
            "level": "ERROR",
            "filters": ["require_debug_false"],
            "class": "django.utils.log.AdminEmailHandler",
            "include_html": True,
        },
        "console": {
            "level": "DEBUG",
            "class": "logging.StreamHandler",
            "formatter": "simple",
        },
        "django.server": {
            "level": "INFO",
            "class": "logging.StreamHandler",
            "formatter": "django.server",
        },
        "syslog": {
            "level": "DEBUG",
            "class": "logging.handlers.SysLogHandler",
            "formatter": "syslog",
            "address": "/dev/log",
            "facility": SysLogHandler.LOG_LOCAL2,
        },
        # Logging to a file
        # "logfile": {
        #     "level": "DEBUG",
        #     "class": "logging.handlers.RotatingFileHandler",
        #     "filename": "/var/log/weblate/weblate.log",
        #     "maxBytes": 100000,
        #     "backupCount": 3,
        #     "formatter": "logfile",
        # },
    },
    "loggers": {
        "django.request": {
            "handlers": ["mail_admins", DEFAULT_LOG],
            "level": "ERROR",
            "propagate": True,
        },
        "django.server": {
            "handlers": ["django.server"],
            "level": "INFO",
            "propagate": False,
        },
        # Logging database queries

```

```

# "django.db.backends": {
#     "handlers": [DEFAULT_LOG],
#     "level": "DEBUG",
# },
"weblate": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
# Logging VCS operations
"weblate.vcs": {"handlers": [DEFAULT_LOG], "level": DEFAULT_
↪LOGLEVEL},
# Python Social Auth
"social": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
# Django Authentication Using LDAP
"django_auth_ldap": {"handlers": [DEFAULT_LOG], "level": DEFAULT_
↪LOGLEVEL},
    },
}

# Remove syslog setup if it's not present
if not HAVE_SYSLOG:
    del LOGGING["handlers"]["syslog"]

# List of machine translations
MT_SERVICES = (
#     "weblate.machinery.apertium.ApertiumAPYTranslation",
#     "weblate.machinery.baidu.BaiduTranslation",
#     "weblate.machinery.deepl.DeepLTranslation",
#     "weblate.machinery.glosbe.GlosbeTranslation",
#     "weblate.machinery.google.GoogleTranslation",
#     "weblate.machinery.googlev3.GoogleV3Translation",
#     "weblate.machinery.microsoft.MicrosoftCognitiveTranslation",
#     "weblate.machinery.microsoftterminology.
↪MicrosoftTerminologyService",
#     "weblate.machinery.modernmt.ModernMTTranslation",
#     "weblate.machinery.mymemory.MyMemoryTranslation",
#     "weblate.machinery.netease.NeteaseSightTranslation",
#     "weblate.machinery.tmsserver.AmagamaTranslation",
#     "weblate.machinery.tmsserver.TMServerTranslation",
#     "weblate.machinery.yandex.YandexTranslation",
#     "weblate.machinery.saptranslationhub.SAPTranslationHub",
#     "weblate.machinery.youdao.YoudaoTranslation",
"weblate.machinery.weblatetm.WeblateTranslation",
"weblate.memory.machine.WeblateMemory",
)

# Machine translation API keys

# URL of the Apertium APy server
MT_APERTIUM_APY = None

# DeepL API key
MT_DEEPL_KEY = None

# Microsoft Cognitive Services Translator API, register at
# https://portal.azure.com/
MT_MICROSOFT_COGNITIVE_KEY = None
MT_MICROSOFT_REGION = None

# ModernMT
MT_MODERNMT_KEY = None

# MyMemory identification email, see
# https://mymemory.translated.net/doc/spec.php
MT_MYMEMORY_EMAIL = None

# Optional MyMemory credentials to access private translation memory
MT_MYMEMORY_USER = None
MT_MYMEMORY_KEY = None

```

```

# Google API key for Google Translate API v2
MT_GOOGLE_KEY = None

# Google Translate API3 credentials and project id
MT_GOOGLE_CREDENTIALS = None
MT_GOOGLE_PROJECT = None

# Baidu app key and secret
MT_BAIDU_ID = None
MT_BAIDU_SECRET = None

# Youdao Zhiyun app key and secret
MT_YOUDAO_ID = None
MT_YOUDAO_SECRET = None

# Netease Sight (Jianwai) app key and secret
MT_NETEASE_KEY = None
MT_NETEASE_SECRET = None

# API key for Yandex Translate API
MT_YANDEX_KEY = None

# tmserver URL
MT_TMSERVER = None

# SAP Translation Hub
MT_SAP_BASE_URL = None
MT_SAP_SANDBOX_APIKEY = None
MT_SAP_USERNAME = None
MT_SAP_PASSWORD = None
MT_SAP_USE_MT = True

# Title of site to use
SITE_TITLE = "Weblate"

# Site domain
SITE_DOMAIN = ""

# Whether site uses https
ENABLE_HTTPS = False

# Use HTTPS when creating redirect URLs for social authentication, see
# documentation for more details:
# https://python-social-auth-docs.readthedocs.io/en/latest/configuration/
# ↪ settings.html#processing-redirects-and-urlopen
SOCIAL_AUTH_REDIRECT_IS_HTTPS = ENABLE_HTTPS

# Make CSRF cookie HttpOnly, see documentation for more details:
# https://docs.djangoproject.com/en/1.11/ref/settings/#csrf-cookie-httponly
CSRF_COOKIE_HTTPONLY = True
CSRF_COOKIE_SECURE = ENABLE_HTTPS
# Store CSRF token in session
CSRF_USE_SESSIONS = True
# Customize CSRF failure view
CSRF_FAILURE_VIEW = "weblate.trans.views.error.csrf_failure"
SESSION_COOKIE_SECURE = ENABLE_HTTPS
SESSION_COOKIE_HTTPONLY = True
# SSL redirect
SECURE_SSL_REDIRECT = ENABLE_HTTPS
# Sent referrrrer 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

```

```

# Increase allowed upload size
DATA_UPLOAD_MAX_MEMORY_SIZE = 5000000

# 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

# 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 = "{0}/accounts/login/".format(URL_PREFIX)

# URL of logout
LOGOUT_URL = "{0}/accounts/logout/".format(URL_PREFIX)

# Default location for login
LOGIN_REDIRECT_URL = "{0}/".format(URL_PREFIX)

# Anonymous user name
ANONYMOUS_USER_NAME = "anonymous"

# Reverse proxy settings
IP_PROXY_HEADER = "HTTP_X_FORWARDED_FOR"
IP_BEHIND_REVERSE_PROXY = False
IP_PROXY_OFFSET = 0

# Sending HTML in mails
EMAIL_SEND_HTML = True

# Subject of emails includes site title
EMAIL_SUBJECT_PREFIX = "[{0}] ".format(SITE_TITLE)

# Enable remote hooks
ENABLE_HOOKS = True

# By default the length of a given translation is limited to the length of
# the source string * 10 characters. Set this option to False to allow
↪ longer
# translations (up to 10.000 characters)
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH = True

# Use simple language codes for default language/country combinations
SIMPLIFY_LANGUAGES = True

# Render forms using bootstrap
CRISPY_TEMPLATE_PACK = "bootstrap3"

# List of quality checks
# CHECK_LIST = (
#     "weblate.checks.same.SameCheck",
#     "weblate.checks.chars.BeginNewlineCheck",
#     "weblate.checks.chars.EndNewlineCheck",
#     "weblate.checks.chars.BeginSpaceCheck",
#     "weblate.checks.chars.EndSpaceCheck",
#     "weblate.checks.chars.DoubleSpaceCheck",

```

```

# "weblate.checks.chars.EndStopCheck",
# "weblate.checks.chars.EndColonCheck",
# "weblate.checks.chars.EndQuestionCheck",
# "weblate.checks.chars.EndExclamationCheck",
# "weblate.checks.chars.EndEllipsisCheck",
# "weblate.checks.chars.EndSemicolonCheck",
# "weblate.checks.chars.MaxLengthCheck",
# "weblate.checks.chars.KashidaCheck",
# "weblate.checks.chars.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.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.chars.ZeroWidthSpaceCheck",
# "weblate.checks.render.MaxSizeCheck",
# "weblate.checks.markup.XMLValidityCheck",
# "weblate.checks.markup.XMLTagsCheck",
# "weblate.checks.markup.MarkdownRefLinkCheck",
# "weblate.checks.markup.MarkdownLinkCheck",
# "weblate.checks.markup.MarkdownSyntaxCheck",
# "weblate.checks.markup.URLCheck",
# "weblate.checks.markup.SafeHTMLCheck",
# "weblate.checks.placeholders.PlaceholderCheck",
# "weblate.checks.placeholders.RegexCheck",
# "weblate.checks.duplicate.DuplicateCheck",
# "weblate.checks.source.OptionalPluralCheck",
# "weblate.checks.source.EllipsisCheck",
# "weblate.checks.source.MultipleFailingCheck",
# "weblate.checks.source.LongUntranslatedCheck",
# "weblate.checks.format.MultipleUnnamedFormatsCheck",
# )

# List of automatic fixups
# AUTOFIX_LIST = (
# "weblate.trans.autofixes.whitespace.SameBookendingWhitespace",
# "weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis",
# "weblate.trans.autofixes.chars.RemoveZeroSpace",
# "weblate.trans.autofixes.chars.RemoveControlChars",
# )

# List of enabled addons
# WEBLATE_ADDONS = (
# "weblate.addons.gettext.GenerateMoAddon",
# "weblate.addons.gettext.UpdateLinguasAddon",
# "weblate.addons.gettext.UpdateConfigureAddon",
# "weblate.addons.gettext.MsgmergeAddon",
# "weblate.addons.gettext.GettextCustomizeAddon",

```

```

# "weblate.addons.gettext.GettextAuthorComments",
# "weblate.addons.cleanup.CleanupAddon",
# "weblate.addons.consistency.LangaugeConsistencyAddon",
# "weblate.addons.discovery.DiscoveryAddon",
# "weblate.addons.flags.SourceEditAddon",
# "weblate.addons.flags.TargetEditAddon",
# "weblate.addons.flags.SameEditAddon",
# "weblate.addons.flags.BulkEditAddon",
# "weblate.addons.generate.GenerateFileAddon",
# "weblate.addons.json.JSONCustomizeAddon",
# "weblate.addons.properties.PropertiesSortAddon",
# "weblate.addons.git.GitSquashAddon",
# "weblate.addons.removal.RemoveComments",
# "weblate.addons.removal.RemoveSuggestions",
# "weblate.addons.resx.ResxUpdateAddon",
# "weblate.addons.yaml.YAMLCustomizeAddon",
# "weblate.addons.cdn.CDNJSAddon",
# "weblate.addons.autotranslate.AutoTranslateAddon",
# )

# E-mail address that error messages come from.
SERVER_EMAIL = "noreply@example.com"

# Default email address to use for various automated correspondence from
# the site managers. Used for registration emails.
DEFAULT_FROM_EMAIL = "noreply@example.com"

# List of URLs your site is supposed to serve
ALLOWED_HOSTS = ["*"]

# Configuration for caching
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/1",
        # If redis is running on same host as Weblate, you might
        # want to use unix sockets instead:
        # "LOCATION": "unix:///var/run/redis/redis.sock?db=1",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
            "PARSER_CLASS": "redis.connection.HiredisParser",
            "PASSWORD": None,
            "CONNECTION_POOL_KWARGS": {},
        },
        "KEY_PREFIX": "weblate",
    },
    "avatar": {
        "BACKEND": "django.core.cache.backends.filebased.FileBasedCache",
        "LOCATION": os.path.join(DATA_DIR, "avatar-cache"),
        "TIMEOUT": 86400,
        "OPTIONS": {"MAX_ENTRIES": 1000},
    },
}

# Store sessions in cache
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
# Store messages in session
MESSAGE_STORAGE = "django.contrib.messages.storage.session.SessionStorage"

# REST framework settings for API
REST_FRAMEWORK = {
    # Use Django's standard `django.contrib.auth` permissions,
    # or allow read-only access for unauthenticated users.
    "DEFAULT_PERMISSION_CLASSES": [
        # Require authentication for login required sites
        "rest_framework.permissions.IsAuthenticated"
    ]
    if REQUIRE_LOGIN
}

```

```

        else "rest_framework.permissions.IsAuthenticatedOrReadOnly"
    ],
    "DEFAULT_AUTHENTICATION_CLASSES": (
        "rest_framework.authentication.TokenAuthentication",
        "weblate.api.authentication.BearerAuthentication",
        "rest_framework.authentication.SessionAuthentication",
    ),
    "DEFAULT_THROTTLE_CLASSES": (
        "weblate.api.throttling.UserRateThrottle",
        "weblate.api.throttling.AnonRateThrottle",
    ),
    "DEFAULT_THROTTLE_RATES": {"anon": "100/day", "user": "5000/hour"},
    "DEFAULT_PAGINATION_CLASS": ("rest_framework.pagination.
↪PageNumberPagination"),
    "PAGE_SIZE": 20,
    "VIEW_DESCRIPTION_FUNCTION": "weblate.api.views.get_view_description",
    "UNAUTHENTICATED_USER": "weblate.auth.models.get_anonymous",
}

# Fonts CDN URL
FONTS_CDN_URL = None

# Django compressor offline mode
COMPRESS_OFFLINE = False
COMPRESS_OFFLINE_CONTEXT = [
    {"fonts_cdn_url": FONTS_CDN_URL, "STATIC_URL": STATIC_URL, "LANGUAGE_
↪BIDI": True},
    {"fonts_cdn_url": FONTS_CDN_URL, "STATIC_URL": STATIC_URL, "LANGUAGE_
↪BIDI": False},
]

# Require login for all URLs
if REQUIRE_LOGIN:
    LOGIN_REQUIRED_URLS = (r"/(.*)$",)

# In such case you will want to include some of the exceptions
# LOGIN_REQUIRED_URLS_EXCEPTIONS = (
#     rf"{URL_PREFIX}/accounts/(.*)$", # Required for login
#     rf"{URL_PREFIX}/admin/login/(.*)$", # Required for admin login
#     rf"{URL_PREFIX}/static/(.*)$", # Required for development mode
#     rf"{URL_PREFIX}/widgets/(.*)$", # Allowing public access to widgets
#     rf"{URL_PREFIX}/data/(.*)$", # Allowing public access to data exports
#     rf"{URL_PREFIX}/hooks/(.*)$", # Allowing public access to_
↪notification hooks
#     rf"{URL_PREFIX}/healthz/$", # Allowing public access to health check
#     rf"{URL_PREFIX}/api/(.*)$", # Allowing access to API
#     rf"{URL_PREFIX}/js/i18n/$", # JavaScript localization
#     rf"{URL_PREFIX}/contact/$", # Optional for contact form
#     rf"{URL_PREFIX}/legal/(.*)$", # Optional for legal app
# )

# Silence some of the Django system checks
SILENCED_SYSTEM_CHECKS = [
    # We have modified django.contrib.auth.middleware.
↪AuthenticationMiddleware
    # as weblate.accounts.middleware.AuthenticationMiddleware
    "admin.E408"
]

# Celery worker configuration for testing
# CELERY_TASK_ALWAYS_EAGER = True
# CELERY_BROKER_URL = "memory://"
# CELERY_TASK_EAGER_PROPAGATES = True
# Celery worker configuration for production
CELERY_TASK_ALWAYS_EAGER = False
CELERY_BROKER_URL = "redis://localhost:6379"
CELERY_RESULT_BACKEND = CELERY_BROKER_URL

```

```

# Celery settings, it is not recommended to change these
CELERY_WORKER_MAX_MEMORY_PER_CHILD = 200000
CELERY_BEAT_SCHEDULE_FILENAME = os.path.join(DATA_DIR, "celery", "beat-
↳schedule")
CELERY_TASK_ROUTES = {
    "weblate.trans.tasks.auto_translate": {"queue": "translate"},
    "weblate.accounts.tasks.notify_*": {"queue": "notify"},
    "weblate.accounts.tasks.send_mails": {"queue": "notify"},
    "weblate.utils.tasks.settings_backup": {"queue": "backup"},
    "weblate.utils.tasks.database_backup": {"queue": "backup"},
    "weblate.wladmin.tasks.backup": {"queue": "backup"},
    "weblate.wladmin.tasks.backup_service": {"queue": "backup"},
    "weblate.memory.tasks.*": {"queue": "memory"},
}

# Enable plain database backups
DATABASE_BACKUP = "plain"

# Enable auto updating
AUTO_UPDATE = False

# PGP commits signing
WEBLATE_GPG_IDENTITY = None

# Third party services integration
MATOMO_SITE_ID = None
MATOMO_URL = None
GOOGLE_ANALYTICS_ID = None
SENTRY_DSN = None
AKISMET_API_KEY = None

```

Management commands

??: 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).

Invoking management commands

As mentioned before, invocation depends on how you installed Weblate.

If using virtualenv for Weblate, you can either specify the full path to **weblate**, or activate the virtualenv prior to invoking it:

```

# Direct invocation
~/weblate-env/bin/weblate

# Activating virtualenv adds it to search path
. ~/weblate-env/bin/activate
weblate

```

If you are using source code directly (either from a tarball or Git checkout), the management script is `./manage.py` available in the Weblate sources. To run it:

```
python ./manage.py list_versions
```

If you've installed Weblate using the pip or pip3 installer, or by using the `./setup.py` script, the **weblate** is installed to your path (or virtualenv path), from where you can use it to control Weblate:

```
weblate list_versions
```

For the Docker image, the script is installed like above, and you can run it using **docker exec**:

```
docker exec --user weblate <container> weblate list_versions
```

For **docker-compose** the process is similar, you just have to use **docker-compose exec**:

```
docker-compose exec --user weblate weblate weblate list_versions
```

In case you need to pass it a file, you can temporary add a volume:

```
docker-compose exec --user weblate /tmp:/tmp weblate weblate importusers /  
↪tmp/users.json
```

???:

Installing using Docker *Installing on Debian and Ubuntu* *Installing on SUSE and openSUSE* *Installing on RedHat, Fedora and CentOS*

Installing from sources, recommended for development.

add_suggestions

weblate add_suggestions <project> <component> <language> <file>

????? 2.5 ???.

Imports a translation from the file to use as a suggestion for the given translation. It skips duplicated translations; only different ones are added.

--author USER@EXAMPLE.COM

E-mail of author for the suggestions. This user has to exist prior to importing (you can create one in the admin interface if needed).

?:

```
weblate --author michal@cihar.com add_suggestions weblate application cs /  
↪tmp/suggestions-cs.po
```

auto_translate

weblate auto_translate <project> <component> <language>

????? 2.5 ???.

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 [???????](#)).

--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.

?:

```
weblate auto_translate --user nijel --inconsistent --source weblate/  
↪application weblate website cs
```

???:

?????

celery_queues

weblate celery_queues

3.7

Displays length of Celery task queues.

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.

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.

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.

?: 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*.

?:

Running maintenance tasks `COMMIT_PENDING_HOURS`

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.

?:

Running maintenance tasks

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: Added parameters `--username`, `--email`, `--name` and `--update`.

dump_memory

weblate dump_memory

2.20:

Export a JSON file containing Weblate Translation Memory content.

2.20:

2.20: *Weblate*

dumpuserdata

weblate dumpuserdata <file.json>

Dumps userdata to a file for later use by `importuserdata`

2.20: This comes in handy when migrating or merging Weblate instances.

import_demo

weblate import_demo

4.1:

Creates a demo project with components based on <https://github.com/WeblateOrg/demo>.

This can be useful when developing Weblate.

import_json

weblate import_json <json-file>

2.7:

Batch import of components based on JSON data.

The imported JSON file structure pretty much corresponds to the component object (see `GET /api/components/(string:project)/(string:component)/`). You have to include the `name` and `filemask` fields.

--project PROJECT

Specifies where the components will be imported from.

--main-component COMPONENT

Use the given VCS repository from this component for all of them.

--ignore
Skip (already) imported components.

--update
Update (already) imported components.

2.9: The parameters `--ignore` and `--update` are there to deal with already imported components.

Example of JSON file:

```
[
  {
    "slug": "po",
    "name": "Gettext PO",
    "file_format": "po",
    "filemask": "po/*.po",
    "new_lang": "none"
  },
  {
    "name": "Android",
    "filemask": "android/values-*/strings.xml",
    "template": "android/values/strings.xml",
    "repo": "weblate://test/test",
    "file_format": "aresource"
  }
]
```

import_memory

import_memory

weblate import_memory <file>

2.20:

Imports a TMX or JSON file into the Weblate translation memory.

--language-map LANGMAP

Allows mapping languages in the TMX to the Weblate translation memory. The language codes are mapped after normalization usually done by Weblate.

`--language-map en_US:en` will for example import all `en_US` strings as `en` ones.

This can be useful in case your TMX file locales happen not to match what you use in Weblate.

import_project

import_project

weblate import_project <project> <gitrepo> <branch> <filemask>

3.0: The `import_project` command is now based on the `add` command, 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>[^\s/]*) / (?P<component>[^\s/]*)\.po`

The import matches existing components based on files and adds the ones that do not exist. It does not change already existing ones.

--name-template TEMPLATE

Customize the name of a component using Django template syntax.

For example: Documentation: `{{ component }}`

--base-file-template TEMPLATE

Customize the base file for monolingual translations.

For example: `{{ component }}/res/values/string.xml`

--new-base-template TEMPLATE

Customize the base file for addition of new translations.

For example: `{{ component }}/ts/en.ts`

--file-format FORMAT

You can also specify the file format to use (see *Supported file formats*), the default is auto-detection.

--language-regex REGEX

You can specify language filtering (see *Component configuration*) with this parameter. It has to be a valid regular expression.

--main-component

You can specify which component will be chosen as the main one—the one actually containing the VCS repository.

--license NAME

Specify the overall, project or component translation license.

--license-url URL

Specify the URL where the translation license is to be found.

--vcs NAME

In case you need to specify which version control system to use, you can do it here. The default version control is Git.

To give you some examples, let's try importing two projects.

First The Debian Handbook translations, where each language has separate a folder with the translations of each chapter:

```
weblate import_project \  
  debian-handbook \  
  git://anonscm.debian.org/debian-handbook/debian-handbook.git \  
  squeeze/master \  
  '*/**.po'
```

Then the Tanaguru tool, where the file format needs be specified, along with the base file template, and how all components and translations are located in single folder:

```
weblate import_project \  
  --file-format=properties \  
  --base-file-template=web-app/tgol-web-app/src/main/resources/i18n/%s-  
↪I18N.properties \  
  tanaguru \  
  https://github.com/Tanaguru/Tanaguru \  
  master \  
  web-app/tgol-web-app/src/main/resources/i18n/**-I18N_*.properties
```

More complex example of parsing of filenames to get the correct component and language out of a filename like `src/security/Numerous_security_holes_in_0.10.1.de.po`:

```
weblate import_project \  
  tails \  
  git://git.tails.boum.org/tails master \  
  'wiki/src/security/(?P<component>.*).\.(?P<language>[^\.]*)\.po$'
```

Filtering only translations in a chosen language:

```
./manage import_project \  
  --language-regex '^ (cs|sk)$' \  
  weblate \  
  https://github.com/WeblateOrg/weblate.git \  
  'weblate/locale/*/LC_MESSAGES/**/*.po'
```

Importing Sphinx documentation split to multiple files:

```
$ weblate import_project --name-template 'Documentation: %s' \
--file-format po \
project https://github.com/project/docs.git master \
'docs/locale/*/LC_MESSAGES/**/*.po'
```

Importing Sphinx documentation split to multiple files and directories:

```
$ weblate import_project --name-template 'Directory 1: %s' \
--file-format po \
project https://github.com/project/docs.git master \
'docs/locale/*/LC_MESSAGES/dir1/**/*.po'
$ weblate import_project --name-template 'Directory 2: %s' \
--file-format po \
project https://github.com/project/docs.git master \
'docs/locale/*/LC_MESSAGES/dir2/**/*.po'
```

??:

More detailed examples can be found in the *Starting with internationalization* chapter, alternatively you might want to use *import_json*.

importuserdata

weblate importuserdata <file.json>

Imports user data from a file created by *dumpuserdata*

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
```

install_addon

????? 3.2 ????

weblate install_addon --addon ADDON <project|project/component>

Installs an addon to a set of components.

--addon ADDON

Name of the addon to install. For example *weblate.gettext.customize*.

--configuration CONFIG

JSON encoded configuration of an addon.

--update

Update the existing addon configuration.

You can either define which project or component to install the addon in (for example *weblate/application*), or use *--all* to include all existing components.

To install *gettext* ?????????? for all components:

```
weblate install_addon --addon weblate.gettext.customize --config '{"width
↵": -1}' --update --all
```

??:

????

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.110n.cz/Slovn%C3%ADk_s_n%C3%A1zvy_jazyk%C5%AF>.

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.

list_versions

weblate list_versions

Lists all Weblate dependencies and their versions.

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.

???: 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.

lock_translation

weblate lock_translation <project|project/component>

Prevents further translation of a component.

???: 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.

???:

`unlock_translation`

move_language

weblate move_language source target

weblate 3.0

Allows you to merge language content. This is useful when updating to a new version which contains aliases for previously unknown languages that have been created with the *(generated)* suffix. It moves all content from the *source* language to the *target* one.

Example:

```
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.

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.

weblate: Weblate pushes changes automatically if *Push on commit* in *Component configuration* is turned on, which is the default.

unlock_translation

weblate unlock_translation <project|project/component>

Unlocks a given component, making it available for translation.

weblate: 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.

Example:

```
lock_translation
```

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 [weblate 3.0](#).

Example:

```
setupgroups
```

setuplang

weblate setuplang

Updates list of defined languages in Weblate.

--no-update

Turns off automatic updates of existing languages (only adds new ones).

updatechecks

weblate updatechecks <project|project/component>

Updates all checks for all strings.

???: 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.

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.

??: Usually it is better to configure hooks in the repository to trigger **?????**, instead of regular polling by `updategit`.

????

???? 4.0 ??: 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 be posted on each level in the *Manage* menu, using *Post announcement*:

Webate Dashboard Projects Languages Checks

WeblateOrg translated 90%

Translations will be used only if they reach 60%

Components Languages Info Search Glossaries Insights Files Tools Manage Share Not watching

Post announcement

Message

You can use Markdown and mention users by @username.

Category

Info (light blue)

Category defines color used for the message.

Expiry date

mm/dd/yyyy

The message will be not shown after this date. Use it to announce string freeze and translation deadline for next release.

Notify users

The message is shown for all translations within the project, until its given expiry, or permanently until it is deleted.

Add

It can be also added using the admin interface:

Add Announcement

Required fields are marked in bold.

Message:

You can use Markdown and mention users by @username.

Project:  

Component:  

Language:  

Category: 

Category defines color used for the message.

Expiry date: 

The message will be not shown after this date. Use it to announce string freeze and translation deadline for next release.

Notify users

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:

Navigation: Weblate Dashboard Projects Languages Checks

Language: Languages / Czech

Announcement: Czech translators rock!

Project: WeblateOrg

Project	Translated	Strings of total	Untranslated	Untranslated words	Checks	Suggestions	Comments
WeblateOrg	97%	97%	1	12	3		

Getting started

Getting started with Weblate 2.20. This page is part of the Weblate documentation. [Return to Weblate](#) / [Documentation](#) / [Change password](#) / [Sign out](#) 1

Weblate 2.20 **new**: [New features](#)

[Getting started](#) / [Component lists](#) / [Automatic component list assignment](#) / [URLs](#)

Weblate 2.13 **new**: [New features](#)

Getting started

Weblate 2.13 **new**.

Automatic component list assignment [Automatic component list assignment](#)

[Getting started](#) / [Component lists](#) / [Automatic component list assignment](#) / [URLs](#) Weblate [Getting started](#) 1 [Getting started](#)

new: Weblate [Getting started](#) / [Component lists](#) / [Automatic component list assignment](#) / [URLs](#)

1. Define *Automatic component list assignment* with `^.*$` as regular expression in both the project and the component fields, as shown on this image:

The screenshot shows the Weblate administration interface for adding a component list. The page title is "Weblate administration" and the breadcrumb is "Home > Weblate translations > Component lists > Add Component list". The main heading is "Add Component list". A note states "Required fields are marked in bold." The form contains the following fields:

- Component list name:** Input field with value "All components" (Display name).
- URL slug:** Input field with value "all-components" (Name used in URLs and filenames).
- Show on dashboard** (When enabled this component list will be shown as a tab on the dashboard).

Below the form are two lists:

- Available components:** A list with a search filter and two items: "WeblateOrg/Django" and "WeblateOrg/Language names". A "Choose all" button is at the bottom.
- Chosen components:** An empty list with a "Remove all" button at the bottom.

A note below the lists says: "Hold down 'Control', or 'Command' on a Mac, to select more than one."

At the bottom, there is a section titled "AUTOMATIC COMPONENT LIST ASSIGNMENTS" with a table:

PROJECT REGULAR EXPRESSION	COMPONENT REGULAR EXPRESSION	DELETE?
<input type="text" value="^.*\$"/>	<input type="text" value="^.*\$"/>	✕

Below the table is a link: "+ Add another Automatic component list assignment". At the very bottom are three buttons: "Save and add another", "Save and continue editing", and "SAVE".

Optional Weblate modules

Several optional modules are available for your setup.

Git exporter

2.10

Provides you read-only access to the underlying Git repository using HTTP(S).

??

1. Add `weblate.gitexport` to installed apps in `settings.py`:

```
INSTALLED_APPS += (  
    'weblate.gitexport',  
)
```

2. Export existing repositories by migrating your database after installation:

```
weblate migrate
```

Usage

The module automatically hooks into Weblate and sets the exported repository URL in the *Component configuration*. The repositories are accessible under the `/git/` part of the Weblate URL, for example `https://example.org/git/weblate/master/`:

```
git clone 'https://example.org/git/weblate/master/'
```

Repositories are available anonymously unless `ANONYMOUS_REPO` is turned on. This requires authenticate using your API token (it can be obtained in your `settings.py`):

```
git clone 'https://user:KEY@example.org/git/weblate/master/'
```

??

2.4

This is used on [Hosted Weblate](#) to define billing plans, track invoices and usage limits.

??

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 [admin interface](#).

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.

Legal documents

[Legal documents](#) 2.15 [Legal documents](#).

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:

Terms of service document

Privacy policy document

Short overview of the terms of service and privacy policy

Legal documents for the Hosted Weblate service is available in this Git repository <<https://github.com/WeblateOrg/hosted/tree/master/wlhosted/legal/templates/legal/documents>>.

Most likely these will not be directly usable to you, but might come in handy as a starting point if adjusted to meet your needs.

Installation

1. Add `weblate.legal` to installed apps in `settings.py`:

```
INSTALLED_APPS += (
    'weblate.legal',
)

# Optional:
# Social auth pipeline to confirm TOS upon registration/subsequent login
SOCIAL_AUTH_PIPELINE += (
    'weblate.legal.pipeline.tos_confirm',
)

# Middleware to enforce TOS confirmation of signed in users
MIDDLEWARE += [
    'weblate.legal.middleware.RequireTOSMiddleware',
]
```

2.Run the database migration to optionally install additional database structures for the module:

```
weblate migrate
```

3.Edit the legal documents in the `weblate/legal/templates/legal/` folder to match your service.

Usage

After installation and editing, the legal documents are shown in the Weblate UI.

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

☒:

`AVATAR_URL_PREFIX` `ENABLE_AVATARS`

Spam protection

You can protect against suggestion spamming by unauthenticated users by using the [akismet.com](https://www.akismet.com) service.

1. Install the `akismet` Python module
2. Configure the Akismet API key.

☒: This (among other things) relies on IP address of the client, please see *Running behind reverse proxy* for properly configuring that.

☒:

Running behind reverse proxy `AKISMET_API_KEY`

Signing Git commits with GnuPG

☒☒☒☒☒ 3.1 ☒☒☒.

All commits can be signed by the GnuPG key of the Weblate instance.

1. Turn on `WEBLATE_GPG_IDENTITY`. (Weblate will generate a GnuPG key when needed and will use it to sign all translation commits.)

This feature needs GnuPG 2.1 or newer installed.

You can find the key in the `DATA_DIR` and the public key is shown on the "About" page:

SSH key ⓘ

SSH key not available.

Commit signing ⓘ

All commits made with Weblate are signed with the GPG key 708ED01754D9C1DA601F9F7734C4F70EEFBE4673, for which the corresponding public key is found below.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
mQGNBF+IPS0BDADnFFIS/jmOQ7uvncUTNlcUcvgaG48tISAX8WTEG2FxfWga3Fl
q67XFkTfYo0abXcRSCOjzsl+0ugalQcA5HEQTlpaP1b9AMPwUq8DALkKsIC6jen
8UYZkvdYB+CUbAWI8Z836HUJpQ1wZ057pPrB2u1u7pYP726RyR9JpLpq2FUg+piY
vmYD3yMiuifjPSzJCJtjTN92rxZaX3xHHKnr6jiRM4Zy4vXn0ITze4jMdmKj8Pf
rBnAhYrtjYcYQVp9RQAXd3w+ZvHIEkPICxEkQ1D8IRQ9qsDHbztP8inkrD7d1q
tWjd5rnfVWM6VIHsXycl5Rq5vwxknWI8QsEj9umfyI86qrc0oSbDEtgcxkKcvQv
5GaLzGHTNB9+S+2Gby7Q+R/CUGYRluPZChsEllwsilFLKTfQfevd1c9mwrOQX1hg
PFuQzysR94R2B19lc8A9abHTV2chp+AJ57/TMV/YyjqUAJvQytdQmKWTJqODUIh
LSn3EYNI70zevM8AEQEAAbQdV2VibGF0ZSA8d2VibGF0ZUBleGFtcGxlMmNvbT6J
Ac4EEwEKADgWIQRWjtAXVNnB2mAfnc0xPcO775GcwUCX4g9LQIbAwULCQgHAgV
CgKIcWIEFgIDAQIeAQIXgAAKCRAOxPcO775GcwUCX4g9LQIbAwULCQgHAgV
ZBgZlI7s4b4WBs37cL8/KNR0hcrW5Vj/e+MLMdReRvTSQQnc93uTZbz+Q3vcOb1s
IWWBGkqEsV5gk8d4gumCd2Rj2MW7+cnDCcGhNj2ToL1plD884GldGy9P4j2tdMy
87h4Ghf0NDTd7Csh3/SCNKH4kMTITXTpWpKu6ktKXl3O594Gmsc8ChGn7sWFO9Gk
```

2. Alternatively you can also import existing keys into Weblate, just set HOME=\$DATA_DIR/home when invoking gpg.

??:

WEBLATE_GPG_IDENTITY

?????

????? 3.2 ????: ???

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.

????????????????????????????????

??	??	????????????????????	????????????????	????????????????
??	REGISTRATION	5	300	600
????????????????	MESSAGE	5	300	600
????????????????	LOGIN	5	300	600
????????????	SEARCH	6	60	60
??	TRANSLATE	30	60	600
????????????	GLOSSARY	30	60	600

????????????? AUTH_LOCK_ATTEMPTS ???

??:

?????Running behind reverse proxy

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.

⚠️: 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.

⚠️:

Weblate [⚠️⚠️⚠️](#)

Creating a Python module

If you are not familiar with Python, you might want to look into [Python For Beginners](#), explaining the basics and pointing to further tutorials.

To write some custom Python code (called a module), a place to store it is needed, either in the system path (usually something like `/usr/lib/python3.7/site-packages/`) or in the Weblate directory, which is also added to the interpreter search path.

Better yet, turn your customization into a proper Python package:

1. Create a folder for your package (we will use `weblate_customization`).
2. Within it, create a `setup.py` file to describe the package:

```
from setuptools import setup

setup(
    name = "weblate_customization",
    version = "0.0.1",
    author = "Your name",
    author_email = "yourname@example.com",
    description = "Sample Custom check for Weblate.",
    license = "GPLv3+",
    keywords = "Weblate check example",
    packages=['weblate_customization'],
)
```

3. Create a folder for the Python module (also called `weblate_customization`) for the customization code.
4. Within it, create a `__init__.py` file to make sure Python can import the module.
5. This package can now be installed using `pip install -e`. More info to be found in [“Editable” Installs](#).
6. Once installed, the module can be used in the Weblate configuration (for example `weblate_customization.checks.FooCheck`).

Your module structure should look like this:

```
weblate_customization
├── setup.py
└── weblate_customization
    ├── __init__.py
    ├── addons.py
    └── checks.py
```

You can find an example of customizing Weblate at <https://github.com/WeblateOrg/customize-example>, it covers all the topics described below.

Changing the logo

1. Create a simple Django app containing the static files you want to overwrite (see *Creating a Python module*).
2. Add it to `INSTALLED_APPS`:

```
INSTALLED_APPS = (  
    # Add your customization as first  
    'weblate_customization',  
  
    # Weblate apps are here...  
)
```

Branding appears in the following files:

Logo shown in the navigation bar.

Web icons depending on screen resolution and web-browser.

Web icon used by legacy browsers.

Avatars for bots or anonymous users. Some web-browsers use these as shortcut icons.

Used in notifications e-mails.

3. Run `weblate collectstatic --noinput`, to collect static files served to clients.

??:

Managing static files (e.g. images, JavaScript, CSS) [Writing addon](#) [Executing scripts from addon](#)

??

???????????? Weblate [Writing addon](#) [Executing scripts from addon](#):

1. Weblate [Writing addon](#) [Executing scripts from addon](#) Python [Writing addon](#) [Executing scripts from addon](#):ref:custom-module
2. Python [Writing addon](#) [Executing scripts from addon](#) `WEBLATE_ADDONS` `CHECK_LIST` `AUTOFIX_LIST`:

```
# Checks  
CHECK_LIST += (  
    'weblate_customization.checks.FooCheck',  
)  
  
# Autofixes  
AUTOFIX_LIST += (  
    'weblate_customization.autofix.FooFixer',  
)  
  
# Addons  
WEBLATE_ADDONS += (  
    'weblate_customization.addons.ExamplePreAddon',  
)
```

??:

[Writing addon](#) [Executing scripts from addon](#)

Management interface

The management interface offer administration settings under the `/management/` URL. It is available for users signed in with admin privileges, accessible by using the wrench icon top right:

[Weblate](#)
[Dashboard](#)
[Projects](#)
[Languages](#)
[Checks](#)
+ Add

[Manage](#)

[Weblate status](#)
[Backups](#)
[Translation memory](#)
[Performance report](#)
[SSH keys](#)
[Alerts](#)
[Repositories](#)
[Users](#)
[Tools](#)

Weblate support status ⓘ

Support status Community support

[Purchase support package](#)
[Donate to Weblate](#)

Activate support package ⓘ

The support packages include priority e-mail support, or cloud backups of your Weblate installation.

Activation token

Please enter the activation token obtained when making the subscription.

[Activate](#)
[Purchase support package](#)

Powered by Weblate 4.3 [About Weblate](#) [Legal](#) [Contact](#) [Documentation](#) [Donate to Weblate](#)

The Django admin interface

⚠️: 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:

Site administration

REPORTS		
Weblate support status		
Status of repositories		
SSH keys		
Performance report		
Translation memory		
ACCOUNTS		
Audit logs	+ Add	Change
Profiles	+ Add	Change
Verified emails	+ Add	Change
AUTH TOKEN		
Tokens	+ Add	Change
AUTHENTICATION		
Groups	+ Add	Change
Roles	+ Add	Change
Users	+ Add	Change
BILLING		
Billings	+ Add	Change
Invoices	+ Add	Change
Plans	+ Add	Change
FONTS		
Font groups	+ Add	Change
Fonts	+ Add	Change
GLOSSARIES		
Glossaries	+ Add	Change
LEGAL		
Agreements	+ Add	Change
PYTHON SOCIAL AUTH		
Associations	+ Add	Change
Nonces	+ Add	Change
User social auths	+ Add	Change
SCREENSHOTS		
Screenshots	+ Add	Change
TRANSLATION MEMORY		
Memors	+ Add	Change
WEBLATE LANGUAGES		
Languages	+ Add	Change
WEBLATE TRANSLATIONS		
Announcements	+ Add	Change
Component lists	+ Add	Change
Components	+ Add	Change
Contributor agreements	+ Add	Change
Projects	+ Add	Change

Recent actions

My actions

None available

In the *Reports* section, you can check the status of your site, tweak it for *Production setup*, or manage SSH keys used to access [\[redacted\]](#).

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 *Language definitions*.

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 WELCOME, WEBLATE TEST [RETURN TO WEBLATE](#) / [DOCUMENTATION](#) / [CHANGE PASSWORD](#) / [SIGN OUT](#)

Home · Weblate translations · Projects · Add Project

Add Project

Required fields are marked in bold.

Project name:
Display name

URL slug:
Name used in URLs and filenames.

Project website:
Main website of translated project.

Mailing list:
Mailing list for translators.

Translation instructions:
You can use Markdown and mention users by @username.

Set "Language-Team" header
Lets Weblate update the "Language-Team" file header of your project.

Use shared translation memory
Uses the pool of shared translations between projects.

Contribute to shared translation memory
Contributes to the pool of shared translations between projects.

Access control: How to restrict access to this project is detailed in the documentation.

Enable reviews
Requires dedicated reviewers to approve translations.

Enable source reviews
Requires dedicated reviewers to approve source strings.

Enable hooks
Whether to allow updating this repository by remote hooks.

Language aliases:
Comma-separated list of language code mappings, for example: en_GB:en,en_US:en



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):

Webdate administration
WELCOME WELCOME TEST RETURN TO PROFILE DOCUMENTATION CHANGE PASSWORD LOG OUT

[Home](#) [Website Translations](#) [Components](#) [Add Component](#)

Add Component Website error documentation

Required fields are marked in bold>.

Component name:	<input type="text" value="Language names"/>
	<small>Display name</small>
URL slug:	<input type="text" value="language-names"/>
	<small>Name used in URLs and folders</small>
Project:	<input type="text" value="WebdateOrg"/> + -
Version control system:	<input type="text" value="Git"/>
	<small>Version control system to use to access your repository containing translations. You can also choose additional integration with third party providers to submit merge requests</small>
Source code repository:	<input type="text" value="https://github.com/WebdateOrg/demo.git"/>
	<small>URL of a repository you wish to use (component to share in with other components)</small>
Repository push URL:	<input type="text" value="https://github.com/WebdateOrg/demo.git"/>
	<small>URL of a push repository pushing to branch off if empty</small>
Repository browser:	<input type="text" value="https://github.com/WebdateOrg/demo.git/branches/{language_name}"/>
	<small>Link to repository browser, use {language_name} for branch, {component_name} and {url} for language and file placeholders</small>
Exported repository URL:	<input type="text"/>
	<small>URL of repository where users can fetch strings from Webdate</small>
Source string bug reporting address:	<input type="text"/>
	<small>Email address for reports on errors in source strings. Leave empty for no email</small>
Repository branch:	<input type="text"/>
	<small>Repository branch to translate</small>
Push branch:	<input type="text"/>
	<small>Branch for pushing changes, leave empty to use repository default</small>
Flaremark:	<input type="text" value="webdate/language/{LC_MESSAGES}/%s"/>
	<small>Path of file to translate relative to repository root, use %s instead of language code, for example path for locale/LC_MESSAGES/strings</small>
Missing all base language file:	<input type="text"/>
	<small>Filename of translation base file, containing all strings and their sources, it is recommended for missing translation features</small>
<input checked="" type="checkbox"/> Edit base file	
	<small>Whether users will be able to edit the base file for missing translations</small>
Intermediate language file:	<input type="text"/>
	<small>Filename of intermediate translation file, it must contain this is a translation file provided by developers and is used when creating actual source strings</small>
Template for new translations:	<input type="text" value="webdate/language/{locale}/strings.pot"/>
	<small>Filename of file used for creating new translations, for getting choose pot file</small>
File format:	<input type="text" value="gettext PO file"/>
<input type="checkbox"/> Locked	<small>Locked components will not get any translation updates</small>
<input checked="" type="checkbox"/> Allow translation propagation	<small>Whether translation updates in other components will create automatic translations in this one</small>
<input checked="" type="checkbox"/> Turn on suggestions	<small>Whether to allow translation suggestions email</small>
<input type="checkbox"/> Suggestion writing	<small>Whether users can write for suggestions</small>
Advanced suggestions:	<input type="text" value="0"/>
	<small>Automatically send suggestions with the number of notes, set to 0 to turn it off</small>
Translation flags:	<input type="text"/>
	<small>Additional comma separated flags to influence quality checks. Possible values can be found in the documentation</small>
Enforced checks:	<input type="text"/>
	<small>List of checks which can not be ignored</small>
Translation license:	<input type="text" value="GNU General Public License v3.0 or later"/>
Contributor agreement:	<input type="text"/>
	<small>User agreement which needs to be approved before a user can translate this component</small>
Adding new translation:	<input type="text" value="Create new language file"/>
	<small>How to handle requests for creating new translations</small>
Language code style:	<input type="text" value="Default based on the file format"/>
	<small>Customize language code used to generate the filename for translations created by Webdate</small>
Merge style:	<input type="text" value="Rebase"/>
	<small>Define whether Webdate should merge the upstream repository or rebase changes onto it</small>
Commit message when translating:	<input type="text" value="Translated using Webdate (language_name)"/> Currently translated at state translated_percent % (state all string) Translate: project_name (component_name) Translate URL: url
	<small>You can use template language for various info, please consult the documentation for more details</small>
Commit message when adding translation:	<input type="text" value="Added translation using Webdate (language_name)"/>
	<small>You can use template language for various info, please consult the documentation for more details</small>
Commit message when removing translation:	<input type="text" value="Deleted translation using Webdate (language_name)"/>
	<small>You can use template language for various info, please consult the documentation for more details</small>
Commit message when merging translation:	<input type="text" value="Merge branch component_name_branch into Webdate"/>
	<small>You can use template language for various info, please consult the documentation for more details</small>
Commit message when adding a new change:	<input type="text" value="Update translation files"/> Updated by editor_name from locale Translate: project_name (component_name) Translate URL: url
	<small>You can use template language for various info, please consult the documentation for more details</small>
Contributor name:	<input type="text" value="Webdate"/>
Contributor e-mail:	<input type="text" value="noreply@webdate.org"/>
<input checked="" type="checkbox"/> Push on commit	<small>Whether the repository should be pushed upstream on every commit</small>
Age of changes to commit:	<input type="text" value="24"/>
	<small>Time in hours after which any pending changes will be committed to the VCS</small>
<input checked="" type="checkbox"/> Lock on error	<small>Whether the component should be locked on repository errors</small>
Source language:	<input type="text" value="English"/>
	<small>Language used for source strings in all components</small>
Language filter:	<input type="text" value="tr[he hu id]"/>
	<small>Regular expression used to filter translation when scoring for Flaremark</small>
Variables regular expressions:	<input type="text"/>
	<small>Regular expressions used to determine contents of a string</small>
Priority:	<input type="text" value="Medium"/>
	<small>Components with higher priority are offered first to translators</small>
<input type="checkbox"/> Restricted component	<small>Restrict access to the component to only those explicitly given permission</small>

Save and add another
Save and continue editing
Save

❓:

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):

Webiate administration
WELCOME WILDALE TEST RETURN TO PROFILE / DOCUMENTATION / CHANGE PASSWORD / LOG OUT

Home / Website Translations / Components / Add Component
(current step: documentation)

Add Component

Required fields are marked in bold>.

Component name: Display name

URL slug: Name used in URLs and filenames

Project: Name used in URLs and filenames

Version control system: Version control system to use to access your repository containing translations. You can also choose additional integration with third party providers to submit merge requests

Source code repository: URL of a repository you wish to use to store your translations. If a repository you wish to use to store your translations is shared with other components

Repository push URL: URL of a push-repository pushing is based off of (e.g. https://github.com)

Repository browser: Link to repository browser, use [owner] for branch, [owner] and [id] as filename and file placeholders

Exported repository URL: URL of repository where users can fetch strings from Webiate

Source string bug reporting address: Email address for reports on errors in source strings. Leave empty for no email

Repository branch: Repository branch to translate

Push branch: Branch for pushing changes, leave empty to use repository default

Filename: Path of file to translate relative to repository root, use "*" instead of language code, for example: src/main/res/values/*/IC_MESSAGES/strings.xml

Mandatory base language file: Filename of translation base file, containing all strings and their sources. It is recommended for monolingual translation features

Edit base file
Whether users will be able to edit the base file for monolingual translations.

Intermediate language file: Filename of intermediate translation file. It must contain this is a translation file provided by developers and is used when creating actual source strings

Template for new translations: Filename of file used for copying new translations, for getting chosen .pot file

File format: File format used for saving translations

Locked
Locked components will not get any translation updates.

Allow translation propagation
Whether translation updates in other components will create automatic translations in this one

Turn on suggestions
Whether to allow translation suggestions email

Suggestion writing
Whether users can write for suggestions

Advanced suggestions: Automatically send suggestions with the number of notes, and it is type 0 off

Translation flags: Additional comma separated flags to influence quality checks. Placeholder values can be found in the documentation

Enforced checks: List of checks which can not be ignored

Translation license: License used for translations

Contributor agreement: User agreement which needs to be approved before a user can translate this component

Adding new translation: How to handle requests for creating new translations

Language code style: Customize language code used to generate the filename for translations created by Webiate

Merge style: Define whether Webiate should merge the upstream repository or release changes only

Commit message when translating:

 You can use template language for various info, please consult the documentation for more details.

Commit message when adding translation:
 You can use template language for various info, please consult the documentation for more details.

Commit message when removing translation:
 You can use template language for various info, please consult the documentation for more details.

Commit message when merging translation:
 You can use template language for various info, please consult the documentation for more details.

Commit message when adding a new change:

 You can use template language for various info, please consult the documentation for more details.

Contributor name: Contributor name

Contributor e-mail: Contributor email

Push on commit
Whether the repository should be pushed after every commit

Age of changes to commit: Time in hours after which any pending changes will be committed to the VCS

Lock on error
Whether the component should be locked on repository errors

Source language: Language used for source strings in all components

Language filter: Regular expression used to filter translation when searching for filename

Variables regular expression: Regular expressions used to determine contents of a string

Priority: Component with higher priority are offered first to translators

Restricted component
Restrict access to the component to only those explicitly given permission

??:

Component configuration [Bilingual and monolingual formats](#)

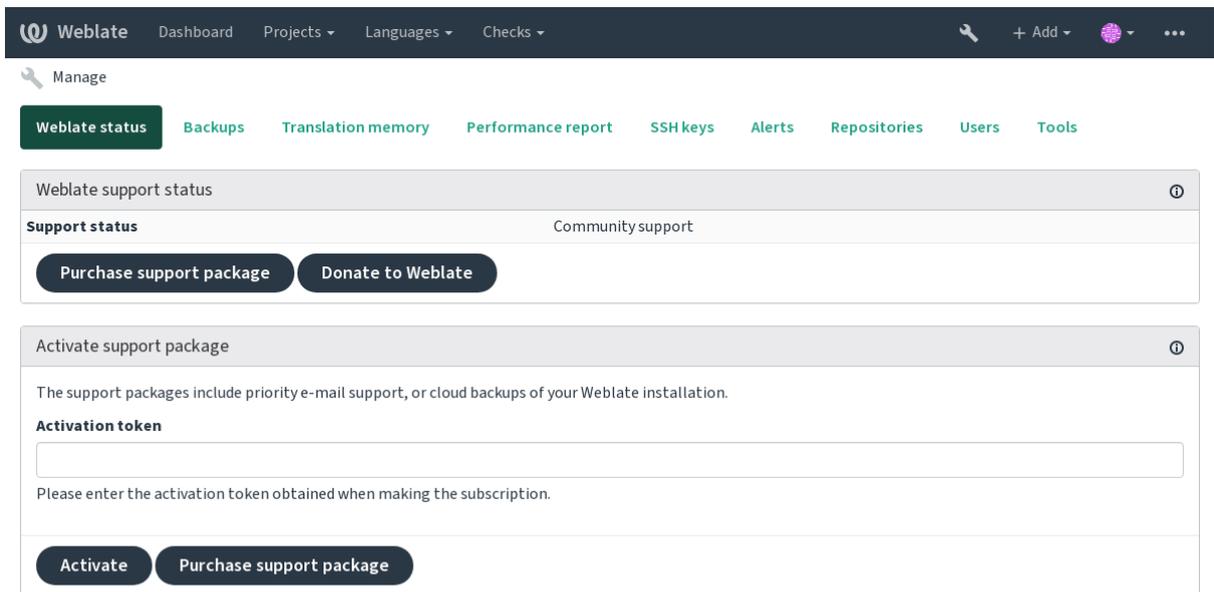
Getting support for Weblate

Weblate <https://weblate.org/support/>

Integrating support

3.8

Purchased support packages can optionally be integrated into your Weblate [subscription management](#) interface, from where you will find a link to it. Basic instance details about your installation are also reported back to Weblate this way.



Powered by Weblate 4.3 [About Weblate](#) [Legal](#) [Contact](#) [Documentation](#) [Donate to Weblate](#)

Data submitted to the Weblate

URL where your Weblate instance is configured

Your site title

The Weblate version you are running

Tallies of some objects in your Weblate database (projects, components, languages, source strings and users)

The public SSH key of your instance

No other data is submitted.

Integration services

See if your support package is still valid

Weblate provisioned backup storage

!!!: Purchased support packages are already activated upon purchase, and can be used without integrating them.

Legal documents

!: 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.

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.

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):

See https://wiki.python.org/moin/PythonSoftwareFoundationLicenseFaq#Is_Python_subject_to_export_laws.3F

Optionally used by Weblate

Optionally used by Weblate

Used by Git

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:

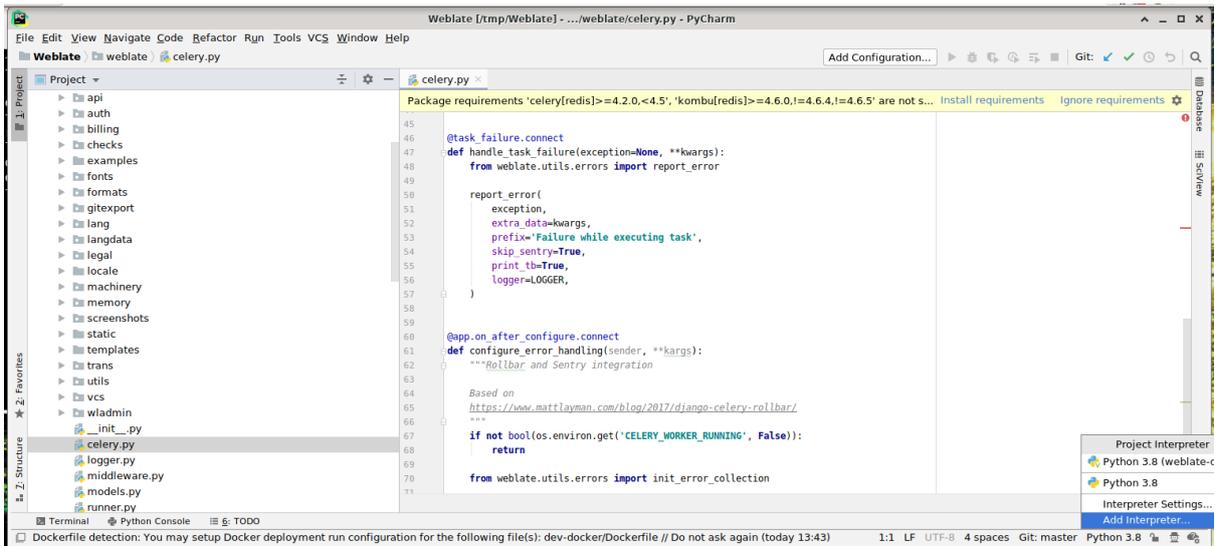
Accessing remote servers using secure protocols (HTTPS)

Generating signatures for code commits (PGP)

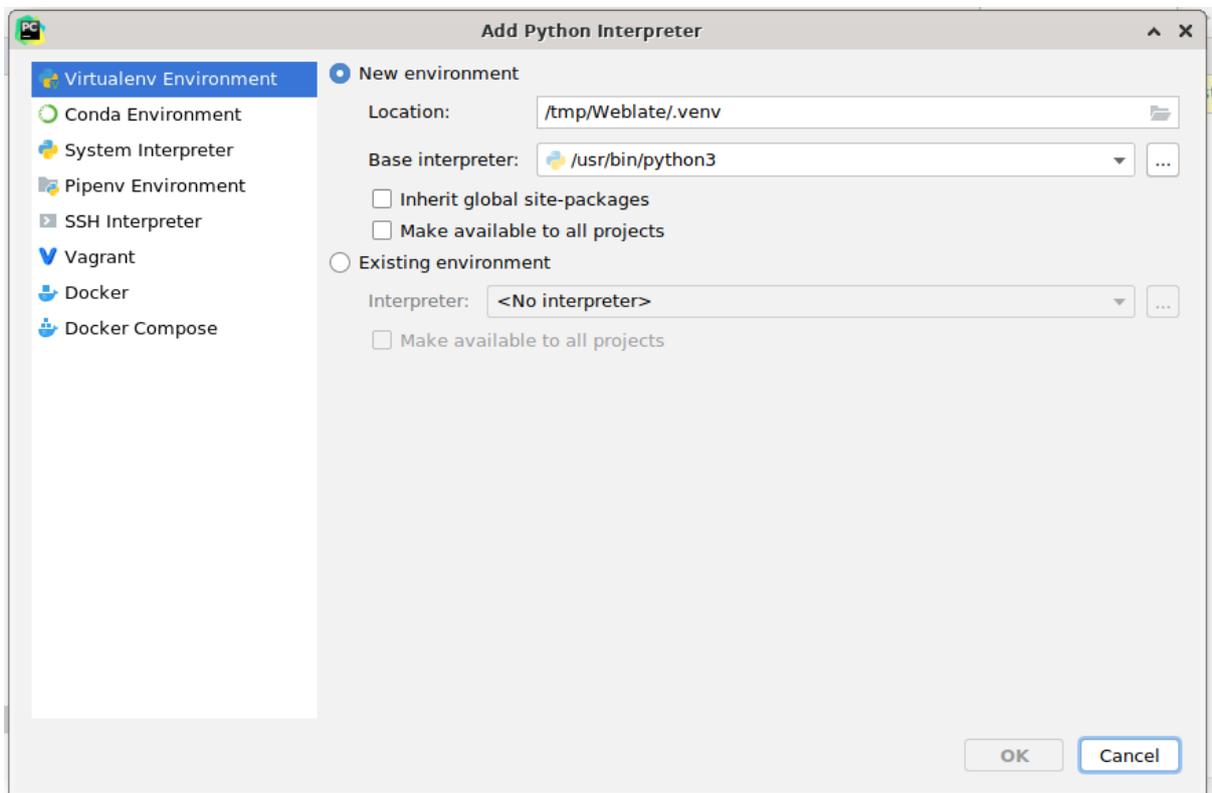
!:

[Export Controls \(EAR\) on Open Source Software](#)

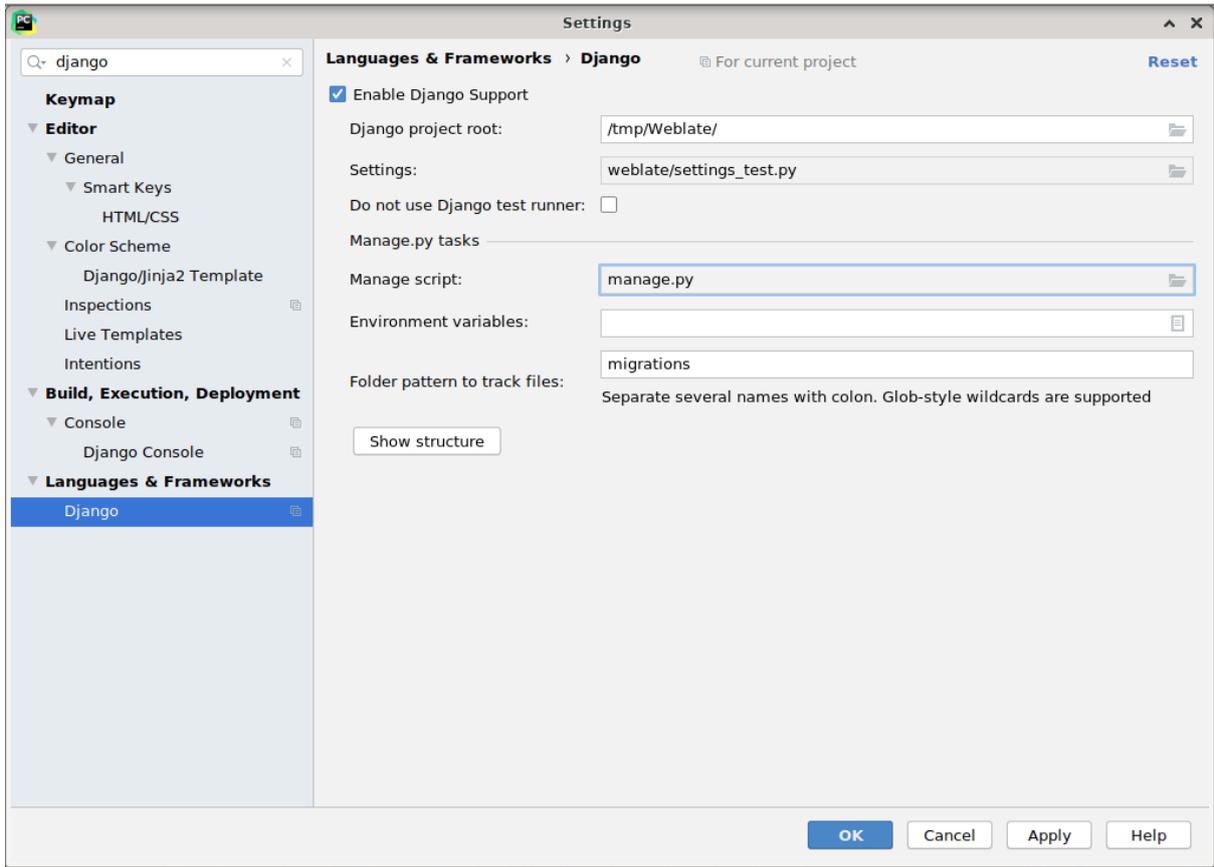
!!!!!!!!!!!!



PyCharm virtualenv

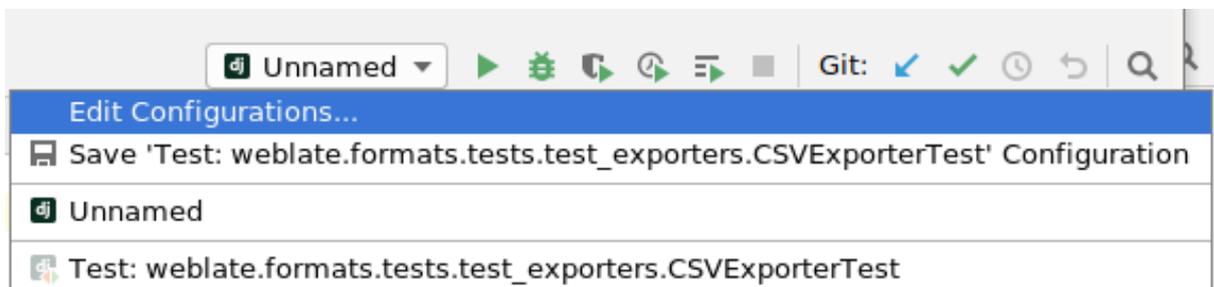


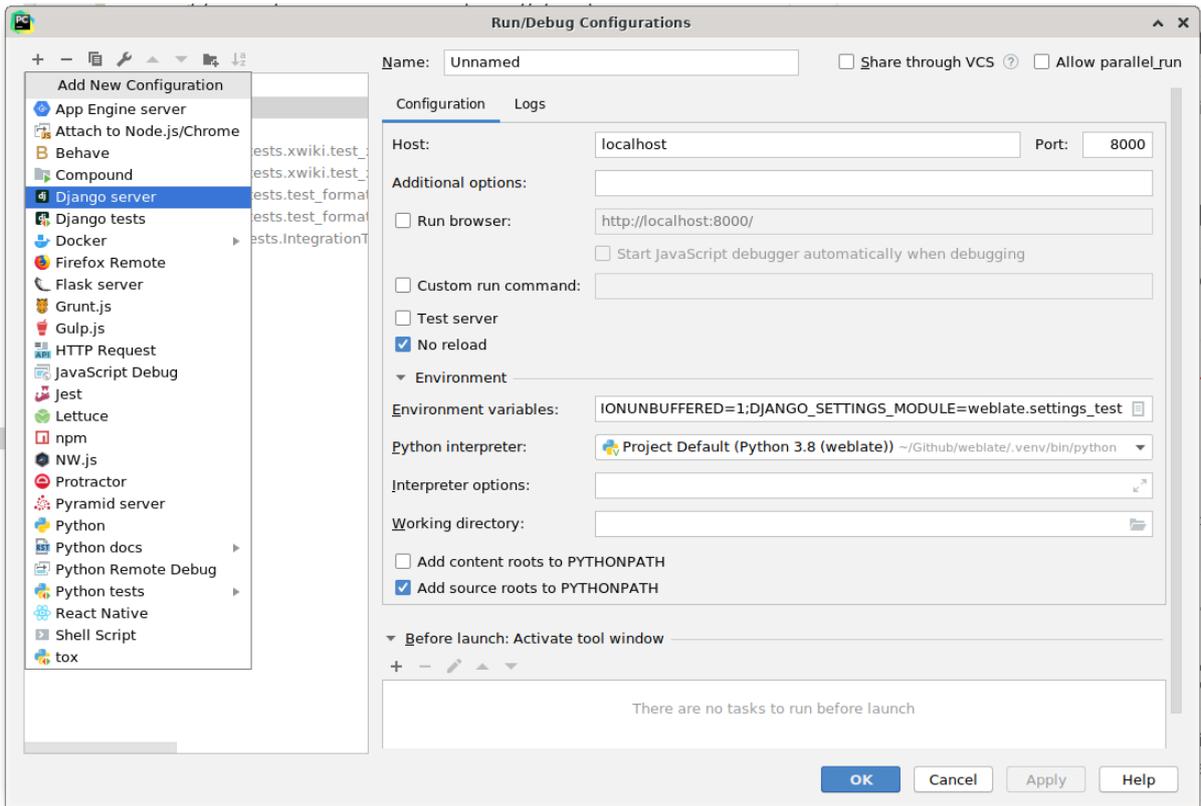
virtualenv PyCharm Django IDE virtualenv PyCharm Django IDE



Django weblate settings_test

Django Server:





Weblate

Installing from sources

```
pip install -r requirements-dev.txt
```

:

```
weblate runserver
```

Celery Worker

```
./weblate/examples/celery start
```

Docker Weblate

Docker docker-compose

```
./rundev.sh
```

Docker <http://127.0.0.1:8080/> admin admin
Adding translation projects and components

Dockerfile docker-compose.yml dev-docker

test test

```
./rundev.sh test --failfast weblate.trans
```

Docker docker ps

:

```
./rundevel.sh stop
```

```
docker-compose up --build
```

```
cd /opt/weblate
```

```
git clone https://github.com/Weblate/weblate.git
```

```
import_demo createadmin
```

Weblate source code

Weblate is developed on [GitHub](#). You are welcome to fork the code and open pull requests. Patches in any other form are welcome too.

Tip:

Check out [Weblate](#) to see how Weblate looks from inside.

Security by Design Principles

Any code for Weblate should be written with [Security by Design Principles](#) in mind.

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
```

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](#).

```
DEBUG=1
```

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.

Tip:

Disable debug mode

Weblate logs

Weblate can produce detailed logs of what is going in the background. In the default configuration it uses syslog and that makes the log appear either in `/var/log/messages` or `/var/log/syslog` (depending on your syslog daemon configuration).

Docker containers log to their output (as usual in the Docker world), so you can look at the logs using `docker-compose logs`.

??:

`weblate.conf` contains LOGGING configuration.

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*.

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.

Performance issues

In case Weblate performs badly in some situation, please collect the relevant logs showing the issue, and anything that might help figuring out where the code might be improved.

In case some requests take too long without any indication, you might want to install *dogslow* <<https://pypi.org/project/dogslow/>> along with *Collecting error reports* and get pinpointed and detailed tracebacks in the error collection tool.

Weblate ???

??: `weblate` Weblate `django`

Weblate `django` Django `django`

???

Weblate `django` `django`:

`django` `sphinx` `django`

`django` `docker` `weblate` `django`

`django` `django` `django` `django` `weblate` `django` `weblate` `django`

`django` `django` `django` `weblate` `django`

???

Weblate `django` Django `django` *Optional Weblate modules*??

`accounts`

`django` `django`

`addons`

Weblate `django` `django` `django`

`api`

Django REST framework `django` API??

`auth`

billing
 checks
 fonts
 formats
 translate-toolkit
 gitexport
 lang
 legal
 machinery
 memory
 screenshots
 trans
 utils
 vcs
 wladmin
 Django

Weblate frontend

The frontend is currently built using Bootstrap, jQuery and few third party libraries.

Dependency management

The yarn package manager is used to update third party libraries. The configuration lives in `scripts/yarn` and there is a wrapper script `scripts/yarn-update` to upgrade the libraries, build them and copy to correct locations in `weblate/static/vendor`, where all third partly frontend code is located.

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.

Continuous integration

Current test results can be found on [GitHub Actions](#) and coverage is reported on [Codecov](#).

There are several jobs to verify different aspects:

Unit tests

Documentation build and external links

Migration testing from all supported releases

Code linting

Setup verification (ensures that generated dist files do not miss anything and can be tested)

The configuration for the CI is in `.github/workflows` directory. It heavily uses helper scripts stored in `ci` directory. The scripts can be also executed manually, but they require several environment variables, mostly defining Django settings file to use and database connection. The example definition of that is in `scripts/test-database`:

```
# Simple way to configure test database from environment

# Database backend to use postgresql / mysql / mariadb
export CI_DATABASE=${1:-postgresql}

# Database server configuration
export CI_DB_USER=weblate
export CI_DB_PASSWORD=weblate
export CI_DB_HOST=127.0.0.1

# Django settings module to use
export DJANGO_SETTINGS_MODULE=weblate.settings_test
```

The simple execution can look like:

```
. scripts/test-database
./ci/run-migrate
./ci/run-test
./ci/run-docs
./ci/run-setup
```

Local testing

To run a testsuite locally, use:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py test
```

!!!: You will need a database (PostgreSQL) server to be used for tests. By default Django creates separate database to run tests with `test_` prefix, so in case your settings is configured to use `weblate`, the tests will use `test_weblate` database. See [Database setup for Weblate](#) for setup instructions.

The `weblate/settings_test.py` is used in CI environment as well (see [Continuous integration](#)) and can be tuned using environment variables:

```
# Simple way to configure test database from environment

# Database backend to use postgresql / mysql / mariadb
export CI_DATABASE=${1:-postgresql}

# Database server configuration
export CI_DB_USER=weblate
export CI_DB_PASSWORD=weblate
```

(XXXXXXXXXXXX)

```
export CI_DB_HOST=127.0.0.1
# Django settings module to use
export DJANGO_SETTINGS_MODULE=weblate.settings_test
```

Prior to running tests you should collect static files as some tests rely on them being present:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py collectstatic
```

You can also specify individual tests to run:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py test weblate.
↳ gitexport
```

!!!: The tests can also be executed inside developer docker container, see [Docker!!! Weblate !!!](#).

!!:

See Testing in Django for more info on running and writing tests for Django.

!!! !!!

Weblate !! JSON Schema to !!!!!!! JSON !!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Weblate !!!!!!!

<https://weblate.org/schemas/weblate-memory.schema.json>

?	??	
??	!!!!!!!!!!!!	
	?	!!!!!!!!
	!!!!!!	
category	!!!!!!	1 = !!!!!!!2 = !!!!!10000000 ?? = !!!!!!!!!!!!20000000 ?? = !!!!!
	?	??
	?	1
	??	0
	!!!!!!	1
origin	!!!!!!!!!!!!	
	!!!!!!!!!!!!	
	?	!!!
	?	test
	!!!!	^(.*)\$
source	!!!!	
	??	
	?	!!!
	?	Hello
	!!!!	^(.*)\$
source_language	!!!!	
	!!!!	ISO 639-1 / ISO 639-2 / IETF BCP 47
	?	!!!
	?	en
	!!!!	^[^]+\$
target	!!!!	
	!!!!!!!!!!!!	
	?	!!!
	?	Ahoj
	!!!!	^(.*)\$
target_language	!!!!	
	!!!!	ISO 639-1 / ISO 639-2 / IETF BCP 47

!!!!!!!!!!!!

Table 4 – `weblate-userdata.schema.json`

	<code>username</code>	string	required	False	cs ^([\]+)\$
--	-----------------------	--------	----------	-------	------------------

dump_memory

`dump_memory` `import_memory`

Weblate `weblate-userdata.schema.json`

<https://weblate.org/schemas/weblate-userdata.schema.json>

	<code>basic</code>	object	required		
	<code>username</code>	string	required	False	cs ^([\]+)\$
	<code>full_name</code>	string	optional	False	
	<code>email</code>	string	optional	False	Weblate cs ^([\]+)\$
	<code>date_joined</code>	string	optional	False	noreply@example.com ^([\]+)\$
	<code>profile</code>	object	optional	False	
	<code>language</code>	string	optional	False	cs ^([\]+)\$
	<code>suggested</code>	boolean	optional	True	
	<code>translated</code>	integer	optional	0	
	<code>uploaded</code>	integer	optional	0	
	<code>hide_completed</code>	boolean	optional	False	boolean False True

`weblate-userdata.schema.json`

Table 5 – `zen`

<code>secondary_in_zen</code>	<code>Zen</code>	<code>boolean</code>	
	<code>?</code>	<code>True</code>	
	<code>?????</code>	<code>True</code>	
<code>hide_source_secondary</code>	<code>?????</code>	<code>boolean</code>	
	<code>?</code>	<code>False</code>	
	<code>?????</code>	<code>True</code>	
<code>editor_link</code>	<code>???? ???</code>	<code>???</code>	
	<code>?</code>		
	<code>?????</code>	<code>^.*\$</code>	
<code>translate_mode</code>	<code>?????</code>		
	<code>?????? ???</code>		
	<code>?</code>	<code>??</code>	
	<code>?</code>	<code>0</code>	
	<code>?????</code>	<code>0</code>	
<code>zen_mode</code>	<code>Zen</code>	<code>??</code>	
	<code>?</code>	<code>0</code>	
	<code>?????</code>	<code>0</code>	
<code>special_chars</code>	<code>?????</code>	<code>???</code>	
	<code>?</code>		
	<code>?????</code>	<code>^.*\$</code>	
	<code>??????</code>		
<code>dashboard_view</code>	<code>????????????????????</code>		
	<code>?</code>	<code>??</code>	
	<code>?</code>	<code>1</code>	
	<code>?????</code>	<code>0</code>	
	<code>???????????????????? ???</code>		
<code>dashboard_component</code>	<code>???</code>	<code>??</code>	
	<code>?????</code>	<code>?</code>	<code>null</code>
		<code>?</code>	<code>??</code>
<code>languages</code>	<code>?????</code>		
	<code>?</code>	<code>??</code>	
	<code>?????</code>		
	<code>??</code>	<code>??????</code>	
		<code>?</code>	<code>???</code>
		<code>?</code>	<code>cs</code>
		<code>?????</code>	<code>^.*\$</code>
		<code>??????</code>	
<code>secondary_languages</code>	<code>?????</code>		
	<code>?</code>	<code>??</code>	
	<code>?????</code>		
	<code>??</code>	<code>??????</code>	
		<code>?</code>	<code>???</code>
		<code>?</code>	<code>sk</code>
		<code>?????</code>	<code>^.*\$</code>
		<code>??????</code>	
<code>watched</code>	<code>????????????????</code>		
	<code>?</code>	<code>??</code>	
	<code>??????</code>		
	<code>??</code>	<code>????????????????</code>	
		<code>?</code>	<code>???</code>
		<code>?</code>	<code>weblate</code>
		<code>?????</code>	<code>^.*\$</code>
		<code>??????</code>	
<code>auditlog</code>	<code>?????</code>		
	<code>?</code>	<code>??</code>	
	<code>??????</code>		
	<code>??</code>	<code>??</code>	
	<code>?</code>	<code>????????</code>	
	<code>??????</code>		
	<code>address</code>	<code>IP</code>	<code>?????</code>

Table 5 – `dumpuserdata`

Field	Example
<code>user_agent</code>	127.0.0.1 ^.*\$
<code>timestamp</code>	PC / Linux / Firefox 70.0 ^.*\$
<code>activity</code>	2019-11- 18T18:58:30.845Z ^.*\$

`dumpuserdata`

`dumpuserdata`:

`dumpuserdata dumpuserdata`

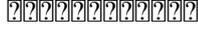
Weblate `dumpuserdata`

`dumpuserdata`:

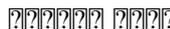
1. `./scripts/list-translated-languages`
 2. `./scripts/prepare-release`
 3. `make -C docs update-screenshots`
 4. `./scripts/create-release --tag`
 5. Docker
 6. GitHub
 7. Docker
 8. Helm
 9. `github/workflows/migrations.yml`
 10. `./scripts/set-version`
- `./scripts/create-release` GnuPG
- Weblate git push push
- hub Weblate
- Weblate SSH Web

Weblate

Web-based continuous localization tool with tight  supporting a wide range of *Supported file formats*, making it easy for translators to contribute.

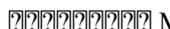
"Weblate"  "web"  "translate" 

Web

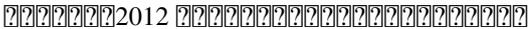
  <https://weblate.org/>    <https://hosted.weblate.org/>  <https://docs.weblate.org/> 

 <https://github.com/WeblateOrg/graphics/> 

Leadership

 Michal Čihař michal@cihar.com 

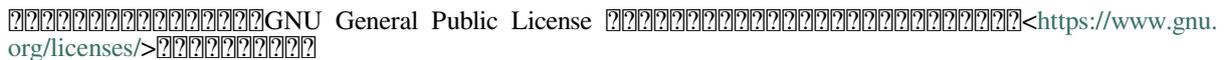
Authors

Weblate  Michal Čihař michal@cihar.com 

Copyright (C) 2012 - 2020 Michal Čihař michal@cihar.com

:  FreeSoftware Foundation  GNU General Public License  3 

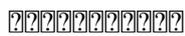
 GNU General Public License 

 GNU General Public License  <https://www.gnu.org/licenses/> 



Weblate 4.3.1

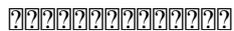
Released on October 21st 2020.



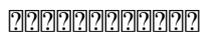
Fixed session expiry for authenticated users.

Add support for hiding version information.

Improve hooks compatibility with Bitbucket Server.



Reduced memory usage.



Added confirmation before removing user from a project.

Weblate 4.3

Released on October 15th 2020.

Include user stats in the API.

Fixed component ordering on paginated pages.

Define source language for a glossary.

Rewritten support for GitHub and GitLab pull requests.

????????????????????????????????

????????????????????????

Fixed configuration of enforced checks.

Improve documentation about built-in backups.

Moved source language attribute from project to a component.

Vue I18n ??????????????

Generic placeholders check now supports regular expressions.

Improved look of matrix mode.

????????????????????????????

Added support for interacting with multiple GitLab or GitHub instances.

Extended API to cover project updates, unit updates and removals and glossaries.

Unit API now properly handles plural strings.

Component creation can now handle ZIP file or document upload.

Consolidated API response status codes.

????????????? markdown ??????????

????????????????????

Improved JSON, YAML and CSV formats compatibility.

????????????????????

Improved performance of file downloads.

Improved repository management view.

Automatically enable java-format for Android.

????????????????????????????????

Python 3.9 ?????

Fixed translating HTML files under certain conditions.

Weblate 4.2.2

Released on September 2nd 2020.

JSON ??????????????????????????

Fixed login redirect for some authentication configurations.

Fixed LDAP authentication with group sync.

????????????????????????????????

Fixed Git commit squashing with trailers enabled.

Fixed creating local VCS components using API.

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*.

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.

[gettext](#): [Laravel PHP](#) [HTML files](#) [OpenDocument Format](#) [IDML Format](#) [Windows RC files](#) [INI translations](#) [Inno Setup INI](#) [GWT properties](#) [go-i18n JSON files](#) [ARB File](#)

Consistently use dismissed as state of dismissed checks.

Add support for configuring default addons to enable.

Fixed editor keyboard shortcut to dismiss checks.

Improved machine translation of strings with placeholders.

Show ghost translation for user languages to ease starting them.

Improved language code parsing.

Show translations in user language first in the list.

Renamed shapings to more generic name variants.

Added new quality checks: [gettext](#), [gettext](#), [gettext](#).

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 [gettext](#).

Always show rendered text check if enabled.

API now supports filtering of changes.

Added support for sharing glossaries between projects.

Weblate 4.0.4

Released on May 07th 2020.

Fixed testsuite execution on some Python 3.8 environments.

Typo fixes in the documentation.

Fixed creating components using API in some cases.

Fixed JavaScript errors breaking mobile navigation.

Fixed crash on displaying some checks.

Fixed screenshots listing.

Fixed monthly digest notifications.

Fixed intermediate translation behavior with units non existing in translation.

Weblate 4.0.3

Released on May 02nd 2020.

Fixed possible crash in reports.

User mentions in comments are now case insensitive.

Fixed PostgreSQL migration for non superusers.

Fixed changing the repository URL while creating component.

Fixed crash when upstream repository is gone.

Weblate 4.0.2

Released on April 27th 2020.

Improved performance of translation stats.

Improved performance of changing labels.

Improved bulk edit performance.

Improved translation memory performance.

Fixed possible crash on component deletion.

Fixed displaying of translation changes in some corner cases.

Improved warning about too long celery queue.

Fixed possible false positives in the consistency check.

Fixed deadlock when changing linked component repository.

Included edit distance in changes listing and CSV and reports.

Avoid false positives of punctuation spacing check for Canadian French.

Fixed XLIFF export with placeholders.

Fixed false positive with zero width check.

Improved reporting of configuration errors.

Fixed bilingual source upload.

Automatically detect supported languages for DeepL machine translation.

Fixed progress bar display in some corner cases.

Fixed some checks triggering on non translated strings.

Weblate 4.0.1

Released on April 16th 2020.

Fixed package installation from PyPI.

Weblate 4.0

Released on April 16th 2020.

Weblate now requires Python 3.6 or newer.

Added management overview of component alerts.

Added component alert for broken repository browser URLs.

Improved sign in and registration pages.

Project access control and workflow configuration integrated to project settings.

Added check and highlighter for i18next interpolation and nesting.

Added check and highlighter for percent placeholders.

Record source string changes in history.

Record source string changes in history.

Upgraded Microsoft Translator to version 3 API.

Reimplemented translation memory backend.

Added support for several `is:` lookups in `is`.

Allow to make `is` avoid internal blacklist.

Improved comments extraction from monolingual po files.

Renamed whiteboard messages to announcements.

Fixed occasional problems with registration mails.

Improved LINGUAS update addon to handle more syntax variants.

Fixed editing monolingual XLIFF source file.

Added support for exact matching in `is`.

Added support for exact matching in `is` API.

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.

Weblate 3.x series

Weblate 3.11.3

Released on March 11th 2020.

Fixed searching for fields with certain priority.

Fixed predefined query for recently added strings.

Fixed searching returning duplicate matches.

Fixed notifications rendering in Gmail.

Fixed reverting changes from the history.

Added links to events in digest notifications.

Fixed email for account removal confirmation.

Added support for Slack authentication in Docker container.

Avoid sending notifications for not subscribed languages.

Include Celery queues in performance overview.

Fixed documentation links for addons.
????????????????????????????
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.

Weblate 3.11.2

Released on February 22nd 2020.
Fixed rendering of suggestions.
Fixed some strings wrongly reported as having no words.

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.

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.
????????????????????????????
Recommend upgrade to new Weblate versions in the system checks.
Provide more detailed analysis for duplicate language alert.
Include more detailed license info on the project pages.
Automatically unshallow local copies if needed.
Fixed download of strings needing action.
New alert to warn about using the same filemask twice.
Improve XML placeables extraction.
The *SINGLE_PROJECT* can now enforce redirection to chosen project.
Added option to resolve comments.
Added bulk editing of flags.
Added support for *String labels*.
Added bulk edit addon.
Added option for *??????*.

Increased default validity of confirmation links.
Improved Matomo integration.
Fixed `???` to correctly handle source string change.
Extended automatic updates configuration by `AUTO_UPDATE`.
LINGUAS addons now do full sync of translations in Weblate.

Weblate 3.10.3

Released on January 18th 2020.
Support for translate-toolkit 2.5.0.

Weblate 3.10.2

Released on January 18th 2020.
Add lock indication to projects.
Fixed CSS bug causing flickering in some web browsers.
Fixed searching on systems with non-English locales.
Improved repository matching for GitHub and Bitbucket hooks.
Fixed data migration on some Python 2.7 installations.
Allow configuration of Git shallow cloning.
Improved background notification processing.
Fixed broken form submission when navigating back in web browser.
New addon to configure YAML formatting.
Fixed same plurals check to not fire on single plural form languages.
Fixed regex search on some fields.

Weblate 3.10.1

Released on January 9th 2020.
Extended API with translation creation.
Fixed several corner cases in data migrations.
Compatibility with Django 3.0.
Improved data cleanup performance.
Added support for customizable security.txt.
Improved breadcrumbs in changelog.
Improved translations listing on dashboard.
Improved HTTP responses for webhooks.
Added support for GitLab merge requests in Docker container.

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.

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.

Weblate 3.9

Released on October 15th 2019.

- Include Weblate metadata in downloaded files.
- Improved UI for failing checks.
- Indicate missing strings in format checks.
- Separate check for French punctuation spacing.
- Add support for fixing some of quality checks errors.
- Add separate permission to create new projects.
- Extend stats for char counts.
- Improve support for Java style language codes.
- Added new generic check for placeholders.
- Added support for WebExtension JSON placeholders.
- Added support for flat XML format.
- Extended API with project, component and translation removal and creation.
- Added support for Gitea and Gitee webhooks.

Added new custom regex based check.
Allow to configure contributing to shared translation memory.
Added ZIP download for more translation files.
Make XLIFF standard compliant parsing of maxwidth and font.
Added new check and fixer for safe HTML markup for translating web applications.
Add component alert on unsupported configuration.
Added automatic translation addon to bootstrap translations.
Extend automatic translation to add suggestions.
Display addon parameters on overview.
Sentry is now supported through modern Sentry SDK instead of Raven.
Changed example settings to be better fit for production environment.
Added automated backups using BorgBackup.
Split cleanup addon for RESX to avoid unwanted file updates.
Added advanced search capabilities.
Allow users to download their own reports.
Added localization guide to help configuring components.
Added support for GitLab merge requests.
Improved display of repository status.
Perform automated translation in the background.

Weblate 3.8

Released on August 15th 2019.
Added support for simplified creating of similar components.
Added support for parsing translation flags from the XML based file formats.
Log exceptions into Celery log.
Improve performance of repository scoped addons.
Improved look of notification e-mails.
Fixed password reset behavior.
Improved performance on most of translation pages.
Fixed listing of languages not known to Weblate.
Add support for cloning addons to discovered components.
Add support for replacing file content with uploaded.
Add support for translating non VCS based content.
Added OpenGraph widget image to use on social networks.
Added support for animated screenshots.
Improved handling of monolingual XLIFF files.
Avoid sending multiple notifications for single event.
Add support for filtering changes.
Extended predefined periods for reporting.
Added webhook support for Azure Repos.
New opt-in notifications on pending suggestions or untranslated strings.
Add one click unsubscribe link to notification e-mails.
Fixed false positives with Has been translated check.
New management interface for admins.
String priority can now be specified using flags.
Added language management views.

Add checks for Qt library and Ruby format strings.
Added configuration to better fit single project installations.
Notify about new string on source string change on monolingual translations.
Added separate view for translation memory with search capability.

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.

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.
Refreshed landing page for new contributors.
Add support for configuring msgmerge addon.
Delay opening SMTP connection when sending notifications.
Improved error logging.
Allow custom location in MO generating addon.
Added addons to cleanup old suggestions or comments.
Added option to enable horizontal mode in the Zen editor.
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.

Weblate 3.6.1

Released on April 26th 2019.

Improved handling of monolingual XLIFF files.

Fixed digest notifications in some corner cases.

Fixed addon script error alert.

Fixed generating MO file for monolingual PO files.

Fixed display of uninstalled checks.

Indicate administered projects on project listing.

Allow update to recover from missing VCS repository.

Weblate 3.6

Released on April 20th 2019.

Add support for downloading user data.

Addons are now automatically triggered upon installation.

Improved instructions for resolving merge conflicts.

Cleanup addon is now compatible with app store metadata translations.

Configurable language code syntax when adding new translations.

Warn about using Python 2 with planned termination of support in April 2020.

Extract special characters from the source string for visual keyboard.

Extended contributor stats to reflect both source and target counts.

Admins and consistency addons can now add translations even if disabled for users.

Fixed description of toggle disabling `Language-Team` header manipulation.

Notify users mentioned in comments.

Removed file format autodetection from component setup.

Fixed generating MO file for monolingual PO files.

Added digest notifications.

Added support for muting component notifications.

Added notifications for new alerts, whiteboard messages or components.

Notifications for administered projects can now be configured.

Improved handling of three letter language codes.

Weblate 3.5.1

Released on March 10th 2019.

Fixed Celery systemd unit example.

Fixed notifications from HTTP repositories with login.

Fixed race condition in editing source string for monolingual translations.

Include output of failed addon execution in the logs.

Improved validation of choices for adding new language.

Allow to edit file format in component settings.

Update installation instructions to prefer Python 3.

Performance and consistency improvements for loading translations.

Make Microsoft Terminology service compatible with current Zeep releases.

Localization updates.

Weblate 3.5

Released on March 3rd 2019.

Improved performance of built-in translation memory.

Added interface to manage global translation memory.

Improved alerting on bad component state.

Added user interface to manage whiteboard messages.

Addon commit message now can be configured.

Reduce number of commits when updating upstream repository.

Fixed possible metadata loss when moving component between projects.

Improved navigation in the Zen mode.

Added several new quality checks (Markdown related and URL).

Added support for app store metadata files.

Added support for toggling GitHub or Gerrit integration.

Kashida ??????????

Added option to squash commits based on authors.

Improved support for XLSX file format.

Compatibility with Tesseract 4.0.

Billing addon now removes projects for unpaid billings after 45 days.

Weblate 3.4

Released on January 22nd 2019.

Added support for XLIFF placeholders.

Celery can now utilize multiple task queues.

Added support for renaming and moving projects and components.

Include characters counts in reports.

Added guided adding of translation components with automatic detection of translation files.

Customizable merge commit messages for Git.

Added visual indication of component alerts in navigation.

Improved performance of loading translation files.

New addon to squash commits prior to push.

Improved displaying of translation changes.

Changed default merge style to rebase and made that configurable.

Better handle private use subtags in language code.

Improved performance of fulltext index updates.

Extended file upload API to support more parameters.

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.

Weblate 3.2.2

Released on October 20th 2018.

Remove no longer needed Babel dependency.

Updated language definitions.

Improve documentation for addons, LDAP and Celery.

Fixed enabling new dos-eol and auto-java-messageformat flags.

Fixed running setup.py test from PyPI package.

Improved plurals handling.

Fixed translation upload API failure in some corner cases.

Fixed updating Git configuration in case it was changed manually.

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.

Weblate 3.2

Released on October 6th 2018.

Add install_addon management command for automated addon installation.

Allow more fine grained ratelimit settings.

Added support for export and import of Excel files.

Improve component cleanup in case of multiple component discovery addons.

Rewritten Microsoft Terminology machine translation backend.

Weblate now uses Celery to offload some processing.

Improved search capabilities and added regular expression search.

Added support for Youdao Zhiyun API machine translation.

Added support for Baidu API machine translation.

Integrated maintenance and cleanup tasks using Celery.

Improved performance of loading translations by almost 25%.

Removed support for merging headers on upload.

Removed support for custom commit messages.

Configurable editing mode (zen/full).

Added support for error reporting to Sentry.

Added support for automated daily update of repositories.

Added support for creating projects and components by users.

Built in translation memory now automatically stores translations done.

Users and projects can import their existing translation memories.

Better management of related strings for screenshots.

Added support for checking Java MessageFormat.
See [3.2 milestone on GitHub](#) for detailed list of addressed issues.

Weblate 3.1.1

Released on July 27th 2018.
Fix testsuite failure on some setups.

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.
██
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.

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.

Weblate 3.0

Released on June 1st 2018.
Rewritten access control.
Several code cleanups that lead to moved and renamed modules.
New addon for automatic component discovery.
The import_project management command has now slightly different parameters.
Added basic support for Windows RC files.
New addon to store contributor names in PO file headers.
The per component hook scripts are removed, use addons instead.
Add support for collecting contributor agreements.
Access control changes are now tracked in history.
New addon to ensure all components in a project have same translations.
Support for more variables in commit message templates.
Add support for providing additional textual context.

Weblate 2.x series

Weblate 2.20

Released on April 4th 2018.

Improved speed of cloning subversion repositories.

Changed repository locking to use third party library.

Added support for downloading only strings needing action.

Added support for searching in several languages at once.

New addon to configure gettext output wrapping.

New addon to configure JSON formatting.

Added support for authentication in API using RFC 6750 compatible Bearer authentication.

Added support for automatic translation using machine translation services.

Added support for HTML markup in whiteboard messages.

Added support for mass changing state of strings.

Translate-toolkit at least 2.3.0 is now required, older versions are no longer supported.

Added built in translation memory.

??

Added support for DeepL machine translation service.

Machine translation results are now cached inside Weblate.

??

Weblate 2.19.1

Released on February 20th 2018.

Fixed migration issue on upgrade from 2.18.

Improved file upload API validation.

Weblate 2.19

Released on February 15th 2018.

Fixed imports across some file formats.

Display human friendly browser information in audit log.

Added TMX exporter for files.

Various performance improvements for loading translation files.

Added option to disable access management in Weblate in favor of Django one.

Improved glossary lookup speed for large strings.

Compatibility with django_auth_ldap 1.3.0.

Configuration errors are now stored and reported persistently.

Honor ignore flags in whitespace autofixer.

Improved compatibility with some Subversion setups.

Improved built in machine translation service.

Added support for SAP Translation Hub service.

Added support for Microsoft Terminology service.

Removed support for advertisement in notification e-mails.

Improved translation progress reporting at language level.

Improved support for different plural formulas.

Added support for Subversion repositories not using stdlayout.

Added addons to customize translation workflows.

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.

Weblate 2.17.1

Released on October 13th 2017.

Fixed running testsuite in some specific situations.

Locales updates.

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.

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.

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.

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.

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.

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.

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.

?????? ???

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.

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.

Weblate 2.11

Released on Jan 31st 2017.

Include language detailed information on language page.

Mercurial backend improvements.

Added option to specify translation component priority.

More consistent usage of Group ACL even with less used permissions.

Added WL_BRANCH variable to hook scripts.

Improved developer documentation.

Better compatibility with various Git versions in Git exporter addon.

Included per project and component stats.

Added language code mapping for better support of Microsoft Translate API.

Moved fulltext cleanup to background job to make translation removal faster.

Fixed displaying of plural source for languages with single plural form.

Improved error handling in import_project.

Various performance improvements.

Weblate 2.10.1

Released on Jan 20th 2017.

Do not leak account existence on password reset form (CVE-2017-5537).

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.

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.
????????????????????????????????
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.

Weblate 2.8

Released on Aug 31st 2016.
Documentation improvements.
Translations.
Updated bundled javascript libraries.
Added list_translators management command.
Django 1.8 is no longer supported.
Fixed compatibility with Django 1.10.
Added Subversion support.
Separated XML validity check from XML mismatched tags.
Fixed API to honor HIDE_REPO_CREDENTIALS settings.
Show source change in Zen mode.
Alt+PageUp/PageDown/Home/End now works in Zen mode as well.
Add tooltip showing exact time of changes.
Add option to select filters and search from translation page.
Added UI for translation removal.
Improved behavior when inserting placeables.
Fixed auto locking issues in Zen mode.

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.

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.

Weblate 2.5

- Released on Mar 10th 2016.
- Fixed automatic translation for project owners.
- Improved performance of commit and push operations.
- New management command to add suggestions from command line.
- Added support for merging comments on file upload.
- Added support for some GNU extensions to C printf format.
- Documentation improvements.
- Added support for generating translator credits.
- Added support for generating contributor stats.
- Site wide search can search only in one language.
- Improve quality checks for Armenian.
- Support for starting translation components without existing translations.
- Support for adding new translations in Qt TS.
- Improved support for translating PHP files.
- Performance improvements for quality checks.
- Added option to specify source language.
- Improved support for XLIFF files.
- Extended list of options for import_project.
- Improved targeting for whiteboard messages.
- Support for automatic translation across projects.
- Optimized fulltext search index.
- Added management command for auto translation.
- Added placeables highlighting.
- Added keyboard shortcuts for placeables, checks and machine translations.
- Improved translation locking.
- Added quality check for AngularJS interpolation.
- Added extensive group based ACLs.
- Clarified terminology on strings needing review (formerly fuzzy).
- Clarified terminology on strings needing action and not translated strings.
- Support for Python 3.
- Dropped support for Django 1.7.
- Dropped dependency on msginit for creating new gettext PO files.
- Added configurable dashboard views.
- Improved notifications on parse errors.

Added option to import components with duplicate name to `import_project`.
Improved support for translating PHP files
Added XLIFF export for dictionary.
Added XLIFF and gettext PO export for all translations.
Documentation improvements.
Added support for configurable automatic group assignments.
Improved adding of new translations.

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.

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.

Weblate 2.2

Released on Feb 19th 2015.

- Performance improvements.
- Fulltext search on location and comments fields.
- New SVG/javascript based activity charts.
- Support for Django 1.8.
- Support for deleting comments.
- Added own SVG badge.
- Added support for Google Analytics.
- Improved handling of translation filenames.
- Added support for monolingual JSON translations.
- Record component locking in a history.
- Support for editing source (template) language for monolingual translations.
- Added basic support for Gerrit.

Weblate 2.1

Released on Dec 5th 2014.

- Added support for Mercurial repositories.
- Replaced Glyphicon font by Awesome.
- Added icons for social authentication services.
- Better consistency of button colors and icons.
- Documentation improvements.
- Various bugfixes.
- Automatic hiding of columns in translation listing for small screens.
- Changed configuration of filesystem paths.
- Improved SSH keys handling and storage.
- Improved repository locking.
- Customizable quality checks per source string.

Allow to hide completed translations from dashboard.

Weblate 2.0

Released on Nov 6th 2014.
New responsive UI using Bootstrap.
Rewritten VCS backend.
Documentation improvements.
Added whiteboard for site wide messages.
Configurable strings priority.
Added support for JSON file format.
Fixed generating mo files in certain cases.
Added support for GitLab notifications.
Added support for disabling translation suggestions.
Django 1.7 support.
ACL projects now have user management.
Extended search possibilities.
Give more hints to translators about plurals.
Fixed Git repository locking.
Compatibility with older Git versions.
Improved ACL support.
Added buttons for per language quotes and other special characters.
Support for exporting stats as JSONP.

Weblate 1.x series

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.

Weblate 1.8

Released on November 7th 2013.
Please check manual for upgrade instructions.
Nicer listing of project summary.
Better visible options for sharing.
More control over anonymous users privileges.
Supports login using third party services, check manual for more details.
Users can login by e-mail instead of username.

Documentation improvements.
Improved source strings review.
Searching across all strings.
Better tracking of source strings.
Captcha protection for registration.

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.

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.

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 libavatar.

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.

Weblate 1.4

Released on January 23rd 2013.

Fixed deleting of checks/comments on string deletion.

Added option to disable automatic propagation of translations.

Added option to subscribe for merge failures.

Correctly import on projects which needs custom ttkit loader.

Added sitemaps to allow easier access by crawlers.

Provide direct links to string in notification e-mails or feeds.

Various improvements to admin interface.

Provide hints for production setup in admin interface.

Added per language widgets and engage page.

Improved translation locking handling.

Show code snippets for widgets in more variants.

Indicate failing checks or fuzzy strings in progress bars.

More options for formatting commit message.

Fixed error handling with machine translation services.

Improved automatic translation locking behaviour.

Support for showing changes from previous source string.

Added support for substring search.

Various quality checks improvements.

Support for per project ACL.

Basic string tests coverage.

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.

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.

????????????????????????????????????

Improved handling of not known languages when importing project.

Support for locking translation by translator.

Optionally maintain Language-Team header in po file.

Include some statistics in about page.

Supports (and requires) django-registration 0.8.

Caching of counted strings with failing checks.

Checking of requirements during setup.

Documentation improvements.

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.

Weblate 1.0

Released on May 10th 2012.
Improved validation while adding/saving component.
Experimental support for Android component files (needs patched tkit).
Updates from hooks are run in background.
Improved installation instructions.
Improved navigation in dictionary.

Weblate 0.x series

Weblate 0.9

Released on April 18th 2012.
Fixed import of unknown languages.
Improved listing of nearby messages.
Improved several checks.
Documentation updates.
Added definition for several more languages.
Various code cleanups.
Documentation improvements.
Changed file layout.
Update helper scripts to Django 1.4.
Improved navigation while translating.
Better handling of po file renames.
Better validation while creating component.
Integrated full setup into syncdb.
Added list of recent changes to all translation pages.
Check for not translated strings ignores format string only messages.

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.

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.

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.

Weblate 0.5

Released on February 12th 2012.

ng 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).

Weblate 0.4

Released on February 8th 2012.
Added usage guide to documentation.
Fixed API hooks not to require CSRF protection.

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.

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.

Weblate 0.1

Released on February 6th 2012.
Initial release.

W

wlc, 125

wlc.config, 126

wlc.main, 126

/

ANY /, 86 /api

GET /api/, 88 /api/changes

GET /api/changes/, 112

GET /api/changes/(int:id)/, 112 /api/component-lists

GET /api/component-lists/, 114

GET /api/component-lists/(str:slug)/, 114

POST /api/component-lists/(str:slug)/components/, 115

PUT /api/component-lists/(str:slug)/, 115

DELETE /api/component-lists/(str:slug)/, 115

DELETE /api/component-lists/(str:slug)/components/(str:component_slug), 115

PATCH /api/component-lists/(str:slug)/, 115 /api/components

GET /api/components/, 100

GET /api/components/(string:project)/(string:component)/, 100

GET /api/components/(string:project)/(string:component)/changes/, 103

GET /api/components/(string:project)/(string:component)/lock/, 103

GET /api/components/(string:project)/(string:component)/monolingual_base/, 105

GET /api/components/(string:project)/(string:component)/new_template/, 105

GET /api/components/(string:project)/(string:component)/repository/, 104

GET /api/components/(string:project)/(string:component)/screenshots/, 103

GET /api/components/(string:project)/(string:component)/statistics/, 106

GET /api/components/(string:project)/(string:component)/translations/, 105

POST /api/components/(string:project)/(string:component)/lock/, 103

POST /api/components/(string:project)/(string:component)/repository/, 104

POST /api/components/(string:project)/(string:component)/translations/, 105

PUT /api/components/(string:project)/(string:component)/, 102

DELETE /api/components/(string:project)/(string:component)/, 103

PATCH /api/components/(string:project)/(string:component)/, 101 /api/glossary

GET /api/glossary/, 115

GET /api/glossary/(int:id)/, 115

GET /api/glossary/(int:id)/projects/, 117

GET /api/glossary/(int:id)/terms/, 117

GET /api/glossary/(int:id)/terms/(int:term_id)/, 117

POST /api/glossary/(int:id)/projects/, 117

POST /api/glossary/(int:id)/terms/, 117

PUT /api/glossary/(int:id)/, 117

PUT /api/glossary/(int:id)/terms/(int:term_id)/, 118

DELETE /api/glossary/(int:id)/, 117

DELETE /api/glossary/(int:id)/projects/, 117

DELETE /api/glossary/(int:id)/terms/(int:term_id)/, 118

PATCH /api/glossary/(int:id)/, 117

PATCH /api/glossary/(int:id)/terms/(int:term_id)/, 118 /api/groups

GET /api/groups/, 91

GET /api/groups/(int:id)/, 91

POST /api/groups/, 91

POST /api/groups/(int:id)/componentlists/, 92
 POST /api/groups/(int:id)/components/, 92
 POST /api/groups/(int:id)/languages/, 92
 POST /api/groups/(int:id)/projects/, 92
 POST /api/groups/(int:id)/roles/, 92
 PUT /api/groups/(int:id)/, 91
 DELETE /api/groups/(int:id)/, 92
 DELETE /api/groups/(int:id)/componentlists/(int:component_list_id), 93
 DELETE /api/groups/(int:id)/components/(int:component_id), 92
 DELETE /api/groups/(int:id)/languages/(string:language_code), 92
 DELETE /api/groups/(int:id)/projects/(int:project_id), 92
 PATCH /api/groups/(int:id)/, 92 /api/languages
 GET /api/languages/, 94
 GET /api/languages/(string:language)/, 94
 GET /api/languages/(string:language)/statistics/, 95
 POST /api/languages/, 94
 PUT /api/languages/(string:language)/, 94
 DELETE /api/languages/(string:language)/, 95
 PATCH /api/languages/(string:language)/, 94 /api/projects
 GET /api/projects/, 95
 GET /api/projects/(string:project)/, 95
 GET /api/projects/(string:project)/changes/, 96
 GET /api/projects/(string:project)/components/, 97
 GET /api/projects/(string:project)/languages/, 99
 GET /api/projects/(string:project)/repository/, 96
 GET /api/projects/(string:project)/statistics/, 99
 POST /api/projects/, 95
 POST /api/projects/(string:project)/components/, 97
 POST /api/projects/(string:project)/repository/, 97
 PUT /api/projects/(string:project)/, 96
 DELETE /api/projects/(string:project)/, 96
 PATCH /api/projects/(string:project)/, 96 /api/roles
 GET /api/roles/, 93
 GET /api/roles/(int:id)/, 93
 POST /api/roles/, 93
 PUT /api/roles/(int:id)/, 93
 DELETE /api/roles/(int:id)/, 93
 PATCH /api/roles/(int:id)/, 93 /api/screenshots
 GET /api/screenshots/, 113
 GET /api/screenshots/(int:id)/, 113
 GET /api/screenshots/(int:id)/file/, 113
 POST /api/screenshots/, 114
 POST /api/screenshots/(int:id)/file/, 113
 POST /api/screenshots/(int:id)/units/, 113
 PUT /api/screenshots/(int:id)/, 114
 DELETE /api/screenshots/(int:id)/, 114

DELETE /api/screenshots/(int:id)/units/(int:unit_id), 113

PATCH /api/screenshots/(int:id)/, 114 /api/translations

GET /api/translations/, 107

GET /api/translations/(string:project)/(string:component)/(string:language)/, 107

GET /api/translations/(string:project)/(string:component)/(string:language)/changes/, 109

GET /api/translations/(string:project)/(string:component)/(string:language)/file/, 109

GET /api/translations/(string:project)/(string:component)/(string:language)/repository/, 110

GET /api/translations/(string:project)/(string:component)/(string:language)/statistics/, 110

GET /api/translations/(string:project)/(string:component)/(string:language)/units/, 109

POST /api/translations/(string:project)/(string:component)/(string:language)/autotransla 109

POST /api/translations/(string:project)/(string:component)/(string:language)/file/, 110

POST /api/translations/(string:project)/(string:component)/(string:language)/repository/ 110

POST /api/translations/(string:project)/(string:component)/(string:language)/units/, 109

DELETE /api/translations/(string:project)/(string:component)/(string:language)/, 108 /api/units

GET /api/units/, 111

GET /api/units/(int:id)/, 111

PUT /api/units/(int:id)/, 112

DELETE /api/units/(int:id)/, 112

PATCH /api/units/(int:id)/, 112 /api/users

GET /api/users/, 88

GET /api/users/(str:username)/, 88

GET /api/users/(str:username)/notifications/, 90

GET /api/users/(str:username)/notifications/(int:subscription_id)/, 90

GET /api/users/(str:username)/statistics/, 89

POST /api/users/, 88

POST /api/users/(str:username)/groups/, 89

POST /api/users/(str:username)/notifications/, 90

PUT /api/users/(str:username)/, 89

PUT /api/users/(str:username)/notifications/(int:subscription_id)/, 90

DELETE /api/users/(str:username)/, 89

DELETE /api/users/(str:username)/notifications/(int:subscription_id)/, 90

PATCH /api/users/(str:username)/, 89

PATCH /api/users/(str:username)/notifications/(int:subscription_id)/, 90 /ex-ports

GET /exports/rss/, 121

GET /exports/rss/(string:project)/, 121

GET /exports/rss/(string:project)/(string:component)/, 121

GET /exports/rss/(string:project)/(string:component)/(string:language)/, 121

GET /exports/rss/language/(string:language)/, 121

GET /exports/stats/(string:project)/(string:component)/, 120 /hooks
GET /hooks/update/(string:project)/, 118
GET /hooks/update/(string:project)/(string:component)/, 118
POST /hooks/azure/, 119
POST /hooks/bitbucket/, 119
POST /hooks/gitea/, 119
POST /hooks/gitee/, 119
POST /hooks/github/, 118
POST /hooks/gitlab/, 118
POST /hooks/pagure/, 119

```

.XML resource file file format, 75
--add auto_translate [options], 287
--addon ADDON install_addon [options], 292
--age HOURS commit_pending [options], 288
--author USER@EXAMPLE.COM add_suggestions [options], 287
--base-file-template TEMPLATE import_project [options], 291
--check importusers [options], 292
--config PATH wlc [options], 122
--config-section SECTION wlc [options], 122
--configuration CONFIG install_addon [options], 292
--convert wlc [options], 123
--email USER@EXAMPLE.COM createadmin [options], 289
--file-format FORMAT import_project [options], 291
--force loadpo [options], 293
--force-commit pushgit [options], 294
--format {csv,json,text,html} wlc [options], 122
--ignore import_json [options], 289
--inconsistent auto_translate [options], 287
--input wlc [options], 124
--key KEY wlc [options], 122
--lang LANGUAGE loadpo [options], 293
--language-code list_translators [options], 293
--language-map LANGMAP import_memory [options], 290
--language-regex REGEX import_project [options], 291
--license NAME import_project [options], 291
--license-url URL import_project [options], 291
--main-component import_project [options], 291
--main-component COMPONENT import_json [options], 289
--mt MT auto_translate [options], 287
--name createadmin [options], 289
--name-template TEMPLATE import_project [options], 290
--new-base-template TEMPLATE import_project [options], 291
--no-password createadmin [options], 289
--no-privs-update setupgroups [options], 294
--no-projects-update setupgroups [options], 294
--no-update setuplang [options], 295
--output wlc [options], 123
--overwrite auto_translate [options], 287 wlc [options], 124
--password PASSWORD createadmin [options], 289
--project PROJECT import_json [options], 289
--source PROJECT/COMPONENT auto_translate [options], 287
--threshold THRESHOLD auto_translate [options], 287
--update createadmin [options], 289 import_json [options], 290 in-
stall_addon [options], 292
--url URL wlc [options], 122
--user USERNAME auto_translate [options], 287
--username USERNAME createadmin [options], 289

```

--vcs NAME import_project `??????????????`, 291
 add_suggestions weblate admin command, 287
 add_suggestions `??????????????` --author USER@EXAMPLE.COM, 287
 ADMINS setting, 157
 AKISMET_API_KEY setting, 250
 ALLOWED_HOSTS setting, 157
 Android file format, 71
 ANONYMOUS_USER_NAME setting, 250
 API, 85, 121, 125
 Apple strings file format, 72
 ARB file format, 74
 AUDITLOG_EXPIRY setting, 250
 AUTH_LOCK_ATTEMPTS setting, 251
 AUTH_TOKEN_VALID setting, 251
 auto_translate weblate admin command, 287
 auto_translate `??????????????` --add, 287 --inconsistent, 287 --mt MT, 287 --overwrite, 287 --source PROJECT/COMPONENT, 287 --threshold THRESHOLD, 287 --user USERNAME, 287
 AUTO_UPDATE setting, 251
 AUTOFIX_LIST setting, 252
 AVATAR_URL_PREFIX setting, 251
 BASE_DIR setting, 252
 bilingual translation, 65
 celery_queues weblate admin command, 288
 changes wlc `??????????????`, 123
 CHECK_LIST setting, 252
 checkgit weblate admin command, 288
 cleanup wlc `??????????????`, 123
 cleanuptrans weblate admin command, 288
 Comma separated values file format, 75
 Command (*wlc.main* `????`), 126
 COMMENT_CLEANUP_DAYS setting, 253
 commit wlc `??????????????`, 123
 commit_pending weblate admin command, 288
 commit_pending `??????????????` --age HOURS, 288
 COMMIT_PENDING_HOURS setting, 253
 commitgit weblate admin command, 288
 createadmin weblate admin command, 289
 createadmin `??????????????` --email USER@EXAMPLE.COM, 289 --name, 289 --no-password, 289 --password PASSWORD, 289 --update, 289 --username USERNAME, 289
 CSP_CONNECT_SRC setting, 252
 CSP_FONT_SRC setting, 252
 CSP_IMG_SRC setting, 252
 CSP_SCRIPT_SRC setting, 252
 CSP_STYLE_SRC setting, 252
 CSV file format, 75
 DATA_DIR setting, 253
 DATABASE_BACKUP setting, 254

DATABASES setting, 157

DEBUG setting, 158

DEFAULT_ACCESS_CONTROL setting, 254

DEFAULT_ADD_MESSAGE setting, 254

DEFAULT_ADDON_MESSAGE setting, 254

DEFAULT_ADDONS setting, 254

DEFAULT_COMMIT_MESSAGE setting, 254

DEFAULT_COMMITTER_EMAIL setting, 255

DEFAULT_COMMITTER_NAME setting, 255

DEFAULT_DELETE_MESSAGE setting, 254

DEFAULT_FROM_EMAIL setting, 158

DEFAULT_MERGE_MESSAGE setting, 254

DEFAULT_MERGE_STYLE setting, 255

DEFAULT_PULL_MESSAGE setting, 256

DEFAULT_RESTRICTED_COMPONENT setting, 254

DEFAULT_TRANSLATION_PROPAGATION setting, 255

download wlc [REDACTED], 123

DTD file format, 77

dump_memory weblate admin command, 289

dumpuserdata weblate admin command, 289

ENABLE_AVATARS setting, 256

ENABLE_HOOKS setting, 256

ENABLE_HTTPS setting, 256

ENABLE_SHARING setting, 256

file format .XML resource file, 75 Android, 71 Apple strings, 72 ARB, 74 Comma separated values, 75 CSV, 75 DTD, 77 gettext, 66 go-i18n, 74 GWT properties, 69 i18next, 73 INI translations, 70 Java properties, 69 Joomla translations, 70 JSON, 72 PHP strings, 72 PO, 66 Qt, 71 RC, 77 RESX, 75 Ruby YAML, 76 Ruby YAML Ain't Markup Language, 76 string resources, 71 TS, 71 XLIFF, 68 XML, 77 YAML, 76 YAML Ain't Markup Language, 76

get () (wlc. Weblate [REDACTED]), 125

gettext file format, 66

GITHUB_CREDENTIALS setting, 257

GITHUB_TOKEN setting, 257

GITHUB_USERNAME setting, 257

GITLAB_CREDENTIALS setting, 256

GITLAB_TOKEN setting, 257

GITLAB_USERNAME setting, 257

go-i18n file format, 74

GOOGLE_ANALYTICS_ID setting, 257

GWT properties file format, 69

HIDE_REPO_CREDENTIALS setting, 258

HIDE_VERSION setting, 258

i18next file format, 73

import_demo weblate admin command, 289

import_json weblate admin command, 289

import_json [REDACTED] --ignore, 289 --main-component COMPONENT, 289 --project PROJECT, 289 --update, 290

import_memory weblate admin command, 290

import_memory `???????????????` --language-map LANGMAP, 290
import_project weblate admin command, 290
import_project `?????????????????` --base-file-template TEMPLATE, 291 --file-format FORMAT, 291 --language-regex REGEX, 291 --license NAME, 291 --license-url URL, 291 --main-component, 291 --name-template TEMPLATE, 290 --new-base-template TEMPLATE, 291 --vcs NAME, 291
importuserdata weblate admin command, 292
importusers weblate admin command, 292
importusers `?????????????????` --check, 292
INI translations file format, 70
install_addon weblate admin command, 292
install_addon `?????????????????` --addon ADDON, 292 --configuration CONFIG, 292 --update, 292
IP_BEHIND_REVERSE_PROXY setting, 258
IP_PROXY_HEADER setting, 258
IP_PROXY_OFFSET setting, 258
iPad translation, 72
iPhone translation, 72
Java properties file format, 69
Joomla translations file format, 70
JSON file format, 72
LEGAL_URL setting, 259
LICENSE_EXTRA setting, 259
LICENSE_FILTER setting, 259
LICENSE_REQUIRED setting, 260
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH setting, 260
list_languages weblate admin command, 293
list_translators weblate admin command, 293
list_translators `?????????????????` --language-code, 293
list_versions weblate admin command, 293
list-components wlc `?????????????????`, 123
list-languages wlc `?????????????????`, 123
list-projects wlc `?????????????????`, 123
list-translations wlc `?????????????????`, 123
load() (*wlc.config.WeblateConfig* `????????`), 126
loadpo weblate admin command, 293
loadpo `?????????????????` --force, 293 --lang LANGUAGE, 293
LOCALIZE_CDN_PATH setting, 260
LOCALIZE_CDN_URL setting, 260
lock wlc `?????????????????`, 123
lock_translation weblate admin command, 293
lock-status wlc `?????????????????`, 123
LOGIN_REQUIRED_URLS setting, 260
LOGIN_REQUIRED_URLS_EXCEPTIONS setting, 260
ls wlc `?????????????????`, 123
MACHINE_TRANSLATION_SERVICES setting, 261
main() (*wlc.main* `????????`), 126
MATOMO_SITE_ID setting, 261

MATOMO_URL setting, 261

monolingual translation, 65

move_language weblate admin command, 294

MT_APERTIUM_APY setting, 262

MT_AWS_ACCESS_KEY_ID setting, 262

MT_AWS_REGION setting, 262

MT_AWS_SECRET_ACCESS_KEY setting, 262

MT_BAIDU_ID setting, 262

MT_BAIDU_SECRET setting, 262

MT_DEEPL_API_VERSION setting, 262

MT_DEEPL_KEY setting, 263

MT_GOOGLE_CREDENTIALS setting, 263

MT_GOOGLE_KEY setting, 263

MT_GOOGLE_LOCATION setting, 263

MT_GOOGLE_PROJECT setting, 263

MT_MICROSOFT_BASE_URL setting, 263

MT_MICROSOFT_COGNITIVE_KEY setting, 264

MT_MICROSOFT_ENDPOINT_URL setting, 264

MT_MICROSOFT_REGION setting, 264

MT_MODERNMT_KEY setting, 264

MT_MODERNMT_URL setting, 264

MT_MYMEMORY_EMAIL setting, 264

MT_MYMEMORY_KEY setting, 264

MT_MYMEMORY_USER setting, 264

MT_NETEASE_KEY setting, 265

MT_NETEASE_SECRET setting, 265

MT_SAP_BASE_URL setting, 265

MT_SAP_PASSWORD setting, 266

MT_SAP_SANDBOX_APIKEY setting, 266

MT_SAP_USE_MT setting, 266

MT_SAP_USERNAME setting, 266

MT_SERVICES setting, 261

MT_TMSERVER setting, 265

MT_YANDEX_KEY setting, 265

MT_YOUDAO_ID setting, 265

MT_YOUDAO_SECRET setting, 265

NEARBY_MESSAGES setting, 266

PHP strings file format, 72

PIWIK_SITE_ID setting, 261

PIWIK_URL setting, 261

PO file format, 66

post () (*wlc. Weblate* [\[?\]](#)), 125

pull wlc [\[?\]](#), 123

push wlc [\[?\]](#), 123

pushgit weblate admin command, 294

pushgit [\[?\]](#) --force-commit, 294

Python, 125