
Weblate Documentation

Выпуск 2.4

Michal Čihař

июл. 20, 2020

1	About Weblate	3
1.1	Project goals	3
1.2	Project name	3
1.3	Project website	3
1.4	Authors	3
2	Translators guide	5
2.1	Weblate basics	5
2.2	Registration and user profile	5
2.3	Translating using Weblate	9
2.4	Checks and fixups	16
3	Application developer guide	21
3.1	Activity reports	21
3.2	Promoting the translation	22
3.3	Reviewing source strings	23
3.4	Adding new translations	24
4	Administrators guide	25
4.1	Quick setup guide	25
4.2	Installation instructions	26
4.3	Weblate deployments	40
4.4	Upgrading Weblate	45
4.5	Authentication	49
4.6	Access control	52
4.7	Translation projects	56
4.8	Continuous translation	66
4.9	Translation process	73
4.10	Checks and fixups	74
4.11	Machine translation	77
4.12	Configuration	81
4.13	Sample configuration	90
4.14	Management commands	100
4.15	Advertisement	106
5	Frequently Asked Questions	107
5.1	Configuration	107

5.2	Usage	109
5.3	Troubleshooting	111
5.4	Features	112
6	Supported formats	115
6.1	Automatic detection	115
6.2	GNU Gettext	115
6.3	XLIFF	116
6.4	Java properties	117
6.5	Qt Linguist .ts	117
6.6	Android string resources	117
6.7	Apple OS X strings	118
6.8	PHP strings	118
6.9	JSON files	119
6.10	.Net Resource files	119
6.11	CSV files	119
6.12	Others	119
7	Weblate's Web API	121
7.1	Notification hooks	121
7.2	Exports	122
7.3	RSS feeds	124
8	Changes	125
8.1	weblate 2.4	125
8.2	weblate 2.3	126
8.3	weblate 2.2	126
8.4	weblate 2.1	127
8.5	weblate 2.0	127
8.6	weblate 1.9	128
8.7	weblate 1.8	128
8.8	weblate 1.7	129
8.9	weblate 1.6	129
8.10	weblate 1.5	130
8.11	weblate 1.4	130
8.12	weblate 1.3	131
8.13	weblate 1.2	131
8.14	weblate 1.1	132
8.15	weblate 1.0	132
8.16	weblate 0.9	133
8.17	weblate 0.8	133
8.18	weblate 0.7	134
8.19	weblate 0.6	134
8.20	weblate 0.5	134
8.21	weblate 0.4	135
8.22	weblate 0.3	135
8.23	weblate 0.2	135
8.24	weblate 0.1	135
9	Contributing	137
9.1	Code and development	137
9.2	Starting with our codebase	137
9.3	Earning money by coding	137
9.4	Translating	138

10 License	139
11 Indices and tables	141
HTTP Routing Table	143
Алфавитный указатель	145

Contents:

1.1 Project goals

Web based translation with tight git integration supporting wide range of file formats and makes it easy for translators to contribute.

The translations should be kept within same repository as source code and translation process should closely follow development.

There is no plan in heavy conflict resolution as these should be primarily handled on git side.

1.2 Project name

The project is named as mixture of words web and translate.

1.3 Project website

You can find project website at <http://weblate.org/>, there is also demonstration server at <https://demo.weblate.org/>. This documentation can be browsed on <http://docs.weblate.org/>.

1.4 Authors

This tool was written by Michal Čihar michal@cihar.com.

2.1 Weblate basics

2.1.1 Projects structure

Each project can contain various components. The reason for this structure is that all components in a project are expected to have a lot in common. Whenever translation is made in single component, it is automatically propagated to others within same project (this is especially useful when translating more version of same project, but can be disabled, see *Component configuration*).

2.2 Registration and user profile

2.2.1 Registration

While everybody can browse projects, view translations or suggest them, only registered users are allowed to actually save changes and are credited for every translation made.

You can register following two simple steps:

1. Fill out the registration form with your credentials
2. Activate registration by following in email you receive
3. Possibly adjust your profile to choose which languages you know

2.2.2 User profile

User profile contains your preferences, name and email. Name and email are being used in VCS commits, so keep this information accurate.

Translated languages

Choose here which languages you prefer to translate. These will be offered to you on main page to have easier access to translations.

Hosted Weblate
 Dashboard
 Your translations ▾
 Projects ▾
 Documentation

Michal Čihař

Dashboard

Your translations
 Projects
 Search
 History
 Activity
 Statistics
 Tools ▾

Your translations

Project ▾	Translated ▾	Words ▾	Fuzzy ▾	Checks ▾	Suggestions ▾		
CopyQ/master (Czech)	<div></div>	100.0%	100.0%	0.0%	0	0	
Gammu/gammu (Czech)	<div></div>	100.0%	100.0%	0.0%	0	0	
Gammu/lbgammu (Czech)	<div></div>	100.0%	100.0%	0.0%	0	0	
Gammu/wammu (Czech)	<div></div>	100.0%	100.0%	0.0%	0	0	
Gammu/wammu-doc (Czech)	<div></div>	100.0%	100.0%	0.0%	0	0	
Gammu/website (Czech)	<div></div>	83.4%	52.9%	0.0%	0	0	
phpMyAdmin/4.2 (Czech)	<div></div>	100.0%	100.0%	0.0%	0	0	
phpMyAdmin/master (Czech)	<div></div>	93.7%	92.2%	3.1%	1	0	
Ukolovnlk/master (Czech)	<div></div>	100.0%	100.0%	0.0%	0	0	
Weblate/javascript (Czech)	<div></div>	100.0%	100.0%	0.0%	0	0	
Weblate/master (Czech)	<div></div>	97.6%	94.7%	1.6%	1	0	
Weblate/website (Czech)	<div></div>	100.0%	100.0%	0.0%	0	0	
Website/master (Czech)	<div></div>	100.0%	100.0%	0.0%	0	0	
WinCompose/master (Czech)	<div></div>	65.4%	61.4%	3.6%	0	0	

- Good translations
 - Translations with failing checks
 - Fuzzy translations

Manage your translations

Powered by Weblate 2.0-dev
 About Weblate
 Contact us
 Documentation
 Donate to Weblate!

Secondary languages

You can define secondary languages, which will be shown you while translating together with source language. Example can be seen on following image, where Slovak language is shown as secondary:

Hosted Weblate
 Dashboard
 Your translations ▾
 Projects ▾
 Documentation
 Michal Čihař

Gammu / website / Czech / translate

⏪

⏴

Untranslated strings (1 / 81)

⏵

⏩

Zen

Translate

Slovak

↵

Marcin pôvodne začal projekt ako fork Gnokii pretože nebol spokojný
 s niektorými časťami existujúceho kódu. prispieval väčšinou do hlavného kódu a
 Nokia modulu a viedol projekt do januára 2007, keď rezignoval
 pre nedostatok času. Už nieje aktívny na projekte.

Source

↵

Marcin has originally started the project as fork of Gnokii because he was not
 happy with some parts of the existing code. He did contribute most of core and
 Nokia modules and he did lead the project till January 2007, when he resigned
 because lack of time. He is not active in the project anymore.

Translation

↵ Copy

→

↵

...

☐ Fuzzy

Save

Suggest

Commit message: Additional text to include in t

Glossary

No related strings were found in the glossary.

Manage glossary

Source information

Source string location

html/authors.html:41

Source string age

3 weeks ago

String priority

Medium

Nearby messages

Comments

Machine translation

Search

History

History

When	User	Action	Translation
No recent activity has been recorded.			

Browse changes

Follow using RSS

Powered by Weblate 2.0-dev

About Weblate

Contact us

Documentation

Donate to Weblate!

Subscriptions

You can subscribe to various notifications on *Subscriptions* tab. You will receive notifications for selected events on chosen projects for languages you have indicated for translation (see above).

If you are an owner of some project, you will always receive some important notifications, like merge failures or new language requests.

Примечание: You will not receive notifications for actions you've done.

2.2. Registration and user profile

7

W

Hosted Weblate

Dashboard

Your translations ▾

Projects ▾

Documentation

Michal Čihař

Your profile

Preferences

Subscriptions

Account

Authentication

Profile

Licenses

Subscribed projects

☐ ARandR

☐ Binary

☐ Bitcoin.cz

☐ BridgeClock

☐ CopyQ

☐ Debian Handbook

☐ django-geoip-redis

☐ Freeplane

☒ Gammu

☐ GePeS

☐ GetBack GPS

☐ Koha

☐ Monkeysphere

☐ MusicBottle

☐ OsmAnd

☒ phpMyAdmin

☐ Scarry On Lin{e}ux³

☐ SDAPS

☐ Spiel

☐ Summoning Wars

☒ Ukolovnik

☐ Unknown Horizons

☒ Weblate

☒ Website

☐ WinCompose

☐ WordPoints

Save

Description

You will receive chosen notifications via email for all your languages.

Documentation

Subscription settings

☐ Notification on any translation

☐ Notification on new string to translate

☐ Notification on new suggestion

☐ Notification on new contributor

☐ Notification on new comment

☐ Notification on merge failure




☐ Notification on new language request

Save

Powered by Weblate 2.0-dev About Weblate Contact us Documentation Donate to Weblate!

Authentication

On the *Authentication* tab you can connect various services which you can use to login into Weblate. List of services depends on Weblate configuration, but can include popular sites such as Google, Facebook, GitHub or Bitbucket.

 Hosted Weblate
 Dashboard
 Your translations ▾
 Projects ▾
 Documentation
  Michal Čihář
 

Your profile

Preferences
 Subscriptions
 Account
 Authentication
 Profile
 Licenses

User identities

You can manage identities which are associated to this account and which can be used to log in.

Currently associated:

Identity	User ID	Action
Password	nijel	<button>Change password</button>
GitHub	212189	<button>Disconnect</button>
Email	michal@cihar.com	<button>Disconnect</button>
Google	michal@cihar.com	<button>Disconnect</button>

Add new association:

Email

Description

You can configure how you will log in on this site.

Documentation

Removal

Removal of the account deletes all your private data.

Remove my account

Powered by Weblate 2.0-dev
 About Weblate
 Contact us
 Documentation
 Donate to Weblate!

Avatar

Weblate can be configured to show avatar for each user (depending on `ENABLE_AVATARS`). These images are obtained using libravatar protocol (see <https://www.libravatar.org/>) or using <http://gravatar.com/>.

2.3 Translating using Weblate

2.3.1 Translation links

Once you navigate to a translation, you will be shown set of links which lead to translation. These are results of various checks, like untranslated or fuzzy strings. Should no other checks fire, there will be still link to all translations. Alternatively you can use search field to find translation you need to fix.

Hosted Weblate Dashboard Your translations Projects Documentation Michal Čihař

Koha / opac-prog / Czech přeloženo 99%

Overview Search History Activity Statistics Files Tools Share

Translation status

Strings 2343 100.0%

Words 9274 100.0%

Good translations Translations with failing checks Fuzzy translations

Strings to check

- All strings 2343
- Strings with any failing checks 455
- This message has more than one translation in this project 25
- Source and translated strings are same 429
- Source and translation do not both end with a question mark or it is not correctly spaced 1

Project Information

Project website <https://github.com/xmorave2/koha-czech>

Translation license GPL-2.0

Git repository <git://github.com/xmorave2/koha-czech.git>

Git branch master 3146783

Git repository with Weblate translations <git://git.weblate.org/koha.git>

Other translations

Project	Translated	Words	Fuzzy	Checks	Suggestions
Koha/staff-prog	99.9%	99.8%	0.0%	1274	1
Koha/opac-ccsr	100.0%	100.0%	0.0%	5	0
Koha/pref	100.0%	100.0%	0.0%	36	0

Good translations Translations with failing checks Fuzzy translations

Powered by Weblate 2.0-dev About Weblate Contact us Documentation Donate to Weblate!

2.3.2 Suggestions

As an anonymous user, you have no other choice than making a suggestion. However if you are logged in you can still decide to make only a suggestion instead of saving translation, for example in case you are unsure about the translation and you want somebody else to review it.

Примечание: Permissions might vary depending on your setup, what is described is default Weblate behaviour.

2.3.3 Translating

On translate page, you are shown source string and edit area for translating. Should the translation be plural, multiple source strings and edit areas are shown, each described with label for plural form.

Any special whitespace chars are underlined in red and indicated with grey symbols. Also more than one space is underlined in red to allow translator to keep formatting.

There are various extra information which can be shown on this page. Most of them are coming from the project source code (like context, comments or where the message is being used). When you configure

secondary languages in your preferences, translation to these languages will be shown.

Bellow translation can be also shown suggestions from other users, which you can accept or delete.

Plurals

What are plurals? Generally spoken plurals are words which take into account numeric meanings. But as you may imagine each language has its own definition of plurals. English, for example, supports one plural. We have a singular definition, for example «car», which means implicit one car, and we have the plural definition, «cars» which could mean more than one car but also zero cars. Other languages like Czech or Arabic have more plurals and also the rules for plurals are different.

Weblate does have support for translating these and offers you one field to translate every plural separately. The number of fields and how it is used in the translated application depends on plural equation which is different for every language. Weblate shows the basic information, but you can find more detailed description in the [Language Plural Rules](#) from the Unicode Consortium.

Hosted Weblate
 Dashboard
 Your translations ▾
 Projects ▾
 Documentation
 Michal Čihař

phpMyAdmin / master / Czech / translate

⏪ ⏩ All strings (582 / 2973) ⏪ ⏩

Zen

Translate

Source

One

Total: <i>%s</i> match

Other

Total: <i>%s</i> matches

One

⌂ Copy → ↶ ↷ ...

Celkem: <i>%s</i> odpovídající záznam

Few

⌂ Copy → ↶ ↷ ...

Celkem: <i>%s</i> odpovídající záznamy

Other

⌂ Copy → ↶ ↷ ...

Celkem: <i>%s</i> odpovídajících záznamů

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

Documentation for plurals.

☐ Fuzzy

Save Suggest

Commit message: Additional text to include in t

Glossary

No related strings were found in the glossary.

Manage glossary

Source information

Flags

php-format

Source string location

libraries/DbSearch.class.php:300

Source string age

3 weeks ago

Translation file

po/cs.po, translation unit 582

String priority

Medium

Failing checks

Multiple failing checks ×

Nearby messages
 Comments
 Machine translation
 Search
 History

History

When	User	Action	Translation
No recent activity has been recorded.			

Browse changes

Follow using RSS

Powered by Weblate 2.0-dev
 About Weblate
 Contact us
 Documentation
 Donate to Weblate!

Keyboard shortcuts

While translating you can use following keyboard shortcuts:

Alt+Home Navigates to first translation in current search.

Alt+End Navigates to last translation in current search.

Alt+PageUp Navigates to previous translation in current search.

Alt+PageDown Navigates to next translation in current search.

12

Глава 2. Translators guide

Alt+Enter or Ctrl+Enter or Command+Enter Saves current translation.

Ctrl+Shift+Enter or Command+Shift+Enter Umarks translation as fuzzy and submits it.

Alt+E Focus translation editor.

Alt+C Focus comment editor.

Alt+M Shows machine translation tab.

Alt+N Shows nearby strings tab.

Alt+S Shows search tab.

Translation context

Translation context part allows you to see related information about current string.

Nearby messages Displays messages which are located nearby in translation file. These usually are also used in similar context and you might want to check them to keep translation consistent.

Similar messages Messages which are similar to currently one, which again can help you to stay consistent within translation.

All locations In case message appears in multiple places (eg. multiple components), this tab shows all of them and for inconsistent translations (see *Inconsistent*) you can choose which one to use.

Glossary Displays words from project glossary which are used in current message.

Recent edits List of people who have changed this message recently using Weblate.

Project Project information like instructions for translators or information about VCS repository.

If translation format supports it, you can also follow links to source code which contains translated strings.

Translation history

Every change is by default (unless disabled in component settings) saved in the database and can be reverted. Of course you can still also revert anything in underlying version control system.

2.3.4 Export and import

Weblate supports both export and import of translation files. This allows you to work offline and then merge changes back. Your changes will be merged within existing translation (even if it has been changed meanwhile).

For some formats you can also download compiled file to use withing application (eg. .mo files for GNU Gettext).

Примечание: This ability might be limited by *Access control*.

Import method

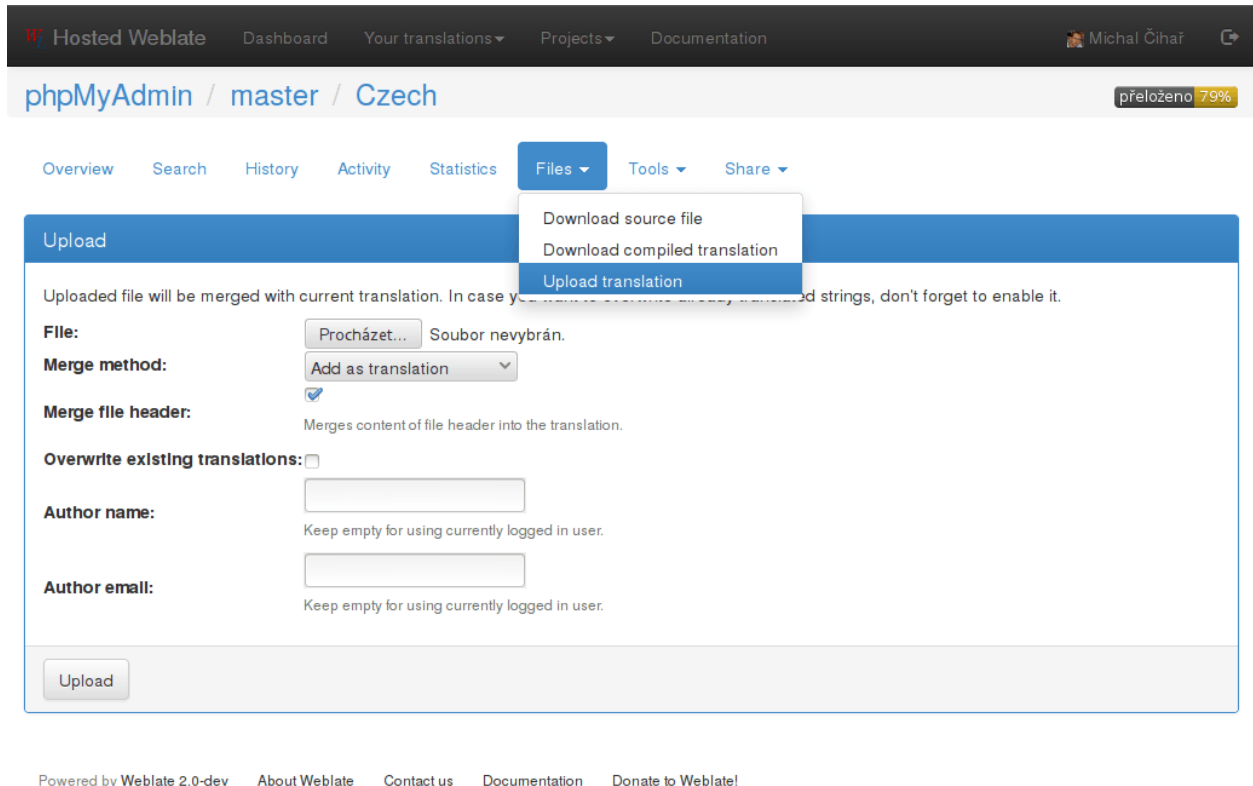
You can choose how imported strings will be merged out of following options:

Add as translation Imported translations are added as translation. This is most usual and default behavior.

Add as a suggestion Imported translations are added as suggestions, do this when you want to review imported strings.

Add as fuzzy translation Imported translations are added as fuzzy translations. This can be useful for review as well.

Additionally, when adding as a translation, you can choose whether to overwrite already translated strings or not or how to handle fuzzy strings in imported file.



2.3.5 Glossary

Each project can have assigned glossary for any language. This could be used for storing terminology for given project, so that translations are consistent. You can display terms from currently translated string in bottom tabs.

Managing glossaries

On project page, on *Glossaries* tab, you can find link *Manage all glossaries*, where you can start new glossaries or edit existing ones. Once glossary is existing, it will also show up on this tab.

Hosted Weblate Dashboard Your translations Projects Documentation Michal Čihař

GePeS translated 88%

Overview History Activity Tools Share

Resources

Resource	Translated
master	88.5%

Good translations
Translations with failing checks
Fuzzy translations

Project Information

Project website <http://cihar.com/software/gepes/>

Glossaries

Czech 1

Manage all glossaries

Powered by Weblate 2.0-dev About Weblate Contact us Documentation Donate to Weblate!

On further page, you can choose which glossary to manage (all languages used in current project are shown). Following this language link will lead you to page, which can be used to edit, import or export the glossary:

Hosted Weblate Dashboard Your translations Projects Documentation Michal Čihař

GePeS / glossaries / Czech

Browse Add new word Import glossary Export glossary History

1 / 1 Starting letter: Any

Source	Translation	
latitude	šířka	Edit Delete

Powered by Weblate 2.0-dev About Weblate Contact us Documentation Donate to Weblate!

2.3.6 Machine translation

Based on configuration and your language, Weblate provides buttons for following machine translation tools. All machine translations are available on single tab on translation page.

См.также:

Machine translation setup

2.4 Checks and fixups

2.4.1 Automatic fixups

In addition to *Quality checks*, Weblate can also automatically fix some common errors in translated strings. This can be quite powerful feature to prevent common mistakes in translations, however use it with caution as it can cause silent corruption as well.

См.также:

AUTOFIX_LIST

2.4.2 Quality checks

Weblate does wide range of quality checks on messages. The following section describes them in more detail. The checks take account also special rules for different languages, so if you think the result is wrong, please report a bug.

См.также:

CHECK_LIST, *Customizing checks*

Translation checks

These are executed on every translation change and help translators to keep good quality of translations.

Unchanged translation

The source and translated strings are the same at least in one of the plural forms. This check ignores some strings which are quite usually same in all languages and strips various markup, which can occur in the string, to reduce number of false positives.

This check can help finding strings which were mistakenly not translated .

Starting or trailing newline

Source and translated do not both start (or end) with a newline.

Newlines usually appear in source string for a good reason, so omitting or adding it can lead to formatting problems when the translated text is used in the application.

Starting spaces

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

Space in beginning is usually used for indentation in the interface and thus is important to keep.

Trailing space

Source and translated do not both end with a space.

Trailing space is usually used to give space between neighbouring elements, so removing it might break application layout.

Trailing stop

Source and translated do not both end with a full stop. Full stop is also checked in various language variants (Chinese, Japanese, Devanagari or Urdu).

Whet the original string is a sentence, the translated one should be sentence as well to be consistent within the translated content.

Trailing colon

Source and translated do not both end with a colon or the colon is not correctly spaced. This includes spacing rules for languages like French or Breton. Colon is also checked in various language variants (Chinese or Japanese).

Colon is part of a label and should be kept to provide consistent translation. Weblate also checks for various typographic conventions for colon, for example in some languages it should be preceded with space.

Trailing question

Source and translated do not both end with a question mark or it is not correctly spaced. This includes spacing rules for languages like French or Breton. Question mark is also checked in various language variants (Armenian, Arabic, Chinese, Korean, Japanese, Ethiopic, Vai or Coptic).

Question mark indicates question and this semantics should be kept in translated string as well. Weblate also checks for various typographic conventions for question mark, for example in some languages it should be preceded with space.

Trailing exclamation

Source and translated do not both end with an exclamation mark or it is not correctly spaced. This includes spacing rules for languages like French or Breton. Exclamation mark is also checked in various language variants (Chinese, Japanese, Korean, Armenian, Limbu, Myanmar or Nko).

Exclamation mark indicates some important statement and this semantics should be kept in translated string as well. Weblate also checks for various typographic conventions for exclamation mark, for example in some languages it should be preceded with space.

Trailing ellipsis

Source and translation do not both end with an ellipsis. This only checks for real ellipsis (...) not for three dots (...).

Ellipsis is usually rendered nicer than three dots, so it's good to keep it when the original string was using that as well.

См.также:

<https://en.wikipedia.org/wiki/Ellipsis>

Format strings

Format string does not match source. Weblate supports following formats:

- Python format
- Python brace format
- PHP format
- C format
- Javascript format

Omitting format string from translation usually cause severe problems, so you should really keep the format string matching the original one.

См.также:

Python string formatting, Python brace format, PHP format strings, C printf format

Missing plurals

Some plural forms are not translated. Check plural form definition to see for which counts each plural form is being used.

Not filling in some plural forms will lead to showing no text in the application in case this plural would be displayed.

Inconsistent

More different translations of one string in a project. This can also lead to inconsistencies in displayed checks. You can find other translations of this string on *All locations* tab.

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

Mismatched \n

Number of \n in translation does not match source.

Usually escaped newlines are important for formatting program output, so this should match to source.

Mismatched BBcode

BBcode in translation does not match source.

This code is used as a simple markup to highlight important parts of a message, so it is usually a good idea to keep them.

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

Zero-width space

Translation contains extra zero-width space (<U+200B>) character.

This character is usually inserted by mistake, though it might have legitimate use. Some programs might have problems when this character is used.

См.также:

https://en.wikipedia.org/wiki/Zero-width_space

XML tags mismatch

XML tags in translation do not match source.

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

Source checks

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

Optional plural

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

For example with Gettext in Python it could be:

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

Ellipsis

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

Using Unicode character is in most cases better approach and looks better when rendered.

См.также:

<https://en.wikipedia.org/wiki/Ellipsis>

Multiple failing checks

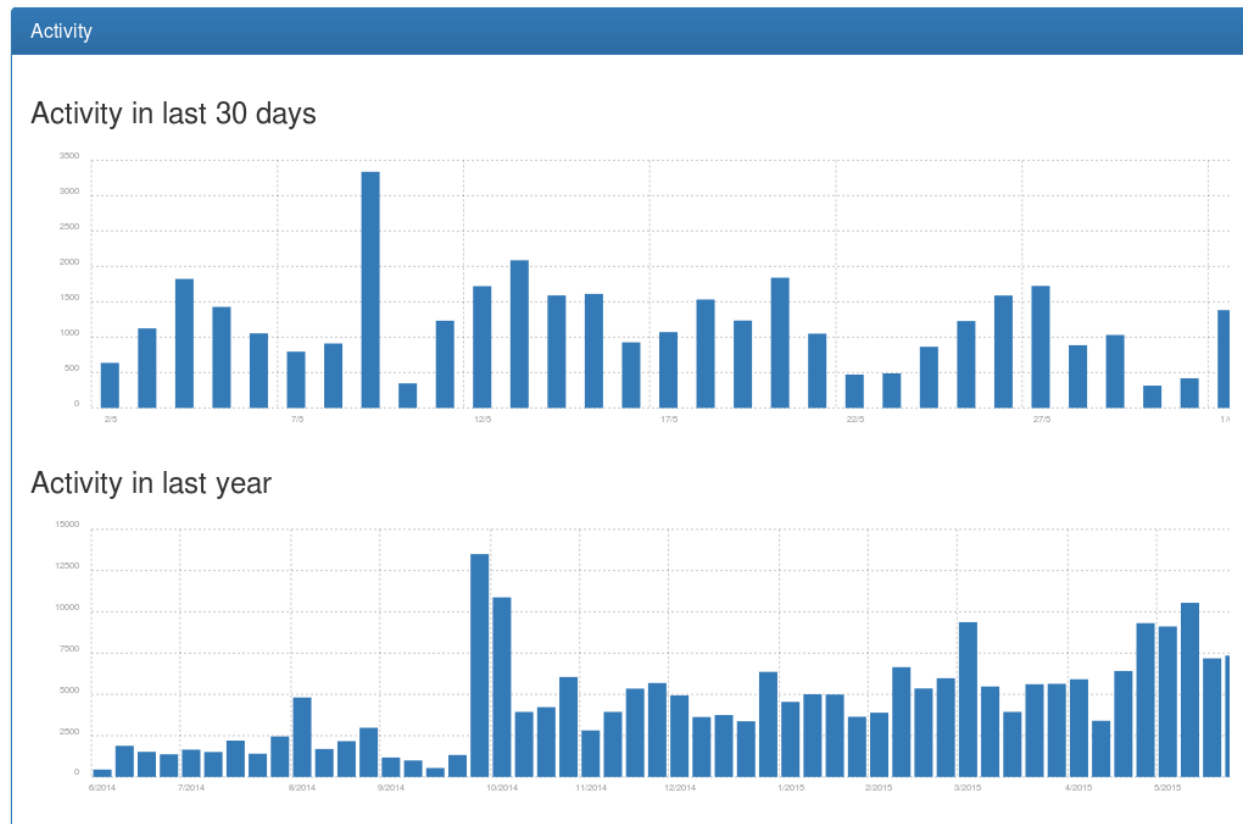
More translations of this string have some failed quality checks. This is usually indication that something could be done about improving the source string.

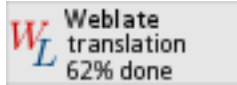
This check can be quite often caused by missing full stop at the end of sentence or similar minor issues which translators tend to fix in translations, while it would be better to fix it in a source string.

Using Weblate for translating your projects can bring you quite a lot of benefits. It's only up to you how much of that you will use.

3.1 Activity reports

You can check activity reports for translations, project or individual users.





All these badges come with links to simple page which explains users how to translate using Weblate:



Get involved in Weblate!

Hi, and thank you for your interest!

Weblate is being translated using [Weblate](#), a web tool designed to ease translating for both developers and translators.

[Translation project for Weblate](#) currently contains 1104 strings for translation and is [being translated into 33 languages](#). Overall, these translations are 62.3% complete.

If you would like to contribute to translation of Weblate, you need to [register on this server](#).

Once you have activated your account just proceed to the [translation section](#).

Powered by [Weblate 1.7](#) [About Weblate](#) [Contact us](#) [Documentation](#) [Donate to Weblate!](#)

3.3 Reviewing source strings

3.3.1 Source strings checks

Weblate includes quite a lot of *Quality checks*. Some of them also focus on quality of source strings. These can give you some hints for making strings easier to translate. You can check failing source checks on *Source* tab of every component.

3.3.2 Failing checks on translation

On the other side, failing translation checks might also indicate problem in the source strings. Translators often tend to fix some mistakes in translation instead of reporting it - typical example is missing full stop at the end of sentence, but there are more such cases.

Reviewing all failing checks on your translation can bring you valuable feedback for improving source strings as well.

3.3.3 String comments

Weblate allows translators to comment on both translation and source strings. Each *Component configuration* can be configured to receive such comments on email address and sending this to developers mailing list is usually best approach. This way you can monitor when translators find problems and fix them quickly.

3.4 Adding new translations

Weblate can add new language files to your project automatically for most of the *Supported formats*. This feature needs to be enabled in the *Component configuration*. In case this is not enabled (or available for your file format) the files have to be added manually to the VCS.

Weblate will automatically detect new languages which are added to the VCS repository and makes them available for translation. This makes adding new translations incredibly easy:

1. Add the translation file to VCS.
2. Let Weblate update the repository (usually set up automatically, see *Updating repositories*).

4.1 Quick setup guide

Примечание: This is just a quick guide for installing and starting to use Weblate for testing purposes. Please check *Installation instructions* for more real world setup instructions.

4.1.1 Installing from sources

1. Install all required dependencies, see *Requirements*.
2. Grab Weblate sources (either using Git or download a tarball) and unpack them.
3. Copy `weblate/settings_example.py` to `weblate/settings.py` and adjust it to match your setup. You will at least need to configure database connection (possibly adding user and creating the database). Check *Configuration* for Weblate specific configuration options.
4. Create database which will be used by Weblate, *Creating database for Weblate*.
5. Build Django tables and initial data:

```
./manage.py migrate
./manage.py collectstatic
./scripts/generate-locales # If you are using Git checkout
```

6. Configure webserver to serve Weblate, see *Running server*.

4.1.2 Using prebuilt appliance

1. Download the appliance and start it. You need to choose format depending on your target environment.
2. Everything should be set up immediately after boot, though you will want to adjust some settings to improve security, see *SUSE Studio appliance*.

4.1.3 Installing on OpenShift

1. You can install Weblate on OpenShift PaaS directly from its git repository using the OpenShift Client Tools:

```
rhc -aweblate app create -t python-2.7 --from-code https://github.com/nijel/weblate.  
↪git#weblate-2.4 --no-git
```

2. After installation everything should be preconfigured and you can immediately start to add a translation project as described below. For more information, including on how to retrieve the generated admin password, see *Weblate on OpenShift*.

4.1.4 Adding translation

1. Open admin interface (<http://localhost/admin/>) and create project you want to translate. See *Project configuration* for more details.

All you need to specify here is project name and its website.

2. Create component which is the real object for translating - it points to VCS repository and selects which files to translate. See *Component configuration* for more details.

The important fields here being component name, VCS repository address and mask for finding translatable files. Weblate supports wide range of formats including Gettext PO files, Android resource strings, OS X string properties, Java properties or Qt Linguist files, see *Supported formats* for more details.

3. Once above is completed (it can be lengthy process depending on size of your VCS repository and number of messages to translate), you can start translating.

4.2 Installation instructions

4.2.1 Requirements

Python (2.7, 3 is not supported) <https://www.python.org/>

Django (>= 1.7) <https://www.djangoproject.com/>

Translate-toolkit (>= 1.10.0) <http://toolkit.translatehouse.org/>

Git (>= 1.6) <http://git-scm.com/>

Mercurial (>= 2.8) (optional for Mercurial repositories support) <https://mercurial.selenic.com/>

python-social-auth (>= 0.2.0) <http://psa.matiasaguirre.net/>

Whoosh (>= 2.5, 2.5.7 is recommended, 2.6.0 is broken) <https://bitbucket.org/mchaput/whoosh/wiki/Home>

PIL or Pillow library <https://python-pillow.github.io/>

lxml (>= 3.1.0) <http://lxml.de/>

dateutil <http://labix.org/python-dateutil>

django_compressor <https://github.com/django-compressor/django-compressor>

libravatar (optional for federated avatar support) <https://pypi.python.org/pypi/pyLibravatar>

pyuca (>= 1.1) (optional for proper sorting of strings) <https://github.com/jtauber/pyuca>

babel (optional for Android resources support) <http://babel.pocoo.org/>

Database backend Any database supported in Django will work, check their documentation for more details.

hub (optional for sending pull requests to GitHub) <https://hub.github.com/>

Requirements on Debian or Ubuntu

On Debian or Ubuntu, most of requirements are already packaged, to install them you can use apt-get:

```
apt-get install python-django translate-toolkit \
    python-whoosh python-pil python-libravatar \
    python-babel Git mercurial python-social-auth

# Optional for database backend

apt-get install python-mysqldb      # For MySQL
apt-get install python-psycopg2    # For PostgreSQL
```

For Debian 7.0 (Wheezy) or older, you need to install several Python modules manually using pip as versions shipped in distribution are too old:

```
# Dependencies for python-social-auth
apt-get install python-requests-oauthlib python-six python-openid

pip install python-social-auth Django Whoosh
```

For proper sorting of a unicode strings, it is recommended to install pyuca:

```
pip install pyuca
```

Depending on how you intend to run Weblate and what you already have installed, you might need additional components:

```
# Web server option 1: nginx and uwsgi
apt-get install nginx uwsgi uwsgi-plugin-python

# Web server option 2: Apache with mod_wsgi
apt-get install apache2 libapache2-mod-wsgi

# Caching backend: memcached
apt-get install memcached

# Database option 1: mariadb
apt-get install mariadb-server

# Database option 2: mysql
apt-get install mysql-server

# Database option 3: postgresql
apt-get install postgresql

# SMTP server
apt-get install exim4

# GitHub PR support: hub
# See https://hub.github.com/
```

Requirements on openSUSE

Most of requirements are available either directly in openSUSE or in `devel:languages:python` repository:

```
zypper install python-Django translate-toolkit \
    python-Whoosh python-Pillow python-python-social-auth \
    python-babel Git mercurial python-pyuca

# Optional for database backend
zypper install python-MySQL-python # For MySQL
zypper install python-psycopg2    # For PostgreSQL
```

Depending on how you intend to run Weblate and what you already have installed, you might need additional components:

```
# Web server option 1: nginx and uwsgi
zypper install nginx uwsgi uwsgi-plugin-python

# Web server option 2: Apache with mod_wsgi
zypper install apache2 apache2-mod_wsgi

# Caching backend: memcached
zypper install memcached

# Database option 1: mariadb
zypper install mariadb

# Database option 2: mysql
zypper install mysql

# Database option 3: postgresql
zypper install postgresql

# SMTP server
zypper install postfix

# GitHub PR support: hub
# See https://hub.github.com/
```

Requirements on OSX

If your python was not installed using brew, make sure you have this in your `.bash_profile` file or executed somehow:

```
export PYTHONPATH="/usr/local/lib/python2.7/site-packages:$PYTHONPATH"
```

This configuration makes the installed libraries available to Python.

Requirements using pip installer

Most requirements can be also installed using pip installer:

```
pip install -r requirements.txt
```

Also you will need header files for `python-dev`, `libxml2`, `libxslt` and `libfreetype6` to compile some of the required Python modules.

All optional dependencies (see above) can be installed using:

```
pip install -r requirements-optional.txt
```

On Debian or Ubuntu you can install them using:

```
apt-get install libxml2-dev libxslt-dev libfreetype6-dev python-dev
```

On openSUSE or SLES you can install them using:

```
zypper install libxslt-devel libxml2-devel freetype-devel python-devel
```

4.2.2 Filesystem permissions

Weblate process needs to be able to read and write to the directory where it keeps data - *DATA_DIR*.

The default configuration places them in same tree as Weblate sources, however you might prefer to move these to 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 run under same user as Weblate itself is running, otherwise permissions on some files might be wrong.

4.2.3 Creating database for Weblate

It is recommended to run Weblate on some database server. Using SQLite backend is really good for testing purposes only.

См.также:

Use powerful database engine, Django's databases

Creating database in PostgreSQL

It is usually good idea to run Weblate in separate database and separate user:

```
# If PostgreSQL was not installed before, set the master password
sudo -u postgres psql postgres -c "\password postgres"

# Create database user called "weblate"
sudo -u postgres createuser -D -A -P weblate

# Create database "weblate" owned by "weblate"
sudo -u postgres createdb -O weblate weblate
```

Creating database in MySQL

When using MySQL, don't forget to create database with UTF-8 encoding:

```
# Grant all privileges to weblate user
GRANT ALL PRIVILEGES ON weblate.* TO 'weblate'@'localhost' IDENTIFIED BY 'password';

# Create database
CREATE DATABASE weblate CHARACTER SET utf8mb4;
# Use utf8 if above fails:
# CREATE DATABASE weblate CHARACTER SET utf8;
```

4.2.4 Other configurations

Outgoing mail

Weblate sends out emails on various occasions - for account activation and on various notifications configured by users. For this it needs access to the SMTP server, which will handle this.

The mail server setup is configured using settings `EMAIL_HOST`, `EMAIL_HOST_PASSWORD`, `EMAIL_HOST_USER` and `EMAIL_PORT`. Their names are quite self-explaining, but you can find our more information in the [Django documentation](#) on them.

Setting up hub

Pushing changes to GitHub as pull request requires a configured *hub* installation on your server. Follow the installation instructions at <https://hub.github.com> and perform an action with *hub* to finish the configuration, for example:

```
hub clone octocat/Spoon-Knife
```

hub will ask you for your GitHub credentials, retrieve a token and store it into `~/.config/hub`.

Примечание: Use the username you configured *hub* with as `GITHUB_USERNAME`.

4.2.5 Installation

См.также:

Sample configuration

Copy `weblate/settings_example.py` to `weblate/settings.py` and adjust it to match your setup. You will probably want to adjust following options:

ADMINS

List of site administrators to receive notifications when something goes wrong, for example notifications on failed merge or Django errors.

См.также:

<https://docs.djangoproject.com/en/stable/ref/settings/#admins>

ALLOWED_HOSTS

If you are running Django 1.5 or newer, you need to set this to list of hosts your site is supposed to serve. For example:

```
ALLOWED_HOSTS = ['demo.weblate.org']
```

См.также:

https://docs.djangoproject.com/en/stable/ref/settings/#std:setting-ALLOWED_HOSTS

SESSION_ENGINE

Configure how your sessions will be stored. In case you keep default database backed engine you should schedule `./manage.py clearsessions` to remove stale session data from the database.

См.также:

<https://docs.djangoproject.com/en/stable/topics/http/sessions/#configuring-sessions>

DATABASES

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

См.также:

Creating database for Weblate <https://docs.djangoproject.com/en/stable/ref/settings/#databases>, <https://docs.djangoproject.com/en/stable/ref/databases/>

DEBUG

Disable this for production server. With debug mode enabled, Django will show backtraces in case of error to users, when you disable it, errors will go by email to `ADMINS` (see above).

Debug mode also slows down Weblate as Django stores much more information internally in this case.

См.также:

<https://docs.djangoproject.com/en/stable/ref/settings/#debug>

DEFAULT_FROM_EMAIL

Email sender address for outgoing email, for example registration emails.

См.также:

[DEFAULT_FROM_EMAIL documentation](#)

SECRET_KEY

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

SERVER_EMAIL

Email used as sender address for sending emails to administrator, for example notifications on failed merge.

См.также:

[SERVER_EMAIL documentation](#)

4.2.6 Filling up the database

After your configuration is ready, you can run `./manage.py migrate` to create database structure. Now you should be able to create translation projects using admin interface.

In case you want to run installation non interactively, you can use `./manage.py migrate --noinput` and then create admin user using `createadmin` command.

You should also login to admin interface (on `/admin/` URL) and adjust default site name to match your domain by clicking on *Sites* and there changing the `example.com` record to match your real domain name.

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

Примечание: If you are running version from Git, you should also regenerate locale files every time you are upgrading. You can do this by invoking script `./scripts/generate-locales`.

См.также:

Configuration, Access control, Why does registration contain example.com as domain?

4.2.7 Production setup

For production setup you should do following adjustments:

Disable debug mode

Disable Django's debug mode by:

```
DEBUG = False
```

With debug mode Django stores all executed queries and shows users backtraces of errors what is not desired in production setup.

См.также:

Installation

Properly configure admins

Set correct admin addresses to `ADMINS` setting for defining who will receive mail in case something goes wrong on the server, for example:

```
ADMINS = (
    ('Your Name', 'your_email@example.com'),
)
```

См.также:

Installation

Set correct site name

Adjust site name in admin interface, otherwise links in RSS or registration emails will not work.

Please open admin interface and edit default site name and domain under the *Sites > Sites* (or you can do that directly at `/admin/sites/site/1/` URL under your Weblate installation). You have to change the *Domain name* to match your setup.

You might want to set `ENABLE_HTTPS` as well if you serve site over https.

Alternatively you can set the site name from command line using `changesite`.

См.также:

Why does registration contain example.com as domain?, *changesite*, <https://docs.djangoproject.com/en/stable/ref/contrib/sites/>

Enable indexing offloading

Enable `OFFLOAD_INDEXING` to prevent locking issues and improve performance. Don't forget to schedule indexing in background job to keep the index up to date.

См.также:

Fulltext search, *OFFLOAD_INDEXING*, *Running maintenance tasks*

Use powerful database engine

Use powerful database engine (SQLite is usually not good enough for production environment), for example setup for MySQL:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'weblate',
        'USER': 'weblate',
        'PASSWORD': 'weblate',
        'HOST': '127.0.0.1',
        'PORT': '',
    }
}
```

См.также:

Installation, Django's databases

Enable caching

If possible, use memcache from Django by adjusting `CACHES` configuration variable, for example:

```
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '127.0.0.1:11211',
    }
}
```

См.также:

Avatar caching, Django's cache framework

Avatar caching

In addition to caching of Django, Weblate performs caching of avatars. It is recommended to use separate, file backed cache for this purpose:

```
CACHES = {
    'default': {
        # Default caching backend setup, see above
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '127.0.0.1:11211',
    },
    'avatar': {
        'BACKEND': 'django.core.cache.backends.filebased.FileBasedCache',
        'LOCATION': os.path.join(BASE_DIR, 'avatar-cache'),
        'TIMEOUT': 604800,
        'OPTIONS': {
            'MAX_ENTRIES': 1000,
        },
    },
}
```

См.также:

ENABLE_AVATARS, *Enable caching*, Django's cache framework

Configure email addresses

Weblate needs to send out emails on several occasions and these emails should have 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'
```

См.также:

Installation, *DEFAULT_FROM_EMAIL* documentation, *SERVER_EMAIL* documentation

Allowed hosts setup

Django 1.5 and newer require `ALLOWED_HOSTS` to hold list of domain names your site is allowed to serve, having it empty will block any request.

См.также:

https://docs.djangoproject.com/en/stable/ref/settings/#std:setting-ALLOWED_HOSTS

Federated avatar support

By default, Weblate relies on <https://www.libravatar.org/> for avatars. When you install `pyLibavatar`, you will get proper support for federated avatars.

pyuca library

`pyuca` library is optionally used by Weblate to sort Unicode strings. This way language names are properly sorted even in non-ASCII languages like Japanese, Chinese or Arabic or for languages with accented letters.

Django secret key

The `SECRET_KEY` setting is used by Django to sign cookies and you should really use own value rather than using the one coming from example setup.

You can generate new key using `examples/generate-secret-key` shipped with Weblate.

См.также:

https://docs.djangoproject.com/en/stable/ref/settings/#std:setting-SECRET_KEY

Static files

If you see purely designed admin interface, the CSS files required for it are not loaded. This is usually if you are running in non-debug mode and have not configured your web server to serve them. Recommended setup is described in the *Serving static files* chapter.

См.также:

Running server, *Serving static files*

Home directory

Изменено в версии 2.1: This is no longer required, Weblate now stores all its data in `DATA_DIR`.

The home directory for user which is running Weblate should be existing and 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 (depends on Git version you use).

You can change the directory used by Weblate in `settings.py`, for example to set it to `configuration` directory under Weblate tree:

```
os.environ['HOME'] = os.path.join(BASE_DIR, 'configuration')
```

Примечание: On Linux and other UNIX like systems, the path to user's home directory is defined in `/etc/passwd`. Many distributions default to 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 different user or change this setting.

См.также:

Private repositories

Template loading

It is recommended to use cached template loader for Django. It caches parsed templates and avoids need to do the parsing with every single request. You can configure it using following snippet:

```
TEMPLATE_LOADERS = (
    ('django.template.loaders.cached.Loader', (
        'django.template.loaders.filesystem.Loader',
        'django.template.loaders.app_directories.Loader',
    )),
)
```

См.также:

[Django documentation on template loading](#)

Running maintenance tasks

For optimal performace, it is good idea to run some maintenance tasks in the background.

On Unix system, this can be scheduled using cron:

```
# Fulltext index updates
*/5 * * * * cd /usr/share/weblate/; ./manage.py update_index

# Cleanup stale objects
@daily cd /usr/share/weblate/; ./manage.py cleanuptrans

# Commit pending changes after 96 hours
@hourly cd /usr/share/weblate/; ./manage.py commit_pending --all --age=96 --verbosity=0
```

См.также:

Enable indexing offloading, `update_index`, `cleanuptrans`, `commit_pending`

4.2.8 Running server

Running Weblate is not different from running any other Django based application. Django is usually executed as uwsgi or fcgi (see examples for different webservers below).

For testing purposes, you can use Django builtin web server:

```
./manage.py runserver
```

Serving static files

Изменено в версии 2.4: Prior to version 2.4 Weblate didn't properly use Django static files framework and the setup was more complex.

Django needs to collect its static files to a single directory. To do so, execute `./manage.py collectstatic --noinput --link`. This will store links to static files into directory specified by `STATIC_ROOT` setting.

It is recommended to serve static files directly by your web server, you should use that for following paths:

`/static/` Serves static files for Weblate and admin interface (from defined by `STATIC_ROOT`).

`/favicon.ico` Should be rewritten to rewrite rule to serve `/static/favicon.ico`

`/robots.txt` Should be rewritten to rewrite rule to serve `/static/robots.txt`

См.также:

<https://docs.djangoproject.com/en/stable/howto/deployment/> <https://docs.djangoproject.com/en/stable/howto/static-files/deployment/>

Sample configuration for Lighttpd

The configuration for Lighttpd web server might look like following (available as `examples/lighttpd.conf`):

```

fastcgi.server = (
    "/weblate.fcgi" => (
        "main" => (
            "socket" => "/var/run/django/weblate.socket",
            "check-local" => "disable",
        ),
    ),
),
),
alias.url = (
    "/static" => "/usr/share/weblate/data/static/",
)

url.rewrite-once = (
    "^(/*static.*)$" => "$1",
    "^/*favicon\.ico$" => "/static/favicon.ico",
    "^/*robots\.txt$" => "/static/robots.txt",
    "^(/.*)$" => "/weblate.fcgi$1",
)

expire.url = (
    "/static/" => "access 1 months",
    "/favicon.ico" => "access 1 months",
)

```

Sample configuration for Apache

Following configuration runs Weblate as WSGI, you need to have enabled `mod_wsgi` (available as `examples/apache.conf`):

```

#
# VirtualHost for weblate
#
WSGIPythonPath /usr/share/weblate
# If using virtualenv, you need to add it to search path as well:
# WSGIPythonPath /usr/share/weblate:/path/to/your/venv/lib/python2.7/site-packages
<VirtualHost *:80>
    ServerAdmin admin@image.weblate.org
    ServerName image.weblate.org

    Alias /robots.txt /usr/share/weblate/data/static/robots.txt
    Alias /favicon.ico /usr/share/weblate/data/static/favicon.ico

    Alias /static/ /usr/share/weblate/data/static/

    <Directory /usr/share/weblate/data/static/>
        Require all denied
    </Directory>

    WSGIScriptAlias / /usr/share/weblate/weblate/wsgi.py
    WSGIPassAuthorization On

    <Directory /usr/share/weblate/weblate>
        <Files wsgi.py>
            Require all denied
        </Files>
    </Directory>

```

(continues on next page)

(продолжение с предыдущей страницы)

```
</VirtualHost>
```

Sample configuration for nginx

Following configuration runs Weblate as uwsgi under nginx webserver.

Configuration for nginx (also available as `examples/weblate.nginx.conf`):

```
server {
    listen 80;
    server_name weblate;
    root /path/to/weblate/weblate;

    location /favicon.ico {
        alias /path/to/weblate/data/static/favicon.ico;
        expires 30d;
    }

    location /robots.txt {
        alias /path/to/weblate/data/static/robots.txt;
        expires 30d;
    }

    location /static {
        alias /path/to/weblate/data/static/;
        expires 30d;
    }

    location / {
        include uwsgi_params;
        # Needed for long running operations in admin interface
        uwsgi_read_timeout 3600;
        uwsgi_pass 127.0.0.1:8080;
    }
}
```

Configuration for uwsgi (also available as `examples/weblate.uwsgi.ini`):

```
[uwsgi]
plugins      = python
master       = true
protocol     = uwsgi
socket       = 127.0.0.1:8080
wsgi-file    = /path/to/weblate/weblate/wsgi.py
python-path  = /path/to/weblate
# Needed for OAuth/OpenID
buffer-size  = 8192
# Increase number of workers for heavily loaded sites
#workers     = 6
# Needed for background processing
enable-threads = true
# Child processes do not need file descriptors
close-on-exec = true
```

Running Weblate under path

Изменено в версии 1.3: This is supported since Weblate 1.3.

Sample Apache configuration to serve Weblate under `/weblate`. Again using `mod_wsgi` (also available as `examples/apache-path.conf`):

```
# Example Apache configuration for running Weblate under /weblate path

WSGIPythonPath /usr/share/weblate
# If using virtualenv, you need to add it to search path as well:
# WSGIPythonPath /usr/share/weblate:/path/to/your/venv/lib/python2.7/site-packages
<VirtualHost *:80>
    ServerAdmin admin@image.weblate.org
    ServerName image.weblate.org

    Alias /weblate/robots.txt /usr/share/weblate/data/static/robots.txt
    Alias /weblate/favicon.ico /usr/share/weblate/data/static/favicon.ico

    Alias /weblate/static/ /usr/share/weblate/data/static/

    <Directory /usr/share/weblate/data/static/>
        Require all denied
    </Directory>

    WSGIScriptAlias /weblate /usr/share/weblate/weblate/wsgi.py
    WSGIPassAuthorization On

    <Directory /usr/share/weblate/weblate>
        <Files wsgi.py>
            Require all denied
        </Files>
    </Directory>
</VirtualHost>
```

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

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

4.2.9 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 migrated 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.

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.

Migrating fulltext index

For the fulltext index (stored in `DATA_DIR`) it is better not to migrate it, but rather to generate fresh one using `rebuild_index`.

Other notes

Don't forget to move other services which Weblate might have been using like memcached, cron jobs or custom authentication backends.

4.3 Weblate deployments

Weblate comes with support for deployment using several technologies. This section brings overview of them.

4.3.1 SUSE Studio appliance

Weblate appliance provides preconfigured Weblate running with MySQL database as backend and Apache as web server. It is provided in many formats suitable for any form of virtualization, cloud or hardware installation.

It comes with standard set of passwords you will want to change:

Username	Password	Scope	Description
root	linux	System	Administrator account, use for local or SSH login
root		MySQL	MySQL administrator
weblate	weblate	MySQL	Account in MySQL database for storing Weblate data
admin	admin	Weblate	Weblate/Django admin user

The appliance is built using SUSE Studio and is based on openSUSE 12.3.

You should also adjust some settings to match your environment, namely:

- *Disable debug mode*
- *Set correct site name*
- *Configure email addresses*

4.3.2 Weblate and 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 configured default database, but you can switch it to MySQL or MariaDB.

Deployment

Following examples assume you have working Docker environment, with docker-compose installed. Please check Docker documentation for instructions on this.

1. Clone weblate-docker repo:

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

2. Optionally change the database provider in `docker-compose.yml`. Following options were tested, but other versions will most likely work as well: `mysql:5.6`, `mariadb:10.0`, `postgres:9.4`

It might be also good idea to change predefined passwords for the database in the `docker-compose.yml`.

3. Build Weblate containers:

```
docker-compose build
```

4. Start Weblate containers:

```
docker-compose up
```

5. Create Weblate database:

```
docker-compose run weblate-web migrate
```

6. Prepare static files:

```
docker-compose run weblate-web collectstatic
```

7. Create admin user:

```
docker-compose run weblate-web createadmin
```

Enjoy your Weblate deployment, it's accessible on port 8000 of the container.

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, Digitalocean and many more providers.

4.3.3 Weblate on OpenShift

This repository contains a configuration for the OpenShift platform as a service product, which facilitates easy installation of Weblate on OpenShift Online (<https://www.openshift.com/>), OpenShift Enterprise (<https://enterprise.openshift.com/>) and OpenShift Origin (<https://www.openshift.org/>).

Prerequisites

1. OpenShift Account

You need an account for OpenShift Online (<https://www.openshift.com/>) or another OpenShift installation you have access to.

You can register a free account on OpenShift Online, which allows you to host up to 3 applications free of charge.

2. OpenShift Client Tools

In order to follow the examples given in this documentation you need to have the OpenShift Client Tools (RHC) installed: <https://developers.openshift.com/en/managing-client-tools.html>

While there are other possibilities to create and configure OpenShift applications, this documentation is based on the OpenShift Client Tools (RHC) because they provide a consistent interface for all described operations.

Installation

You can install Weblate on OpenShift directly from Weblate's github repository with the following command:

```
rhc -aweblate app create -t python-2.7 --from-code https://github.com/nijel/weblate.git --no-git
```

The `-a` option defines the name of your weblate installation, `weblate` in this instance. You are free to specify a different name.

Optionally you can specify tag identifier right of the `#` sign to identify the version of Weblate to install (for example specify `https://github.com/nijel/weblate.git#weblate-2.0` to install Weblate 2.0). For a list of available versions see here: <https://github.com/nijel/weblate/tags>. Please note that only version 2.0 and newer can be installed on OpenShift, as older versions don't include the necessary configuration files. The `--no-git` option skips the creation of a local git repository.

You can also specify which database you want to use:

```
# For MySQL
rhc -aweblate app create -t python-2.7 -t mysql-5.5 --from-code https://github.com/nijel/weblate.
↪git --no-git

# For PostgreSQL
rhc -aweblate app create -t python-2.7 -t postgresql-9.2 --from-code https://github.com/nijel/
↪weblate.git --no-git
```

Default Configuration

After installation on OpenShift Weblate is ready to use and preconfigured as follows:

- SQLite embedded database (DATABASES)
- Random admin password
- Random Django secret key (SECRET_KEY)
- Indexing offloading if the cron cartridge is installed (`OFFLOAD_INDEXING`)
- Committing of pending changes if the cron cartridge is installed (`commit_pending`)
- Weblate machine translations for suggestions bases on previous translations (`MACHINE_TRANSLATION_SERVICES`)
- Source language for machine translations set to «en-us» (`SOURCE_LANGUAGE`)
- Weblate directories (STATIC_ROOT, `DATA_DIR`, `TTF_PATH`, Avatar cache) set according to OpenShift requirements/conventions
- Django site name and ALLOWED_HOSTS set to DNS name of your OpenShift application
- Email sender addresses set to no-reply@<OPENSHIFT_CLOUD_DOMAIN>, where <OPENSHIFT_CLOUD_DOMAIN> is the domain OpenShift runs under. In case of OpenShift Online it's rhcloud.com.

См.также:

[Customize Weblate Configuration](#)

Retrieve Admin Password

You can retrieve the generated admin password with the following command:

```
rhc -aweblate ssh credentials
```

Indexing Offloading

To enable the preconfigured indexing offloading you need to add the cron cartridge to your application and restart it:

```
rhc -aweblate add-cartridge cron
rhc -aweblate app stop
rhc -aweblate app start
```

The fulltext search index will then be updated every 5 minutes. Restarting with `rhc restart` instead will not enable indexing offloading in Weblate. You can verify that indexing offloading is indeed enabled by visiting the URL `/admin/performance/` of your application.

Pending Changes

Weblate's OpenShift configuration contains a cron job which periodically commits pending changes older than a certain age (24h by default). To enable the cron job you need to add the cron cartridge and restart Weblate as described in the previous section. You can change the age parameter by setting the environment variable `WEBLATE_PENDING_AGE` to the desired number of hours, e.g.:

```
rhc -aweblate env set WEBLATE_PENDING_AGE=48
```

Customize Weblate Configuration

You can customize the configuration of your Weblate installation on OpenShift through environment variables. Override any of Weblate's setting documented under *Configuration* using `rhc env set` by prepending the settings name with `WEBLATE_`. The variable is parsed as Python string, after replacing environment variables in it (eg. `$PATH`). To put literal `$` you need to escape it as `$$`.

For example override the `ADMINS` setting like this:

```
rhc -aweblate env set WEBLATE_ADMINS='(("John Doe", "jdoe@example.org"),)'
```

New settings will only take effect after restarting Weblate:

```
rhc -aweblate app stop
rhc -aweblate app start
```

Restarting using `rhc -aweblate app restart` does not work. For security reasons only constant expressions are allowed as values. With the exception of environment variables which can be referenced using `${ENV_VAR}`. For example:

```
rhc -aweblate env set WEBLATE_PRE_COMMIT_SCRIPTS='("${OPENSIFT_DATA_DIR}/examples/hook-generate-mo
↩",)'
```

You can check the effective settings Weblate is using by running:

```
rhc -aweblate ssh settings
```

This will also print syntax errors in your expressions. To reset a setting to its preconfigured value just delete the corresponding environment variable:

```
rhc -aweblate env unset WEBLATE_ADMINS
```

См.также:

Configuration

Updating

It is recommended that you try updates on a clone of your Weblate installation before running the actual update. To create such a clone run:

```
rhc -aweblate2 app create --from-app weblate
```

Visit the newly given URL with a browser and wait for the install/update page to disappear.

You can update your Weblate installation on OpenShift directly from Weblate's github repository by executing:

```
rhc -aweblate2 ssh update https://github.com/nijel/weblate.git
```

The identifier right of the # sign identifies the version of Weblate to install. For a list of available versions see here: <https://github.com/nijel/weblate/tags>. Please note that the update process will not work if you modified the git repository of you weblate installation. You can force an update by specifying the `--force` option to the update script. However any changes you made to the git repository of your installation will be discarded:

```
rhc -aweblate2 ssh update --force https://github.com/nijel/weblate.git
```

The `--force` option is also needed when downgrading to an older version. Please note that only version 2.0 and newer can be installed on OpenShift, as older versions don't include the necessary configuration files.

The update script takes care of the following update steps as described under *Generic upgrade instructions*.

- Install any new requirements
- `manage.py migrate`
- `manage.py setupgroups --move`
- `manage.py setuplang`
- `manage.py rebuild_index --all`
- `manage.py collectstatic --noinput --link`

4.3.4 Bitnami Weblate stack

Bitnami provides 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.

4.4 Upgrading Weblate

4.4.1 Upgrading

Generic upgrade instructions

Изменено в версии 1.2: Since version 1.2 the migration is done using South module, to upgrade to 1.2, please see [Version specific instructions](#).

Изменено в версии 1.9: Since version 1.9, Weblate also supports Django 1.7 migrations, please check [Upgrading to Django 1.7](#) for more information.

Изменено в версии 2.3: Since version 2.3, Weblate supports only Django native migrations, South is no longer supported, please check [Upgrading to Django 1.7](#) for more information.

Before upgrading, please check current [Requirements](#) as they might have changed.

To upgrade database structure, you should run:

```
./manage.py migrate
```

To collect new static files, run:

```
./manage.py collectstatic --noinput --link
```

To upgrade default set of privileges definitions (optional), run:

```
./manage.py setupgroups
```

To upgrade default set of language definitions (optional), run:

```
./manage.py setuplang
```

Version specific instructions

Upgrade from 0.5 to 0.6

On upgrade to version 0.6 you should run `./manage.py syncdb` and `./manage.py setupgroups --move` to setup access control as described in installation section.

Upgrade from 0.6 to 0.7

On upgrade to version 0.7 you should run `./manage.py syncdb` to setup new tables and `./manage.py rebuild_index` to build index for fulltext search.

Upgrade from 0.7 to 0.8

On upgrade to version 0.8 you should run `./manage.py syncdb` to setup new tables, `./manage.py setupgroups` to update privileges setup and `./manage.py rebuild_index` to rebuild index for fulltext search.

Upgrade from 0.8 to 0.9

On upgrade to version 0.9 file structure has changed. You need to move `repos` and `whoosh-index` to `weblate` folder. Also running `./manage.py syncdb`, `./manage.py setupgroups` and `./manage.py setuplang` is recommended to get latest updates of privileges and language definitions.

Upgrade from 0.9 to 1.0

On upgrade to version 1.0 one field has been added to database, you need to invoke following SQL command to adjust it:

```
ALTER TABLE `trans_subproject` ADD `template` VARCHAR(200);
```

Upgrade from 1.0 (1.1) to 1.2

On upgrade to version 1.2, the migration procedure has changed. It now uses South for migrating database. To switch to this new migration schema, you need to run following commands:

```
./manage.py syncdb
./manage.py migrate trans 0001 --fake
./manage.py migrate accounts 0001 --fake
./manage.py migrate lang 0001 --fake
```

Also please note that there are several new requirements and version 0.8 of `django-registration` is now being required, see [Requirements](#) for more details.

Once you have done this, you can use [Generic upgrade instructions](#).

Upgrade from 1.2 to 1.3

Since 1.3, `settings.py` is not shipped with Weblate, but only example settings as `settings_example.py` it is recommended to use it as new base for your setup.

Upgrade from 1.4 to 1.5

Several internal modules and paths have been renamed and changed, please adjust your `settings.py` to match that (consult `settings_example.py` for correct values).

- Many modules lost their `weblate.` prefix.
- Checks were moved to submodules.
- Locales were moved to top level directory.

The migration of database structure to 1.5 might take quite long, it is recommended to put your site offline, while the migration is going on.

Примечание: If you have update in same directory, stale `*.pyc` files might be left around and cause various import errors. To recover from this, delete all of them in Weblate's directory, for example by `find . -name '*.pyc' -delete`.

Upgrade from 1.6 to 1.7

The migration of database structure to 1.7 might take quite long, it is recommended to put your site offline, while the migration is going on.

If you are translating monolingual files, it is recommended to rerun quality checks as they might have been wrongly linked to units in previous versions.

Upgrade from 1.7 to 1.8

The migration of database structure to 1.8 might take quite long, it is recommended to put your site offline, while the migration is going on.

Authentication setup has been changed and some internal modules have changed name, please adjust your `settings.py` to match that (consult `settings_example.py` for correct values).

Also please note that there are several new requirements, see [Requirements](#) for more details.

Upgrade from 1.8 to 1.9

Several internal modules and paths have been renamed and changed, please adjust your `settings.py` to match that (consult `settings_example.py` for correct values).

См.также:

If you are upgrading to Django 1.7 in same step, please consult [Upgrading to Django 1.7](#).

Upgrade from 1.9 to 2.0

Several internal modules and paths have been renamed and changed, please adjust your `settings.py` to match that (consult `settings_example.py` for correct values).

This upgrade also requires you to upgrade python-social-auth from 0.1.x to 0.2.x series, what will most likely need to fake one of their migrations (see [Upgrading PSA with South](#) for more information):

```
./manage.py migrate --fake default
```

См.также:

If you are upgrading to Django 1.7 in same step, please consult [Upgrading to Django 1.7](#).

Upgrade from 2.0 to 2.1

Please adjust your `settings.py` to match several changes in the configuration (consult `settings_example.py` for correct values).

The filesystem paths configuration has changed, the `GIT_ROOT` and `WHOOSH_INDEX` are gone and now all data resides in `DATA_DIR`. The existing data should be automatically migrated by supplied migration, but in case of non standard setup, you might need to move these manually.

См.также:

If you are upgrading to Django 1.7 in same step, please consult [Upgrading to Django 1.7](#).

Upgrade from 2.1 to 2.2

Weblate now supports fulltext search on additional fields. In order to make it work on existing data you need to update fulltext index by:

```
./manage.py rebuild_index --clean --all
```

If you have some monolingual translations, Weblate now allows to edit template (source) strings as well. To see them, you need to reload translations, what will either happen automatically on next repository update or you can force it manually:

```
./manage.py loadpo --all
```

См.также:

If you are upgrading to Django 1.7 in same step, please consult [Upgrading to Django 1.7](#).

Upgrade from 2.2 to 2.3

If you have not yet performed upgrade to Django 1.7 and newer, first upgrade to 2.2 following instructions above. Weblate 2.3 no longer supports migration from Django 1.6.

If you were using Weblate 2.2 with Django 1.6, you will now need to fake some migrations:

```
./manage.py migrate --fake accounts 0004_auto_20150108_1424
./manage.py migrate --fake lang 0001_initial
./manage.py migrate --fake trans 0018_auto_20150213_1447
```

Previous Weblate releases contained bug which made some monolingual translations behave inconsistently for fuzzy and not translated strings, if you have such, it is recommended to run:

```
./manage.py fixup_flags --all
```

Upgrade from 2.3 to 2.4

Please adjust your `settings.py` to match several changes in the configuration (consult `settings_example.py` for correct values).

Handling of static content has been rewritten, please adjust configuration of your webserver accordingly (see [Serving static files](#) for more details). Most importantly:

- `/media/` path is no longer used
- `/static/` path now holds both admin and Weblate static files

There is now also additional dependency - `django_compressor`, please install it prior to upgrading.

4.4.2 Upgrading to Django 1.7

Django 1.7 has a new feature to handle database schema upgrade called «migrations» which is incompatible with South (used before by Weblate).

Before migrating to Django 1.7, you first need to apply all migrations from South. If you already have upgraded Django to 1.7, you can do this using `virtualenv` and `examples/migrate-south` script:

```
examples/migrate-south --settings weblate.settings
```

Once you have done that, you can run Django migrations and work as usual. For the initial setup, you might need to fake some of the migrations though:

```
./manage.py migrate --fake-initial
```

4.4.3 Migrating from Pootle

As Weblate was originally written as replacement from Pootle, it is supported to migrate user accounts from Pootle. All you need to do is to copy `auth_user` table from Pootle, user profiles will be automatically created for users as they log in and they will be asked to update their settings. Alternatively you can use `importusers` to import dumped user credentials.

4.5 Authentication

4.5.1 User registration

The default setup for Weblate is to use `python-social-auth` for handling new users. This allows them to register using form on the website and after confirming their email they can contribute or by using some third party service to authenticate.

You can also completely disable new users registration using `REGISTRATION_OPEN`.

4.5.2 Authentication backends

By default Weblate uses Django built-in authentication and includes various social authentication options. Thanks to using Django authentication, you can also import user database from other Django based projects (see *Migrating from Pootle*).

Django can be additionally configured to authenticate against other means as well.

Social authentication

Thanks to `python-social-auth`, Weblate support authentication using many third party services such as Facebook, GitHub, Google or Bitbucket.

Please check their documentation for generic configuration instructions:

<http://psa.matiasaguirre.net/docs/configuration/django.html>

Примечание: By default, Weblate relies on third-party authentication services to provide validated email address, in case some of services you want to use do not support this, please remove `social.pipeline.social_auth.associate_by_email` from `SOCIAL_AUTH_PIPELINE` settings.

Enabling individual backends is quite easy, it's just a matter of adding entry to `AUTHENTICATION_BACKENDS` setting and possibly adding keys needed for given authentication. Please note that some backends do not provide user email by default, you have to request it explicitly, otherwise Weblate will not be able to properly credit users contributions.

OpenID authentication

For OpenID based services it's usually just a matter of enabling them. Following section enables OpenID authentication for OpenSUSE, Fedora and Ubuntu:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social.backends.email.EmailAuth',
    'social.backends.suse.OpenSUSEOpenId',
    'social.backends.ubuntu.UbuntuOpenId',
    'social.backends.fedora.FedoraOpenId',
    'weblate.accounts.auth.WeblateUserBackend',
)
```

GitHub authentication

You need to register application on GitHub and then tell Weblate all the secrets:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social.backends.github.GithubOAuth2',
    'social.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']
```

См.также:

<http://psa.matiasaguirre.net/docs/backends/index.html>

Google OAuth2

For using Google OAuth2, you need to register application on <https://console.developers.google.com/> and enable Google+ API.

The redirect URL is `https://WEBLATE_SERVER/accounts/complete/google-oauth2/`

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social.backends.google.GoogleOAuth2',
    'social.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'
```


Facebook OAuth2

As usual with OAuth2 services, you need to register your application with Facebook. Once this is done, you can configure Weblate to use it:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social.backends.facebook.FacebookOAuth2',
    'social.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
SOCIAL_AUTH_FACEBOOK_KEY = 'key'
SOCIAL_AUTH_FACEBOOK_SECRET = 'secret'
SOCIAL_AUTH_FACEBOOK_SCOPE = ['email', 'public_profile']
```

LDAP authentication

LDAP authentication can be best achieved using *django-auth-ldap* package. You can install it by usual means:

```
# Using PyPI
pip install django-auth-ldap

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

Once you have the package installed, you can hook it to 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 different DN
# like:
# AUTH_LDAP_USER_DN_TEMPLATE = 'ou=users,dc=example,dc=com'

# List of attributes to import from LDAP on login
# Weblate stores full user name in the first_name attribute
AUTH_LDAP_USER_ATTR_MAP = {
    'first_name': 'name',
    'email': 'mail',
}
```

См.также:

<http://pythonhosted.org/django-auth-ldap/>

4.6 Access control

Weblate uses privileges system based on Django. The default setup (after you run `setupgroups`) consists of three groups *Guests*, *Users*, *Owners* and *Managers* which have privileges as described above. All new users are automatically added to *Users* group. The *Guests* groups is used for not logged in users. The *Owners* groups adds special privileges to users owning a project.

Basically *Users* are meant as regular translators and *Managers* for developers who need more control over the translation - they can force committing changes to VCS, push changes upstream (if Weblate is configured to do so) or disable translation (eg. when there are some major changes happening upstream).

To customize this setup, it is recommended to remove privileges from *Users* group and create additional groups with finer privileges (eg. *Translators* group, which will be allowed to save translations and manage suggestions) and add selected users to this group. You can do all this from Django admin interface.

To completely lock down your Weblate installation you can use `LOGIN_REQUIRED_URLS` for forcing users to login and `REGISTRATION_OPEN` for disallowing new registrations.

4.6.1 Extra privileges

Weblate defines following extra privileges:

Can upload translation [Users, Managers, Owners] Uploading of translation files.

Can overwrite with translation upload [Users, Managers, Owners] Overwriting existing translations by uploading translation file.

Can define author of translation upload [Managers, Owners] Allows to define custom authorship when uploading translation file.

Can force committing of translation [Managers, Owners] Can force VCS commit in the web interface.

Can see VCS repository URL [Users, Managers, Owners, Guests] Can see VCS repository URL inside Weblate

Can update translation from VCS [Managers, Owners] Can force VCS pull in the web interface.

Can push translations to remote VCS [Managers, Owners] Can force VCS push in the web interface.

Can do automatic translation using other project strings [Managers, Owners] Can do automatic translation based on strings from other components

Can lock whole translation project [Managers, Owners] Can lock translation for updates, useful while doing some major changes in the project.

Can reset translations to match remote VCS [Managers, Owners] Can reset VCS repository to match remote VCS.

Can save translation [Users, Managers, Owners] Can save translation (might be disabled with *Suggestion voting*).

Can accept suggestion [Users, Managers, Owners] Can accept suggestion (might be disabled with *Suggestion voting*).

Can delete suggestion [Users, Managers, Owners] Can delete suggestion (might be disabled with *Suggestion voting*).

Can delete comment [Managers, Owners] Can delete comment.

Can vote for suggestion [Users, Managers, Owners] Can vote for suggestion (see *Suggestion voting*).

Can override suggestion state [Managers, Owners] Can save translation, accept or delete suggestion when automatic accepting by voting for suggestions is enabled (see *Suggestion voting*).

Can import dictionary [Users, Managers, Owners] Can import dictionary from translation file.

Can add dictionary [Users, Managers, Owners] Can add dictionary entries.

Can change dictionary [Users, Managers, Owners] Can change dictionary entries.

Can delete dictionary [Users, Managers, Owners] Can delete dictionary entries.

Can lock translation for translating [Users, Managers, Owners] Can lock translation while translating (see *Translation locking*).

Can add suggestion [Users, Managers, Owners, Guests] Can add new suggestions.

Can use machine translation [Users, Managers, Owners] Can use machine translations (see *Machine translation setup*).

Can manage ACL rules for a project [Managers, Owners] Can add users to ACL controlled projects (see *Per project access control*).

Can edit priority [Managers, Owners] Can adjust source string priority

Can edit check flags [Managers, Owners] Can adjust source string check flags

Can download changes [Managers, Owners] Can download changes in a CSV format.

4.6.2 Per project access control

Добавлено в версии 1.4: This feature is available since Weblate 1.4.

Примечание: By enabling ACL, all users are prohibited to access anything within given project unless you add them the permission to do that.

Additionally you can limit users access to individual projects. This feature is enabled by *Enable ACL* at Project configuration. Once you enable this, users without specific privilege (*trans / project / Can access project NAME*) can not access this project. An user group with same name as a project is also automatically created to ease you management of the privilege.

To allow access to this project, you have to add the privilege to do so either directly to given user or group of users in Django admin interface. Or using user management on project page as described in *Managing per project access control*.

См.также:

<https://docs.djangoproject.com/en/stable/topics/auth/default/#auth-admin>

4.6.3 Managing users and groups

All users and groups can be managed using Django admin interface, which is available under `/admin/` URL.

Managing per project access control

Примечание: This feature only works for ACL controlled projects, see *Per project access control*.

Users with *Can manage ACL rules for a project* privilege (see *Access control*) can also manage users in projects with access control enabled on the project page. You can add or remove users to the project or make them owners.

The user management is available in *Tools* menu of a project:

Hosted Weblate

Website

translated 100%

Overview

History

Activity

Tools

Share

Users

Username	Full name	
nijel Owner	Michal Čihař	<div><div>Remove</div><div>Revoke ownership</div></div>
lem9 Owner	Marc Delisle marc@infomarc.info	<div><div>Remove</div><div>Revoke ownership</div></div>
pavolzibrita	Pavol Zibrita pavol.zibrita@gmail.com	<div><div>Remove</div><div>Make owner</div></div>

Data exports

Commit

Repository maintenance

Failing checks

Ignored checks

Manage users

Add new user

User to add

Please provide username or email. User needs to already have an active account in Weblate.

Add

См.также:

Per project access control

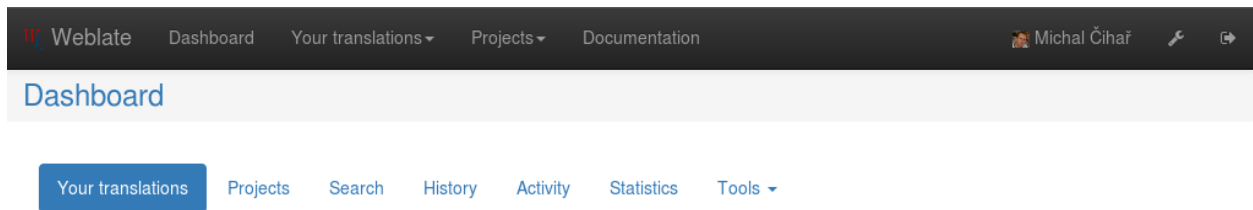
4.7 Translation projects

4.7.1 Translation organization

Weblate organizes translatable content into tree like structure. The toplevel object is *Project configuration*, which should hold all translations which belong together (for example translation of an application in several versions and/or documentation). On the next level, there is *Component configuration*, which is actually the component to translate. Here you define VCS repository to use and mask of files to translate. Bellow *Component configuration* there are individual translations, which are handled automatically by Weblate as the translation files (matching mask defined in *Component configuration*) appear in VCS repository.

4.7.2 Administration

Administration of Weblate is done through standard Django admin interface, which is available under `/admin/` URL. Once logged in as user with proper privileges, you can access it using wrench icon in top navigation:



Here you can manage objects stored in the database, such as users, translations and other settings:

Django administration

Welcome, **Michal Čihař**. [View site](#) / [Documentation](#) / [Change password](#) / [Log out](#)

Site administration

Reports		
Status of repositories		
SSH keys		
Performance report		
Accounts		
Profiles	+ Add	Change
Verified emails	+ Add	Change
Authentication and Authorization		
Groups	+ Add	Change
Users	+ Add	Change
Python Social Auth		
Associations	+ Add	Change
Nonces	+ Add	Change
User social auths	+ Add	Change
Sites		
Sites	+ Add	Change
Weblate languages		
Languages	+ Add	Change
Weblate translations		
Advertisements	+ Add	Change
Components	+ Add	Change
Projects	+ Add	Change
Whiteboard messages	+ Add	Change

Recent Actions
My Actions

- [iptv/PHP](#)
Component
- [+ iptv/PHP](#)
Component
- [+ iptv](#)
Project
- [127.0.0.1:8888](#)
Site
- [✗ Colognean](#)
Language
- [+ Advertisement object](#)
Advertisement
- [✗ nijel4751bd8a87334443](#)
User
- [✗ nijeladecac24b8cc4e9c](#)
User
- [✗ nijelb129e2b3f6544df4](#)
User
- [✗ nijel7e9d081948914b7a](#)
User

In the *Reports* section you can check status of your site, tweak it for *Production setup* or manage SSH keys to access *Private repositories*.

All sections below you can manage database objects. The most interesting one is probably *Weblate translations*, where you can manage translatable projects, see *Project configuration* and *Component configuration*.

4.7.3 Adding new components

All translation components need to be available as VCS repositories and are organized as project/component structure.


Weblate supports wide range of translation formats (both bilingual and monolingua) supported by translate toolkit, see *Supported formats* for more information.

Adding project

First you have to add project, which will serve as container for all components. Usually you create one project for one piece of software or book (see *Project configuration* for information on individual parameters):

Add Project

Required fields are marked as **bold**, you can find more information in the [documentation](#).

Project name:	<input type="text" value="Weblate"/>
	Name to display
URL slug:	<input type="text" value="weblate"/>
	Name used in URLs and file names.
Project website:	<input type="text" value="http://weblate.org/"/>
	Main website of translated project.
Mailing list:	<input type="text" value="weblate@lists.cihar.com"/>
	Mailing list for translators.
Translation instructions:	<input type="text" value="http://weblate.org/contribute/"/>
	URL with instructions for translators.
<input checked="" type="checkbox"/> Push on commit	
	Whether the repository should be pushed upstream on every commit.
<input checked="" type="checkbox"/> Set Translation-Team header	
	Whether the Translation-Team in file headers should be updated by Weblate.
<input type="checkbox"/> Enable ACL	
	Whether to enable ACL for this project, please check documentation before enabling this.
<input checked="" type="checkbox"/> Enable hooks	
	Whether to allow updating this repository by remote hooks.
Owner:	<input type="text" value="nijel"/> 
	Owner of the project.

См.также:

Project configuration

Bilingual components

Once you have added a project, you can add translation components to it (see *Component configuration* for information on individual parameters):

Django administration

Welcome, **Michal Čihař**. [Documentation](#) / [Change password](#) / [Log out](#)[Home](#) > [Weblate translations](#) > [Components](#) > Add Component

Add Component

Required fields are marked as **bold**, you can find more information in the [documentation](#).

Importing a new translation can take some time, please check [our documentation](#) for information on how to improve this.

Component name:	<input type="text" value="website"/> Name to display
URL slug:	<input type="text" value="website"/> Name used in URLs and file names.
Project:	<input type="text" value="Weblate"/>
Version control system:	<input type="text" value="Git"/> Version control system to use to access your repository with translations.
Source code repository:	<input type="text" value="git://github.com/nijel/weblate-web.git"/> URL of a repository, use weblate://project/component for sharing with other component.
Repository push URL:	<input type="text" value="git@github.com:nijel/weblate-web.git"/> URL of a push repository, pushing is disabled if empty.
Repository browser:	<input type="text" value="://github.com/nijel/weblate-web/blob/%(branch)s/%(file)s#L%(line)s"/> Link to repository browser, use %(branch)s for branch, %(file)s and %(line)s as filename and line placeholders.
Exported repository URL:	<input type="text" value="git://git.weblate.org/weblate-web.git"/> URL of a repository where users can fetch changes from Weblate
Source string bug report address:	<input type="text" value="weblate@lists.cihar.com"/> Email address where errors in source string will be reported, keep empty for no emails.
Repository branch:	<input type="text" value="master"/> Repository branch to translate
File mask:	<input type="text" value="locale/*/LC_MESSAGES/django.po"/> Path of files to translate, use * instead of language code, for example: po/*.po or locale/*/LC_MESSAGES/django.po.
Monolingual base language file:	<input type="text"/> Filename of translations base file, which contains all strings and their source; this is recommended to use for monolingual translation formats.
<input checked="" type="checkbox"/> Edit base file	Whether users will be able to edit base file for monolingual translations.
Base file for new translations:	<input type="text" value="locale/django.pot"/> Filename of file which is used for creating new translations. For Gettext choose .pot file.
File format:	<input type="text" value="Gettext PO file"/> Automatic detection might fail for some formats and is slightly slower.

4.7. Translation projects

Additional

commit file:

Additional file to include in commits; please check documentation for more details.

Post-update

См.также:

[Component configuration](#)

Monolingual components

For easier translating of monolingual formats, you should provide template file, which contains mapping of message IDs to source language (usually English) (see *[Component configuration](#)* for information on individual parameters):

Django administration

Welcome, **Michal Čihař**. [Documentation](#) / [Change password](#) / [Log out](#)[Home](#) > [Weblate translations](#) > [Components](#) > Add Component

Add Component

Required fields are marked as **bold**, you can find more information in the [documentation](#).

Importing a new translation can take some time, please check [our documentation](#) for information on how to improve this.

Component name:	<input type="text" value="iOS"/> Name to display
URL slug:	<input type="text" value="ios"/> Name used in URLs and file names.
Project:	<input type="text" value="OsmAnd"/>
Version control system:	<input type="text" value="Git"/> Version control system to use to access your repository with translations.
Source code repository:	<input type="text" value="https://github.com/osmandapp/OsmAnd-ios.git"/> URL of a repository, use weblate://project/component for sharing with other component.
Repository push URL:	<input type="text" value="git@github.com:/osmandapp/OsmAnd-ios.git"/> URL of a push repository, pushing is disabled if empty.
Repository browser:	<input type="text"/> Link to repository browser, use %(branch)s for branch, %(file)s and %(line)s as filename and line placeholders.
Exported repository URL:	<input type="text" value="git://git.weblate.org/osmand-ios.git"/> URL of a repository where users can fetch changes from Weblate
Source string bug report address:	<input type="text"/> Email address where errors in source string will be reported, keep empty for no emails.
Repository branch:	<input type="text" value="master"/> Repository branch to translate
File mask:	<input type="text" value="Resources/*.lproj/Localizable.strings"/> Path of files to translate, use * instead of language code, for example: po/*.po or locale/*/LC_MESSAGES/django.po.
Monolingual base language file:	<input type="text" value="Resources/en.lproj/Localizable.strings"/> Filename of translations base file, which contains all strings and their source; this is recommended to use for monolingual translation formats.
<input type="checkbox"/> Edit base file	Whether users will be able to edit base file for monolingual translations.
Base file for new translations:	<input type="text"/> Filename of file which is used for creating new translations. For Gettext choose .pot file.
File format:	<input type="text" value="OS X Strings (UTF-8)"/> Automatic detection might fail for some formats and is slightly slower.

4.7. Translation projects

Additional

commit file:

 Additional file to include in commits; please check documentation for more details.

Post-update

См.также:

Component configuration

4.7.4 Project configuration

To add new component to translate, you need to create translation project first. The project is sort of shelf, in which real translations are folded. All components in same project share suggestions and dictionary, also the translations are automatically propagated through the all component in single project (unless disabled in component configuration).

The project has only few attributes giving translators information about project:

Project website URL where translators can find more information about the project.

Mailing list Mailing list where translators can discuss or comment translations.

Translation instructions URL where you have more detailed instructions for translators.

Push on commit Whether any committed changes should be automatically pushed to upstream repository.

Set Translation-Team header Whether Weblate should manage Translation-Team header (this is *GNU Gettext* only feature right now).

Enable ACL Enable per project access control, see *Per project access control* for more details.

Enable hooks Whether unauthenticated *Notification hooks* will be enabled for this repository.

Owner You can also configure project owner, who will always get important notifications about project and will have additional privileges to control translations within this project (see *Access control*).

Adjusting interaction

There are also additional features which you can control, like automatic pushing of changes (see also *Pushing changes*) or maintaining of Translation-Team header.

4.7.5 Component configuration

Component is real component for translating. You enter VCS repository location and file mask which files to translate and Weblate automatically fetches the VCS and finds all matching translatable files.

Should the language definition for translation be missing, empty definition is created and named as «cs_CZ (generated)». You should adjust the definition and report this back to Weblate authors so that missing language can be included in next release.

The component contains all important parameters for working with VCS and getting translations out of it:

Source code repository VCS repository used to pull changes.

This can be either real VCS URL or `weblate://project/component` indicating that the repository should be shared with another component.

Repository push URL Repository URL used for pushing, this is completely optional and push support will be disabled when this is empty.

Примечание: Weblate currently does not support HTTP authentication on push URLs

Repository browser URL of repository browser to display source files (location where messages are used). When empty no such links will be generated.

For example on GitHub, you would use something like `https://github.com/nijel/weblate-hello/blob/%(branch)s/%(file)s#L%(line)s`.

Exported repository URL URL where changes made by Weblate are exported. This is important when *Continuous translation* is not used or when there is need to manually merge changes.

Repository branch Which branch to checkout from the VCS and where to look for translations.

File mask Mask of files to translate including path. It should include one `*` replacing language code (Weblate can handle language names as well, but it is recommended to use ISO 639-1 language codes). In case your repository contains more than one translation files (eg. more Gettext domains), you need to create separate component for each.

For example `po/*.po` or `locale/*/LC_MESSAGES/django.po`.

Monolingual base language file Base file containing strings definition for *Monolingual components*.

Edit base file Whether to allow editing of base file for *Monolingual components*.

Base file for new translations Base file used to generate new translations, eg. `.pot` file with Gettext.

File format Translation file format, see also *Supported formats*.

Source string bug report address Email address used for reporting upstream bugs. This address will also receive notification about any source string comments made in Weblate.

Locked You can lock the translation to prevent updates by users.

Allow translation propagation You can disable 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 same string used.

It's usually good idea to disable this for monolingual translations unless you are using same IDs across whole project.

Post-update script One of scripts defined in *POST_UPDATE_SCRIPTS* which is executed after receiving update. This can be used to update the translation files.

Pre-commit script One of scripts defined in *PRE_COMMIT_SCRIPTS* which is executed before commit. This can be used to generate some metadata about translation or to generate binary form of a translation.

Post-commit script One of scripts defined in *POST_COMMIT_SCRIPTS* which is executed after commit. This can be used to notify external parties about the change.

Post-push script One of scripts defined in *POST_PUSH_SCRIPTS* which is executed after push to remote repository. This can be used to generate notify external parties about the change in repository (i.e. create pull request).

Post-add script One of scripts defined in *POST_ADD_SCRIPTS* which is executed when new translation has been added. This can be used to adjust additional files in the repository when adding new translation.

Additional commit files Additional files to include in the commit (separated by newline), usually this one is generated by the pre commit or post add scripts described above.

Supply the `%(language)s` in the path like this: `path/to/additinal/%(language)s_file.example`

Save translation history Whether to store history of translation changes in database.

Suggestion voting Enable voting for suggestions, see *Suggestion voting*.

Autoaccept suggestions Automatically accept voted suggestions, see *Suggestion voting*.

Quality checks flags Additional flags to pass to quality checks, see *Customizing checks*.

Translation license License of this translation.

License URL URL where users can find full text of a license.

New language How to handle requests for creating new languages. Please note that availability of choices depends on the file format, see *Supported formats*.

Merge style You can configure how the updates from upstream repository are handled. This might not be supported for some VCS. See *Merge or rebase* for more details.

Commit message Message used when committing translation, see *Commit message formatting*.

Committer name Name of committer used on Weblate commits, the author will be always the real translator. On some VCS this might be not supported. Default value can be changed by *DEFAULT_COMMITTER_NAME*.

Committer email Email of committer used on Weblate commits, the author will be always the real translator. On some VCS this might be not supported. Default value can be changed by *DEFAULT_COMMITTER_EMAIL*.

Language filter Regular expression which is used to filter translation when scanning for file mask. This can be used to limit list of languages managed by Weblate (eg. `^(cs|de|es)$` will include only those there languages. Please note that you need to list language codes as they appear in the filename.

Commit message formatting

The commit message on each commit Weblate does, it can use following format strings in the message:

`%(language)s` Language code

`%(language_name)s` Language name

`%(component)s` Component name

`%(project)s` Project name

`%(total)s` Total strings count

`%(fuzzy)s` Fuzzy strings count

`%(fuzzy_percent)s` Fuzzy strings percent

`%(translated)s` Translated strings count

`%(translated_percent)s` Translated strings percent

См.также:

Does Weblate support other VCS than Git and Mercurial?, Processing repository with scripts

4.7.6 Importing speed

Fetching VCS repository and importing translations to Weblate can be lengthy process depending on size of your translations. Here are some tips to improve this situation:

Clone Git repository in advance

You can put in place Git repository which will be used by Weblate. The repositories are stored in `vcs` directory in path defined by *DATA_DIR* in `settings.py` in `<project>/<component>` directories.

This can be especially useful if you already have local clone of this repository and you can use `--reference` option while cloning:

```
git clone \
  --reference /path/to/checkout \
  git://github.com/nijel/weblate.git \
  weblate/repos/project/component
```

Optimize configuration

The default configuration is useful for testing and debugging Weblate, while for production setup, you should do some adjustments. Many of them have quite big impact on performance. Please check [Production setup](#) for more details, especially:

- *Enable indexing offloading*
- *Enable caching*
- *Use powerful database engine*
- *Disable debug mode*

Disable not needed checks

Some quality checks can be quite expensive and if you don't need them, they can save you some time during import. See [CHECK_LIST](#) for more information how to configure this.

4.7.7 Automatic creation of components

In case you have project with dozen of po files, you might want to import all at once. This can be achieved using `import_project`.

First you need to create project which will contain all components and then it's just a matter of running `import_project`.

См.также:

Management commands

4.7.8 Accessing repositories

Private repositories

In case you want Weblate to access private repository it needs to get to it somehow. Most frequently used method here is based on SSH. To have access to such repository, you generate SSH key for Weblate and authorize it to access the repository.

You also need to verify SSH host keys of servers you are going to access.

You can generate or display key currently used by Weblate in the admin interface (follow *SSH keys* link on main admin page).

If you are trying to connect to a GitHub repository either use the SSH address (eg. `git@github.com:nijel/weblate.git`) or generate personal access token (see [Creating an access token for command-line use](#)) and include it in the URL. The full URL should look like `https://user:your_access_token@github.com/nijel/weblate.git`.

On GitHub, you can add the key to only one repository. If you plan to access more of them, it might be better to create separate user for that, assign him Weblate's SSH key and grant him access to all repositories.

Примечание: The keys need to be without password to make it work, so be sure they are well protected against malicious usage.

Using proxy

If you need to access http/https VCS repositories using a proxy server, you need to configure VCS to use it.

This can be configured using the `http_proxy`, `https_proxy`, and `all_proxy` environment variables (check cURL documentation for more details) or by enforcing it in VCS configuration, for example:

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

Примечание: The proxy setting needs to be done in context which is used to execute Weblate. For the environment it should be set for both server and cron jobs. The VCS configuration has to be set for the user which is running Weblate.

См.также:

<http://curl.haxx.se/docs/manpage.html>, <http://git-scm.com/docs/git-config>

4.7.9 Fulltext search

Fulltext search is based on Whoosh. You can either allow Weblate to directly update index on every change to content or offload this to separate process by `OFFLOAD_INDEXING`.

The first approach (immediate updates) allows more up to date index, but suffers locking issues in some setup (eg. Apache's `mod_wsgi`) and produces more fragmented index.

Offloaded indexing is always better choice for production setup - it only marks which items need to be reindexed and you need to schedule background process (`update_index`) to update index. This leads to faster response of the site and less fragmented index with cost that it might be slightly outdated.

См.также:

update_index, OFFLOAD_INDEXING, Fulltext search is too slow, I get «Lock Error» quite often while translating, Rebuilding index has failed with «No space left on device»

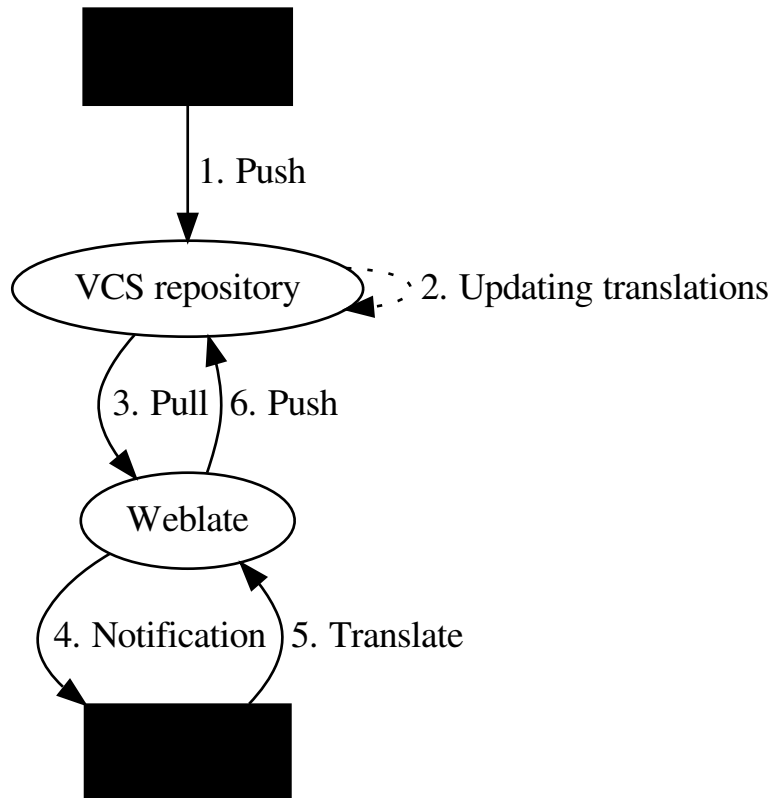
4.8 Continuous translation

Weblate provides you great infrastructure for translation to closely follow your development. This way translators can work on translations whole time and are not forced to translate huge amount of new texts before release.

The complete process can be described in following steps:

1. Developers make some 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 shows 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 will be notified based on their subscription settings.
5. Translators make translations using Weblate web interface.
6. Once translators are done, Weblate commits the changes to the local repository (see [Lazy commits](#)) and pushes them back if it has permissions to do that (see [Pushing changes](#)).



4.8.1 Updating repositories

You should set up some way how backend repositories are updated from their source. You can either use hooks (see [Notification hooks](#)) or just regularly run `updategit --all`.

Whenever Weblate updates the repository, the *Post-update script* hooks are executed.

With Gettext po files, you might be often bitten by conflict in PO file headers. To avoid it, you can use shipped merge driver (`examples/git-merge-gettext-po`). To use it just put following configuration to your `.gitconfig`:

```
[merge "merge-gettext-po"]
name = merge driver for gettext po files
driver = /path/to/weblate/examples/git-merge-gettext-po %0 %A %B
```

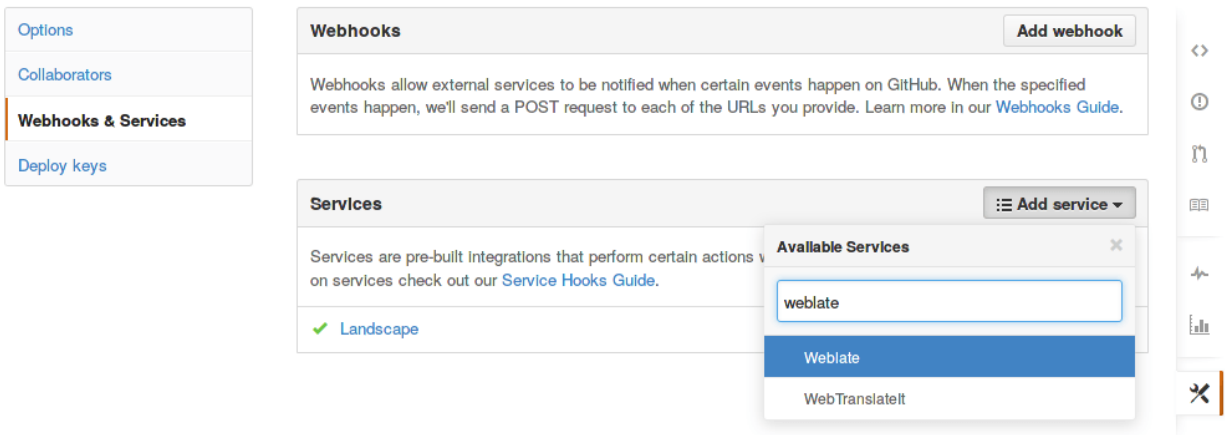
And enable its use by defining proper attributes in given repository (eg. in `.git/info/attributes`):

```
*.po merge=merge-gettext-po
```

Примечание: This merge driver assumes the changes in POT files always are done in branch we're trying to merge.

Automatically receiving changes from GitHub

Weblate comes with native support for GitHub. To receive notifications on every push to GitHub repository, you just need to enable Weblate Service in the repository settings (*Webhooks & Services*) as shown on the image below:



To set the base URL of your Weblate installation (for example `https://hosted.weblate.org`) and Weblate will be notified about every push to GitHub repository:

Options
Collaborators
Webhooks & Services
Deploy keys

Services / Manage Weblate

Test service

This service will notify Weblate about pushes to your repository. Weblate will then refresh updated translations and merge them with your changes.

More info: <http://weblate.org/>

Install Notes

You need to have enabled hooks in your Weblate installation (this is default) see documentation for more information: <https://weblate.readthedocs.org/en/latest/api.html#notification-hooks>

Url

https://hosted.weblate.org

☒ Active
 We will run this service when an event is triggered.

Update service

Delete service

<>
ⓘ
🔗
📄
📈
🔧

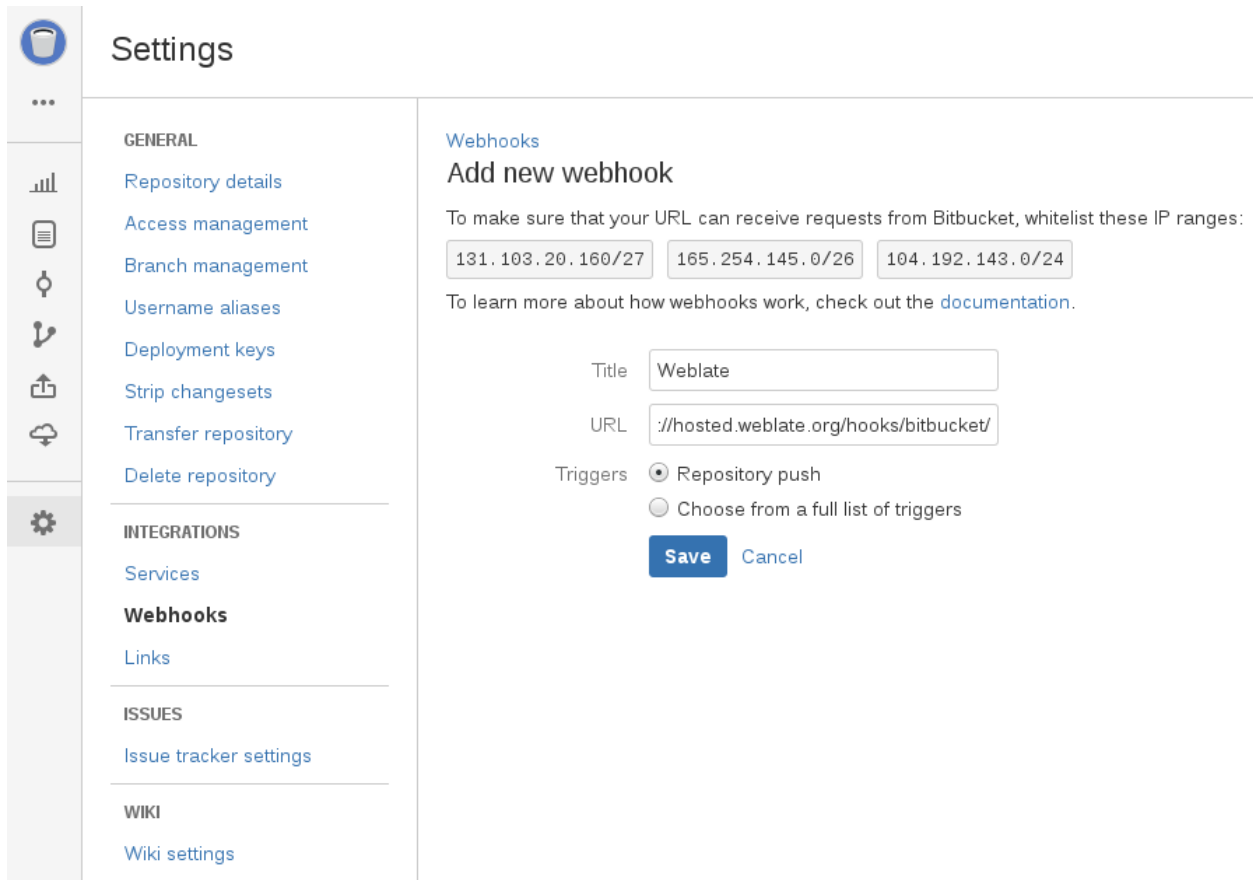
You can also use generic *Webhook*, in that case the *Payload URL* would have to be full path to the handler, for example `https://hosted.weblate.org/hooks/github/`.

См.также:

POST /hooks/github/, Pushing changes from Hosted Weblate

Automatically receiving changes from Bitbucket

Weblate has support for Bitbucket webhooks, all you need to do is add webhook which triggers on repository push with destination to `/hooks/bitbucket/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/bitbucket/`).



The screenshot shows the 'Settings' page in Weblate. On the left is a sidebar with icons for various settings categories: GENERAL, INTEGRATIONS, ISSUES, and WIKI. The 'Webhooks' section is highlighted under 'INTEGRATIONS'. The main content area is titled 'Webhooks' and 'Add new webhook'. It includes a note about whitelisting IP ranges for Bitbucket requests, with three input fields containing the ranges: 131.103.20.160/27, 165.254.145.0/26, and 104.192.143.0/24. Below this is a link to the documentation. The 'Title' field is set to 'Weblate' and the 'URL' field is set to '://hosted.weblate.org/hooks/bitbucket/'. Under 'Triggers', the 'Repository push' option is selected. At the bottom are 'Save' and 'Cancel' buttons.

См.также:

POST /hooks/bitbucket/, Pushing changes from Hosted Weblate

Automatically receiving changes from GitLab

Weblate has support for GitLab hooks, all you need to do is add project web hook with destination to `/hooks/gitlab/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/gitlab/`).

См.также:

POST /hooks/gitlab/, Pushing changes from Hosted Weblate

4.8.2 Pushing changes

Each project can have configured push URL and in such case Weblate offers button to push changes to remote repository in web interface.

If you are using SSH to push, you will need to have a key without a passphrase (or use ssh-agent for Django) and the remote server needs to be verified by you via the admin interface first, otherwise pushing will fail.

Примечание: You can also enable automatic pushing changes on commit, this can be done in project configuration.

См.также:

Private repositories for setting up SSH keys

Pushing changes to GitHub as pull request

Примечание: This feature is currently not available on Hosted Weblate due to technical limitations. See *Pushing changes from Hosted Weblate* for available options.

If you are translating a project that's hosted on GitHub and don't want to push translations to the repository, you can have them sent as a pull request instead.

You need to configure the *hub* command line tool and set *GITHUB_USERNAME* for this to work.

См.также:

GITHUB_USERNAME, *Setting up hub* for configuration instructions

Pushing changes from Hosted Weblate

For Hosted Weblate there is dedicated push user registered on GitHub, Bitbucket and GitLab (with username *weblate* and named *Weblate push user*). You need to add this user as a collaborator and give him permissions to push to your repository. Let us know when you've done so and we will enable pushing changes from Hosted Weblate for you.

4.8.3 Merge or rebase

By default Weblate merges upstream repository into its own. This is safest way in case you also access underlying repository by other means. In case you don't need this, you can enable rebasing of changes on upstream, what will produce history with less merge commits.

Примечание: Rebasing can cause you troubles in case of complicated merges, so carefully consider whether you want to enable them or not.

4.8.4 Interacting with others

Weblate makes it easy to interact with others using its API.

См.также:

Weblate's Web API

4.8.5 Lazy commits

Default behaviour (configured by *LAZY_COMMITS*) of Weblate is to group commits from same author into one if possible. This heavily reduces number of commits, however you might need to explicitly tell to do the commits in case you want to get VCS repository in sync, eg. for merge (this is by default allowed for Managers group, see *Access control*).

The changes are in this mode committed once any of following conditions is fulfilled:

- somebody else works on the translation
- merge from upstream occurs
- import of translation happens
- translation for a language is completed
- explicit commit is requested

You can also additionally set a cron job to commit pending changes after some delay, see [commit_pending](#) and [Running maintenance tasks](#).

4.8.6 Processing repository with scripts

You can customize way how Weblate manipulates with repository by set of scripts. These include *Post-update script*, *Pre-commit script*, *Post-commit script*, *Post-add script* and *Post-push script* and are briefly described in [Component configuration](#).

Their naming quite clearly tells when given script is executed. The commit related scripts always get one parameter with full path to the translation file which has been changed.

The script is executed with the current directory set to root of VCS repository for given component.

Additionally following environment variables are available:

WL_VCS
Used version control system.

WL_REPO
Upstream repository URL.

WL_PATH
Absolute path to VCS repository.

WL_FILEMASK
File mask for current component.

WL_FILE_FORMAT
File format used in current component.

WL_LANGUAGE
Language of currently processed translation (not available for component level hooks).

См.также:

[POST_UPDATE_SCRIPTS](#), [PRE_COMMIT_SCRIPTS](#), [POST_COMMIT_SCRIPTS](#), [POST_PUSH_SCRIPTS](#), [Component configuration](#)

Pre commit processing of translations

In many cases you might want to automatically do some changes to translation before it is committed to the repository. The pre commit script is exactly the place to achieve this.

Before using any scripts, you need to list them in [PRE_COMMIT_SCRIPTS](#) configuration variable. Then you can enable them at [Component configuration](#) configuration as *Pre commit script*.

It is passed single parameter consisting of file name of current translation.

The script can also generate additional file to be included in the commit. This can be configured as *Extra commit file* at [Component configuration](#) configuration. You can use following format strings in the filename:

`%(language)s` Language code

Example - generating mo files in repository

Allow usage of the hook in the configuration

```
PRE_COMMIT_SCRIPTS = (
    '/usr/share/weblate/examples/hook-generate-mo',
)
```

To enable it, choose now *hook-generate-mo* as *Pre commit script*. You will also want to add path to generated files to be included in VCS commit, for example `po/%(language)s.mo` as *Extra commit file*.

You can find more example scripts in `examples` folder within Weblate sources, their name start with `hook-`.

4.9 Translation process

4.9.1 Suggestion voting

Добавлено в версии 1.6: This feature is available since Weblate 1.6.

In default Weblate setup, everybody can add suggestions and logged in users can accept them. You might however want to have more eyes on the translation and require more people to accept them. This can be achieved by suggestion voting. You can enable this on *Component configuration* configuration by *Suggestion voting* and *Autoaccept suggestions*. The first one enables voting feature, while the latter allows you to configure threshold at which suggestion will gets automatically accepted (this includes own vote from suggesting user).

Примечание: Once you enable automatic accepting, normal users lose privilege to directly save translations or accept suggestions. This can be overridden by *Can override suggestion state* privilege (see *Access control*).

You can combine these with *Access control* into one of following setups:

- Users can suggest and vote for suggestions, limited group controls what is accepted - enable voting but not automatic accepting and remove privilege from users to save translations.
- Users can suggest and vote for suggestions, which get automatically accepted once defined number of users agree on this - enable voting and set desired number of votes for automatic accepting.
- Optional voting for suggestions - you can also only enable voting and in this case it can be optionally used by users when they are not sure about translation (they can suggest more of them).

4.9.2 Translation locking

To improve collaboration, it is good to prevent duplicate effort on translation. To achieve this, translation can be locked for single translator. This can be either done manually on translation page or is done automatically when somebody starts to work on translation. The automatic locking needs to be enabled using *AUTO_LOCK*.

The automatic lock is valid for *AUTO_LOCK_TIME* seconds and is automatically extended on every translation made and while user has opened translation page.

User can also explicitly lock translation for *LOCK_TIME* seconds.

4.10 Checks and fixups

4.10.1 Custom automatic fixups

You can also implement own automatic fixup in addition to standard ones and include them in `AUTOFIX_LIST`.

The automatic fixes are powerful, but can also cause damage, be careful when writing one.

For example following automatic fixup would replace every occurrence of string `foo` in translation with `bar`:

```
# -*- coding: utf-8 -*-
#
# Copyright © 2012 - 2015 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <http://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 <http://www.gnu.org/licenses/>.
#

from weblate.trans.autofixes.base import AutoFix
from django.utils.translation import ugettext_lazy as _

class ReplaceFooWithBar(AutoFix):
    '''
    Replaces foo with bar.
    '''

    name = _('Foobar')

    def fix_single_target(self, target, source, unit):
        if 'foo' in target:
            return target.replace('foo', 'bar'), True
        return target, False
```

4.10.2 Customizing checks

Fine tuning existing checks

You can fine tune checks for each source string (in source strings review) or in the *Component configuration* (*Quality checks flags*), here is current list of flags accepted:

`rst-text` Treat text as RST document, affects *Unchanged translation*.

`python-format`, `c-format`, `php-format`, `python-brace-format`, `javascript-format` Treats all string like format strings, affects *Format strings*, *Format strings*, *Format strings*, *Format strings*, *Format strings*, *Unchanged translation*.

`ignore-end-space` Skip the «Trailing space» quality check.

`ignore-inconsistent` Skip the «Inconsistent» quality check.

`ignore-begin-newline` Skip the «Starting newline» quality check.

`ignore-zero-width-space` Skip the «Zero-width space» quality check.

`ignore-escaped-newline` Skip the «Mismatched n» quality check.

`ignore-same` Skip the «Unchanged translation» quality check.

`ignore-end-question` Skip the «Trailing question» quality check.

`ignore-end-ellipsis` Skip the «Trailing ellipsis» quality check.

`ignore-ellipsis` Skip the «Ellipsis» quality check.

`ignore-python-brace-format` Skip the «Python brace format» quality check.

`ignore-end-newline` Skip the «Trailing newline» quality check.

`ignore-c-format` Skip the «C format» quality check.

`ignore-javascript-format` Skip the «Javascript format» quality check.

`ignore-optional-plural` Skip the «Optional plural» quality check.

`ignore-end-exclamation` Skip the «Trailing exclamation» quality check.

`ignore-end-colon` Skip the «Trailing colon» quality check.

`ignore-xml-tags` Skip the «XML tags mismatch» quality check.

`ignore-python-format` Skip the «Python format» quality check.

`ignore-plurals` Skip the «Missing plurals» quality check.

`ignore-begin-space` Skip the «Starting spaces» quality check.

`ignore-bbcode` Skip the «Mismatched BBcode» quality check.

`ignore-multiple-failures` Skip the «Multiple failing checks» quality check.

`ignore-php-format` Skip the «PHP format» quality check.

`ignore-end-stop` Skip the «Trailing stop» quality check.

Примечание: Generally the rule is named `ignore-*` for any check, using its identifier, so you can use this even for your custom checks.

These flags are understood both in *Component configuration* settings, per source string settings and in translation file itself (eg. in GNU Gettext).

Writing own checks

Weblate comes with wide range of quality checks (see *Quality checks*), though they might not 100% cover all you want to check. The list of performed checks can be adjusted using *CHECK_LIST* and you can also add custom checks. All you need to do is to subclass `trans.checks.Check`, set few attributes and implement

either `check` or `check_single` methods (first one if you want to deal with plurals in your code, the latter one does this for you). You will find below some examples.

Checking translation text does not contain «foo»

This is pretty simple check which just checks whether translation does not contain string «foo».

```
# -*- coding: utf-8 -*-
#
# Copyright © 2012 - 2015 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <http://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 <http://www.gnu.org/licenses/>.
#
'''
Simple quality check example.
'''

from weblate.trans.checks.base import TargetCheck
from django.utils.translation import ugettext_lazy as _

class FooCheck(TargetCheck):

    # Used as identifier for check, should be unique
    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, cache_slot):
        return 'foo' in target
```

Checking Czech translation text plurals differ

Check using language information to verify that two plural forms in Czech language are not same.

```
# -*- coding: utf-8 -*-
#
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Copyright © 2012 - 2015 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <http://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 <http://www.gnu.org/licenses/>.
#
'''
Quality check example for Czech plurals.
'''

from weblate.trans.checks.base import TargetCheck
from django.utils.translation import ugettext_lazy as _

class PluralCzechCheck(TargetCheck):

    # Used as identifier for check, should be unique
    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, cache_slot):
        '''
        We don't check target strings here.
        '''
        return False
```

4.11 Machine translation

4.11.1 Machine translation setup

Weblate has builtin support for several machine translation services and it's up to administrator to enable them. The services have different terms of use, so please check whether you are allowed to use them before

enabling in Weblate. The individual services are enabled using `MACHINE_TRANSLATION_SERVICES`.
The source language can be configured by `SOURCE_LANGUAGE` and is shared for all translations within Weblate.

Amagama

Special installation of `tmserver` run by Virtaal authors.

To enable this service, add `trans.machine.tmserver.AmagamaTranslation` to `MACHINE_TRANSLATION_SERVICES`.

См.также:

<http://docs.translatehouse.org/projects/virtaal/en/latest/amagama.html>

Apertium

A free/open-source machine translation platform providing translation to limited set of languages.

You should get API key from them, otherwise number of requests is rate limited.

To enable this service, add `trans.machine.apertium.ApertiumTranslation` to `MACHINE_TRANSLATION_SERVICES`.

См.также:

`MT_APERTIUM_KEY`, <https://www.apertium.org/>

Glosbe

Free dictionary and translation memory for almost every living language.

API is free to use, regarding indicated data source license. There is a limit of call that may be done from one IP in fixed period of time, to prevent from abuse.

To enable this service, add `trans.machine.glosbe.GlosbeTranslation` to `MACHINE_TRANSLATION_SERVICES`.

См.также:

<https://glosbe.com/>

Google Translate

Machine translation service provided by Google.

This service uses Translation API and you need to obtain API key and enable billing on Google API console.

To enable this service, add `trans.machine.google.GoogleTranslation` to `MACHINE_TRANSLATION_SERVICES`.

См.также:

`MT_GOOGLE_KEY`, <https://cloud.google.com/translate/docs>

Google Web Translate

Machine translation service provided by Google.

Please note that this does not use official Translation API but rather web based translation interface.

To enable this service, add `trans.machine.google.GoogleWebTranslation` to *MACHINE_TRANSLATION_SERVICES*.

См.также:

<https://translate.google.com/>

Microsoft Translator

Machine translation service provided by Microsoft.

You need to register at Azure market and use Client ID and secret from there.

To enable this service, add `trans.machine.microsoft.MicrosoftTranslation` to *MACHINE_TRANSLATION_SERVICES*.

См.также:

MT_MICROSOFT_ID, *MT_MICROSOFT_SECRET*, <http://www.bing.com/translator/>, <https://datamarket.azure.com/developer/applications/>

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 contact email in *MT_MYMEMORY_EMAIL*. you can also ask them for more.

To enable this service, add `trans.machine.mymemory.MyMemoryTranslation` to *MACHINE_TRANSLATION_SERVICES*.

См.также:

MT_MYMEMORY_EMAIL, *MT_MYMEMORY_USER*, *MT_MYMEMORY_KEY*, <http://mymemory.translated.net/>

tmserver

You can run your own translation memory server which is bundled with Translate-toolkit and let Weblate talk to it. You can also use it with amaGama server, which is enhanced version of tmserver.

First you will want to import some data to the translation memory:

To enable this service, add `trans.machine.tmserver.TMServerTranslation` to *MACHINE_TRANSLATION_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
```

Now you can start tmserver to listen to your requests:

```
tmserver -d /var/lib/tm/db
```

And configure Weblate to talk to it:

```
MT_TMSERVER = 'http://localhost:8888/'
```

См.также:

`MT_TMSERVER`, <http://docs.translatehouse.org/projects/translate-toolkit/en/latest/commands/tmserver.html>, <http://amagama.translatehouse.org/>

Weblate

Weblate can be source of machine translation as well. There are two services to provide you results - one does exact search for string, the other one finds all similar strings.

First one is useful for full string translations, the second one for finding individual phrases or words to keep the translation consistent.

To enable these services, add `trans.machine.weblatetm.WeblateSimilarTranslation` (for similar string matching) and/or `trans.machine.weblatetm.WeblateTranslation` (for exact string matching) to `MACHINE_TRANSLATION_SERVICES`.

Примечание: For similarity matching, it is recommended to have Whoosh 2.5.2 or later, earlier versions can cause infinite looks under some occasions.

4.11.2 Custom machine translation

You can also implement own machine translation services using few lines of Python code. Following example implements translation to fixed list of languages using `dictionary` Python module:

```
# -*- coding: utf-8 -*-
#
# Copyright © 2012 - 2015 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <http://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 <http://www.gnu.org/licenses/>.
#
'''
Machine translation example.
'''

from weblate.trans.machine.base import MachineTranslation
import dictionary
```

(continues on next page)

(продолжение с предыдущей страницы)

```
class SampleTranslation(MachineTranslation):
    '''
    Sample machine translation interface.
    '''
    name = 'Sample'

    def download_languages(self):
        '''
        Returns list of languages your machine translation supports.
        '''
        return set(('cs',))

    def download_translations(self, language, text, unit, request):
        '''
        Returns tuple with translations.
        '''
        return [(t, 100, self.name, text) for t in dictionary.translate(text)]
```

You can list own class in `MACHINE_TRANSLATION_SERVICES` and Weblate will start using that.

4.12 Configuration

All settings are stored in `settings.py` (as usual for Django).

Примечание: After changing any of these settings, you need to restart Weblate. In case it is run as `mod_wsgi`, you need to restart Apache to reload the configuration.

См.также:

Please check also [Django's documentation](#) for parameters which configure Django itself.

4.12.1 ANONYMOUS_USER_NAME

User name of user for defining privileges of not logged in user.

См.также:

Access control

4.12.2 AUTO_LOCK

Enables automatic locking of translation when somebody is working on it.

См.также:

Translation locking

4.12.3 AUTO_LOCK_TIME

Time in seconds for how long the automatic lock for translation will be active.

См.также:

Translation locking

4.12.4 AUTOFIX_LIST

List of automatic fixups to apply when saving the message.

Available fixes:

`trans.autofixes.whitespace.SameBookendingWhitespace` Fixes up whitespace in beginning and end of the string to match source.

`trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis` Replaces trailing dots with ellipsis if source string has it.

`trans.autofixes.chars.RemoveZeroSpace` Removes zero width space char if source does not contain it.

For example you can enable only few of them:

```
AUTOFIX_LIST = (  
    'weblate.trans.autofixes.whitespace.SameBookendingWhitespace',  
    'weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis',  
)
```

См.также:

Automatic fixups, Custom automatic fixups

4.12.5 BACKGROUND_HOOKS

Whether to run hooks in background. This is generally recommended unless you are debugging.

4.12.6 CHECK_LIST

List of quality checks to perform on translation.

Some of the checks are not useful for all projects, so you are welcome to adjust list of performed on your installation.

For example you can enable only few of them:

```
CHECK_LIST = (  
    'weblate.trans.checks.same.SameCheck',  
    'weblate.trans.checks.chars.BeginNewlineCheck',  
    'weblate.trans.checks.chars.EndNewlineCheck',  
    'weblate.trans.checks.chars.BeginSpaceCheck',  
    'weblate.trans.checks.chars.EndSpaceCheck',  
    'weblate.trans.checks.chars.EndStopCheck',  
    'weblate.trans.checks.chars.EndColonCheck',  
    'weblate.trans.checks.chars.EndQuestionCheck',  
    'weblate.trans.checks.chars.EndExclamationCheck',  
    'weblate.trans.checks.chars.EndEllipsisCheck',  
    'weblate.trans.checks.format.PythonFormatCheck',  
    'weblate.trans.checks.format.PythonBraceFormatCheck',  
    'weblate.trans.checks.format.PHPFormatCheck',  
    'weblate.trans.checks.format.CFormatCheck',  
)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
'weblate.trans.checks.format.JavascriptFormatCheck',
'weblate.trans.checks.consistency.PluralsCheck',
'weblate.trans.checks.consistency.ConsistencyCheck',
'weblate.trans.checks.chars.NewlineCountingCheck',
'weblate.trans.checks.markup.BBCodeCheck',
'weblate.trans.checks.chars.ZeroWidthSpaceCheck',
'weblate.trans.checks.markup.XMLTagsCheck',
'weblate.trans.checks.source.OptionalPluralCheck',
'weblate.trans.checks.source.EllipsisCheck',
'weblate.trans.checks.source.MultipleFailingCheck',
)
```

См.также:*Quality checks, Customizing checks*

4.12.7 DATA_DIR

Добавлено в версии 2.1: In previous versions the directories were configured separately as *GIT_ROOT* and *WHOOSH_INDEX*.

Directory where Weblate stores all data. This consists of VCS repositories, fulltext index and various configuration files for external tools.

Following subdirectories usually exist:

vcs Version control repositories.

whoosh Fulltext search index using Whoosh engine.

4.12.8 DEFAULT_COMMITTER_EMAIL

Добавлено в версии 2.4.

Default commiter email when creating translation component (see *Component configuration*), defaults to *noreply@weblate.org*.

См.также:*DEFAULT_COMMITTER_NAME, Component configuration*

4.12.9 DEFAULT_COMMITTER_NAME

Добавлено в версии 2.4.

Default commiter name when creating translation component (see *Component configuration*), defaults to *Weblate*.

См.также:*DEFAULT_COMMITTER_EMAIL, Component configuration*

4.12.10 ENABLE_AVATARS

Whether to enable libavatar/gravatar based avatars for users. By default this is enabled.

The avatars are fetched and cached on the server, so there is no risk in leaking private information or slowing down the user experiences with enabling this.

См.также:

Avatar caching

4.12.11 ENABLE_HOOKS

Whether to enable anonymous remote hooks.

См.также:

Notification hooks

4.12.12 ENABLE_HTTPS

Whether to send links to the Weblate as https or http. This setting only affects sent mails.

4.12.13 ENABLE_SHARING

Whether to show links to share translation progress on social networks.

4.12.14 GIT_ROOT

Не рекомендуется, начиная с версии 2.1: This setting is no longer used, use *DATA_DIR* instead.

Path where Weblate will store cloned VCS repositories. Defaults to `repos` subdirectory.

4.12.15 GITHUB_USERNAME

GitHub username that will be used to send pull requests for translation updates.

См.также:

Pushing changes to GitHub as pull request

4.12.16 GOOGLE_ANALYTICS_ID

Google Analytics ID to enable monitoring of Weblate using Google Analytics.

4.12.17 HIDE_REPO_CREDENTIALS

Hide repository credentials in the web interface. In case you have repository URL with user and password, Weblate will hide it when showing it to the users.

For example instead of `https://user:password@git.example.com/repo.git` it will show just `https://git.example.com/repo.git`.

4.12.18 LAZY_COMMITS

Delay creating VCS commits until this is necessary. This heavily reduces number of commits generated by Weblate at expense of temporarily not being able to merge some changes as they are not yet committed.

См.также:

Lazy commits

4.12.19 LOCK_TIME

Time in seconds for how long the translation will be locked for single translator when locked manually.

См.также:

Translation locking

4.12.20 LOGIN_REQUIRED_URLS

List of URL which require login (besides standard rules built into Weblate). This allows you to password protect whole installation using:

```
LOGIN_REQUIRED_URLS = (
    r'/(.*)$',
)
```

4.12.21 LOGIN_REQUIRED_URLS_EXCEPTIONS

List of exceptions for `LOGIN_REQUIRED_URLS`, in case you won't specify this list, the default value will be used, which allows users to access 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
)
```

4.12.22 MACHINE_TRANSLATION_SERVICES

List of enabled machine translation services to use.

Примечание: Many of services need additional configuration like API keys, please check their documentation for more details.

```
MACHINE_TRANSLATION_SERVICES = (
    'weblate.trans.machine.apertium.ApertiumTranslation',
    'weblate.trans.machine.glosbe.GlosbeTranslation',
)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
'weblate.trans.machine.google.GoogleTranslation',
'weblate.trans.machine.microsoft.MicrosoftTranslation',
'weblate.trans.machine.mymemory.MyMemoryTranslation',
'weblate.trans.machine.tmserver.TMServerTranslation',
'weblate.trans.machine.weblatetm.WeblateSimilarTranslation',
'weblate.trans.machine.weblatetm.WeblateTranslation',
)
```

См.также:

Machine translation setup, Machine translation

4.12.23 MT__APERTIUM__KEY

API key for Apertium Web Service, you can register at <http://api.apertium.org/register.jsp>

См.также:

Apertium, Machine translation setup, Machine translation

4.12.24 MT__GOOGLE__KEY

API key for Google Translate API, you can register at <https://cloud.google.com/translate/docs>

См.также:

Google Translate, Machine translation setup, Machine translation

4.12.25 MT__MICROSOFT__ID

Client ID for Microsoft Translator service.

См.также:

Microsoft Translator, Machine translation setup, Machine translation, <https://datamarket.azure.com/developer/applications/>

4.12.26 MT__MICROSOFT__SECRET

Client secret for Microsoft Translator service.

См.также:

Microsoft Translator, Machine translation setup, Machine translation, <https://datamarket.azure.com/developer/applications/>

4.12.27 MT__MYMEMORY__EMAIL

MyMemory identification email, you can get 1000 requests per day with this.

См.также:

MyMemory, Machine translation setup, Machine translation, <http://mymemory.translated.net/doc/spec.php>

4.12.28 MT_MYMEMORY_KEY

MyMemory access key for private translation memory, use together with *MT_MYMEMORY_USER*.

См.также:

MyMemory, *Machine translation setup*, *Machine translation*, <http://mymemory.translated.net/doc/keygen.php>

4.12.29 MT_MYMEMORY_USER

MyMemory user id for private translation memory, use together with *MT_MYMEMORY_KEY*.

См.также:

MyMemory, *Machine translation setup*, *Machine translation*, <http://mymemory.translated.net/doc/keygen.php>

4.12.30 MT_TMSERVER

URL where tmserver is running.

См.также:

tmserver, *Machine translation setup*, *Machine translation*, <http://docs.translatehouse.org/projects/translate-toolkit/en/latest/commands/tmserver.html>

4.12.31 NEARBY_MESSAGES

How many messages around current one to show during translating.

4.12.32 OFFLOAD_INDEXING

Offload updating of fulltext index to separate process. This heavily improves responsiveness of online operation on expense of slightly outdated index, which might still point to older content.

While enabling this, don't forget scheduling runs of *update_index* in cron or similar tool.

This is recommended setup for production use.

См.также:

Fulltext search

4.12.33 PIWIK_SITE_ID

ID of a site in Piwik you want to track.

См.также:

PIWIK_URL

4.12.34 PIWIK_URL

URL of a Piwik installation you want to use to track Weblate users. For more information about Piwik see <http://piwik.org/>.

См.также:

PIWIK_SITE_ID

4.12.35 POST_ADD_SCRIPTS

Добавлено в версии 2.4.

List of scripts which are allowed as post add scripts. The script needs to be later enabled in the *Component configuration*.

Weblate comes with few example hook scripts which you might find useful:

`examples/hook-update-linguas` Updates LINGUAS file or ALL_LINGUAS in configure script.

См.также:

Processing repository with scripts

4.12.36 POST_UPDATE_SCRIPTS

Добавлено в версии 2.3.

List of scripts which are allowed as post update scripts. The script needs to be later enabled in the *Component configuration*.

См.также:

Processing repository with scripts

4.12.37 PRE_COMMIT_SCRIPTS

List of scripts which are allowed as pre commit scripts. The script needs to be later enabled in the *Component configuration*.

For example you can allow script which does some cleanup:

```
PRE_COMMIT_SCRIPTS = (  
    '/usr/local/bin/cleanup-translation',  
)
```

Weblate comes with few example hook scripts which you might find useful:

`examples/hook-generate-mo` Generates MO file from a PO file

`examples/hook-unwrap-po` Unwraps lines in a PO file.

`examples/hook-sort-properties` Sort and cleanups Java properties file.

`examples/hook-replace-single-quotes` Replaces single quotes in a file.

См.также:

Processing repository with scripts

4.12.38 POST_COMMIT_SCRIPTS

Добавлено в версии 2.4.

List of scripts which are allowed as post commit scripts. The script needs to be later enabled in the *Component configuration*.

См.также:

Processing repository with scripts

4.12.39 POST_PUSH_SCRIPTS

Добавлено в версии 2.4.

List of scripts which are allowed as post push scripts. The script needs to be later enabled in the *Component configuration*.

См.также:

Processing repository with scripts

4.12.40 REGISTRATION_CAPTCHA

A boolean (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.

4.12.41 REGISTRATION_OPEN

A boolean (either `True` or `False`) indicating whether registration of new accounts is currently permitted. This setting is optional, and a default of `True` will be assumed if it is not supplied.

4.12.42 SELF_ADVERTISEMENT

Enables self advertisement of Weblate in case there are no configured ads.

См.также:

Advertisement

4.12.43 SIMPLIFY_LANGUAGES

Use simple language codes for default language/country combinations. For example `fr_FR` translation will use `fr` language code. This is usually desired behavior as it simplifies listing of the languages for these default combinations.

Disable this if you are having different translations for both variants.

4.12.44 SITE_TITLE

Site title to be used in website and emails as well.

4.12.45 SOURCE_LANGUAGE

Source language used for translation. This is mostly useful for machine translation services.

4.12.46 TTF_PATH

Path to Droid fonts used for widgets and charts.

4.12.47 URL_PREFIX

This settings allows you to run Weblate under some path (otherwise it relies on being executed from 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`.

Примечание: This setting does not work with Django's builtin server, you would have to adjust `urls.py` to contain this prefix.

4.12.48 WHOOSH_INDEX

Не рекомендуется, начиная с версии 2.1: This setting is no longer used, use `DATA_DIR` instead.

Directory where Whoosh fulltext indices will be stored. Defaults to `whoosh-index` subdirectory.

4.13 Sample configuration

Following example is shipped as `weblate/settings_example.py` with Weblate:

```
# -*- coding: utf-8 -*-
#
# Copyright © 2012 - 2015 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <http://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 <http://www.gnu.org/licenses/>.
#

import django
import os
from logging.handlers import SysLogHandler
```

(continues on next page)

(продолжение с предыдущей страницы)

```

#
# Safety check for running with too old Django version
#

if django.VERSION < (1, 4, 0):
    raise Exception(
        'Weblate needs Django 1.4 or newer, you are using %s!' %
        django.get_version()
    )

#
# Django settings for Weblate project.
#

DEBUG = True
TEMPLATE_DEBUG = DEBUG

ADMINS = (
    # ('Your Name', 'your_email@example.com'),
)

MANAGERS = ADMINS

DATABASES = {
    'default': {
        # Use 'postgresql_psycopg2', 'mysql', 'sqlite3' or 'oracle'.
        'ENGINE': 'django.db.backends.sqlite3',
        # Database name or path to database file if using sqlite3.
        'NAME': 'weblate.db',
        # Database user, not used with sqlite3.
        'USER': 'weblate',
        # Database password, not used with sqlite3.
        'PASSWORD': 'weblate',
        # Set to empty string for localhost. Not used with sqlite3.
        'HOST': '127.0.0.1',
        # Set to empty string for default. Not used with sqlite3.
        'PORT': '',
    }
}

BASE_DIR = 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.
# On Unix systems, a value of None will cause Django to use the same
# timezone as the operating system.
# If running in a Windows environment this must be set to the same as your
# system time zone.
TIME_ZONE = 'UTC'

# Language code for this installation. All choices can be found here:

```

(continues on next page)

(продолжение с предыдущей страницы)

```
# http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = 'en-us'

LANGUAGES = (
    ('az', u'Azərbaycan'),
    ('be', u'Беларуская'),
    ('be@latin', u'Biełaruskaja'),
    ('br', u'Brezhoneg'),
    ('ca', u'Català'),
    ('cs', u'Čeština'),
    ('da', u'Dansk'),
    ('de', u'Deutsch'),
    ('en', u'English'),
    ('el', u'Ελληνική'),
    ('es', u'Español'),
    ('fi', u'Suomi'),
    ('fr', u'Français'),
    ('fy', u'Frysk'),
    ('gl', u'Galego'),
    ('he', u''),
    ('hu', u'Magyar'),
    ('id', u'Indonesia'),
    ('ja', u''),
    ('ko', u''),
    ('ksh', u'Kölsch'),
    ('nl', u'Nederlands'),
    ('pl', u'Polski'),
    ('pt', u'Português'),
    ('pt-BR', u'Português brasileiro'),
    ('ru', u'Русский'),
    ('sk', u'Slovenčina'),
    ('sl', u'Slovenščina'),
    ('sr', u'Српски'),
    ('sv', u'Svenska'),
    ('tr', u'Türkçe'),
    ('uk', u'Українська'),
    ('zh-Hans', u''),
    ('zh-Hant', u''),
)

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 = False

# URL prefix to use, please see documentation for more details
URL_PREFIX = ''
```

(continues on next page)

(продолжение с предыдущей страницы)

```

# Absolute filesystem path to the directory that will hold user-uploaded files.
# Example: "/home/media/media.lawrence.com/media/"
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

# URL that handles the media served from MEDIA_ROOT. Make sure to use a
# trailing slash.
# Examples: "http://media.lawrence.com/media/", "http://example.com/media/"
MEDIA_URL = '%s/media/' % URL_PREFIX

# 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.
# Example: "/home/media/media.lawrence.com/static/"
STATIC_ROOT = os.path.join(DATA_DIR, 'static')

# URL prefix for static files.
# Example: "http://media.lawrence.com/static/"
STATIC_URL = '%s/static/' % URL_PREFIX

# 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 examples/generate-secret-key
SECRET_KEY = 'jm8fqjlg+5!#xu%e-oh#7!$aa7!6avf7ud*_v=chdrb9qdco6('

# List of callables that know how to import templates from various sources.
TEMPLATE_LOADERS = (
    ('django.template.loaders.cached.Loader', (
        'django.template.loaders.filesystem.Loader',
        'django.template.loaders.app_directories.Loader',
    )),
)

# GitHub username for sending pull requests.
# Please see the documentation for more details.
GITHUB_USERNAME = None

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'weblate.accounts.auth.EmailAuth',
    # 'social.backends.google.GoogleOAuth2',
    # 'social.backends.github.GithubOAuth2',
    # 'social.backends.bitbucket.BitbucketOAuth',

```

(continues on next page)

(продолжение с предыдущей страницы)

```

    # 'social.backends.suse.OpenSUSEOpenId',
    # 'social.backends.ubuntu.UbuntuOpenId',
    # 'social.backends.fedora.FedoraOpenId',
    # 'social.backends.facebook.FacebookOAuth2',
    'weblate.accounts.auth.WeblateUserBackend',
)

# 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_GOOGLE_OAUTH2_KEY = ''
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = ''

# Social auth settings
SOCIAL_AUTH_PIPELINE = (
    'social.pipeline.social_auth.social_details',
    'social.pipeline.social_auth.social_uid',
    'social.pipeline.social_auth.auth_allowed',
    'social.pipeline.social_auth.associate_by_email',
    'social.pipeline.social_auth.social_user',
    'social.pipeline.user.get_username',
    'weblate.accounts.pipeline.require_email',
    'social.pipeline.mail.mail_validation',
    'social.pipeline.social_auth.associate_by_email',
    'weblate.accounts.pipeline.verify_open',
    'weblate.accounts.pipeline.verify_username',
    'social.pipeline.user.create_user',
    'social.pipeline.social_auth.associate_user',
    'social.pipeline.social_auth.load_extra_data',
    'weblate.accounts.pipeline.user_full_name',
    'weblate.accounts.pipeline.store_email',
)

# Custom authentication strategy
SOCIAL_AUTH_STRATEGY = 'weblate.accounts.strategy.WeblateStrategy'

SOCIAL_AUTH_EMAIL_VALIDATION_FUNCTION = \
    'weblate.accounts.pipeline.send_validation'
SOCIAL_AUTH_EMAIL_VALIDATION_URL = '%s/accounts/email-sent/' % URL_PREFIX
SOCIAL_AUTH_LOGIN_ERROR_URL = '%s/accounts/login/' % URL_PREFIX
SOCIAL_AUTH_EMAIL_FORM_URL = '%s/accounts/email/' % URL_PREFIX
SOCIAL_AUTH_NEW_ASSOCIATION_REDIRECT_URL = \
    '%s/accounts/profile/#auth' % URL_PREFIX
SOCIAL_AUTH_PROTECTED_USER_FIELDS = ('email',)
SOCIAL_AUTH_SLUGIFY_USERNAMES = True

```

(continues on next page)

(продолжение с предыдущей страницы)

```

# Middleware
MIDDLEWARE_CLASSES = (
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.locale.LocaleMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'social.apps.django_app.middleware.SocialAuthExceptionMiddleware',
    'weblate.accounts.middleware.RequireLoginMiddleware',
)

ROOT_URLCONF = 'weblate.urls'

TEMPLATE_DIRS = (
    # Put strings here, like "/home/html/django_templates"
    # or "C:/www/django/templates".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not relative paths.
    os.path.join(BASE_DIR, 'html'),
)

INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.admin',
    'django.contrib.admindocs',
    'django.contrib.sitemaps',
    'social.apps.django_app.default',
    'crispy_forms',
    'compressor',
    'weblate.trans',
    'weblate.lang',
    'weblate.accounts',
    # Needed for javascript localization
    'weblate',
)

LOCALE_PATHS = (os.path.join(BASE_DIR, '..', 'locale'), )

TEMPLATE_CONTEXT_PROCESSORS = (
    'django.contrib.auth.context_processors.auth',
    'django.core.context_processors.debug',
    'django.core.context_processors.i18n',
    'django.core.context_processors.request',
    'django.core.context_processors.csrf',
    'django.contrib.messages.context_processors.messages',
    'weblate.trans.context_processors.weblate_context',
)

```

(continues on next page)

(продолжение с предыдущей страницы)

```

# 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

if DEBUG or not os.path.exists('/dev/log'):
    DEFAULT_LOG = 'console'
else:
    DEFAULT_LOG = 'syslog'

# 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': False,
    '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'
        },
    },
    'handlers': {
        'mail_admins': {
            'level': 'ERROR',
            'filters': ['require_debug_false'],
            'class': 'django.utils.log.AdminEmailHandler'
        },
        'console': {
            'level': 'DEBUG',
            'class': 'logging.StreamHandler',
            'formatter': 'simple'
        },
        'syslog': {
            'level': 'DEBUG',
            'class': 'logging.handlers.SysLogHandler',
            'formatter': 'syslog',
            'address': '/dev/log',
            'facility': SysLogHandler.LOG_LOCAL2,

```

(continues on next page)

(продолжение с предыдущей страницы)

```

    },
    # 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,
        },
        # Logging database queries
        # 'django.db.backends': {
        #     'handlers': [DEFAULT_LOG],
        #     'level': 'DEBUG',
        # },
        'weblate': {
            'handlers': [DEFAULT_LOG],
            'level': 'DEBUG',
        }
    }
}

# Logging of management commands to console
if (os.environ.get('DJANGO_IS_MANAGEMENT_COMMAND', False) and
    'console' not in LOGGING['loggers']['weblate']['handlers']):
    LOGGING['loggers']['weblate']['handlers'].append('console')

# Remove syslog setup if it's not present
if not os.path.exists('/dev/log'):
    del LOGGING['handlers']['syslog']

# Machine translation API keys

# Apertium Web Service, register at http://api.apertium.org/register.jsp
MT_APERTIUM_KEY = None

# Microsoft Translator service, register at
# https://datamarket.azure.com/developer/applications/
MT_MICROSOFT_ID = None
MT_MICROSOFT_SECRET = None

# MyMemory identification email, see
# http://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

```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Google API key for Google Translate API
MT_GOOGLE_KEY = None

# tmserver URL
MT_TMSERVER = None

# Title of site to use
SITE_TITLE = u'Weblate'

# URL of login
LOGIN_URL = '%s/accounts/login/' % URL_PREFIX

# URL of logout
LOGOUT_URL = '%s/accounts/logout/' % URL_PREFIX

# Default location for login
LOGIN_REDIRECT_URL = '%s/' % URL_PREFIX

# Anonymous user name
ANONYMOUS_USER_NAME = 'anonymous'

# Sending HTML in mails
EMAIL_SEND_HTML = False

# Subject of emails includes site title
EMAIL_SUBJECT_PREFIX = u'[{0}] '.format(SITE_TITLE)

# Enable remote hooks
ENABLE_HOOKS = True

# Whether to run hooks in background
BACKGROUND_HOOKS = True

# Number of nearby messages to show in each direction
NEARBY_MESSAGES = 5

# Enable lazy commits
LAZY_COMMITS = True

# Offload indexing
OFFLOAD_INDEXING = False

# Translation locking
AUTO_LOCK = True
AUTO_LOCK_TIME = 60
LOCK_TIME = 15 * 60

# Render forms using bootstrap
CRISPY_TEMPLATE_PACK = 'bootstrap3'

# List of quality checks
# CHECK_LIST = (
#     'weblate.trans.checks.same.SameCheck',
#     'weblate.trans.checks.chars.BeginNewlineCheck',
#     'weblate.trans.checks.chars.EndNewlineCheck',
#     'weblate.trans.checks.chars.BeginSpaceCheck',
```

(continues on next page)

(продолжение с предыдущей страницы)

```

# 'weblate.trans.checks.chars.EndSpaceCheck',
# 'weblate.trans.checks.chars.EndStopCheck',
# 'weblate.trans.checks.chars.EndColonCheck',
# 'weblate.trans.checks.chars.EndQuestionCheck',
# 'weblate.trans.checks.chars.EndExclamationCheck',
# 'weblate.trans.checks.chars.EndEllipsisCheck',
# 'weblate.trans.checks.format.PythonFormatCheck',
# 'weblate.trans.checks.format.PythonBraceFormatCheck',
# 'weblate.trans.checks.format.PHPFormatCheck',
# 'weblate.trans.checks.format.CFormatCheck',
# 'weblate.trans.checks.format.JavascriptFormatCheck',
# 'weblate.trans.checks.consistency.PluralsCheck',
# 'weblate.trans.checks.consistency.ConsistencyCheck',
# 'weblate.trans.checks.chars.NewlineCountingCheck',
# 'weblate.trans.checks.markup.BBCodeCheck',
# 'weblate.trans.checks.chars.ZeroWidthSpaceCheck',
# 'weblate.trans.checks.markup.XMLTagsCheck',
# 'weblate.trans.checks.source.OptionalPluralCheck',
# 'weblate.trans.checks.source.EllipsisCheck',
# 'weblate.trans.checks.source.MultipleFailingCheck',
# )

# List of automatic fixups
# AUTOFIX_LIST = (
#     'weblate.trans.autofixes.whitespace.SameBookendingWhitespace',
#     'weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis',
#     'weblate.trans.autofixes.chars.RemoveZeroSpace',
# )

# List of scripts to use in custom processing
# POST_UPDATE_SCRIPTS = (
# )
# PRE_COMMIT_SCRIPTS = (
# )

# List of machine translations
# MACHINE_TRANSLATION_SERVICES = (
#     'weblate.trans.machine.apertium.ApertiumTranslation',
#     'weblate.trans.machine.glosbe.GlosbeTranslation',
#     'weblate.trans.machine.google.GoogleTranslation',
#     'weblate.trans.machine.google.GoogleWebTranslation',
#     'weblate.trans.machine.microsoft.MicrosoftTranslation',
#     'weblate.trans.machine.mymemory.MyMemoryTranslation',
#     'weblate.trans.machine.tmserver.AmagamaTranslation',
#     'weblate.trans.machine.tmserver.TMServerTranslation',
#     'weblate.trans.machine.weblatetm.WeblateSimilarTranslation',
#     'weblate.trans.machine.weblatetm.WeblateTranslation',
# )

# E-mail address that error messages come from.
SERVER_EMAIL = 'noreply@weblate.org'

# Default email address to use for various automated correspondence from
# the site managers. Used for registration emails.
DEFAULT_FROM_EMAIL = 'noreply@weblate.org'

```

(continues on next page)

(продолжение с предыдущей страницы)

```
# List of URLs your site is supposed to serve
ALLOWED_HOSTS = []

# Example configuration to use memcached for caching
# CACHES = {
#     'default': {
#         'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
#         'LOCATION': '127.0.0.1:11211',
#     },
#     'avatar': {
#         'BACKEND': 'django.core.cache.backends.filebased.FileBasedCache',
#         'LOCATION': os.path.join(BASE_DIR, 'avatar-cache'),
#         'TIMEOUT': 604800,
#         'OPTIONS': {
#             'MAX_ENTRIES': 1000,
#         },
#     },
# }

# Example for restricting access to logged in users
# LOGIN_REQUIRED_URLS = (
#     r'/(.*)$',
# )

# In such case you will want to include some of the exceptions
# 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
# )

# Enable whiteboard functionality - under development so disabled by default.
ENABLE_WHITEBOARD = False

# Force sane test runner
TEST_RUNNER = 'django.test.runner.DiscoverRunner'
```

4.14 Management commands

Примечание: Running management commands under different user than is running your webserver can cause wrong permissions on some files, please check *Filesystem permissions* for more details.

The `./manage.py` is extended with following commands:

4.14.1 changesite

`manage.py changesite`

Добавлено в версии 2.4.

You can use this to changes site name from command line with `--set-name` parameter. The `--get-name` prints currently configured site name.

См.также:

Set correct site name

4.14.2 checkgit <project|project/component>

```
manage.py checkgit
```

Prints current state of backend git repository.

You can either define which project or component to update (eg. `weblate/master`) or use `--all` to update all existing components.

4.14.3 commitgit <project|project/component>

```
manage.py commitgit
```

Commits any possible pending changes to backend git repository.

You can either define which project or component to update (eg. `weblate/master`) or use `--all` to update all existing components.

4.14.4 commit_pending <project|project/component>

```
manage.py commit_pending
```

Commits pending changes older than given age (using `--age` parameter, defaults to 24 hours).

You can either define which project or component to update (eg. `weblate/master`) or use `--all` to update all existing components.

This is most useful if executed periodically from cron or similar tool:

```
./manage.py commit_pending --all --age=48
```

См.также:

Running maintenance tasks

4.14.5 cleanuptrans

```
manage.py cleanuptrans
```

Cleanups orphaned checks and translation suggestions.

См.также:

Running maintenance tasks

4.14.6 createadmin

`manage.py createadmin`

Creates `admin` account with random password. You can specify `--password` to provide password on the command line.

4.14.7 dumpuserdata <file.json>

`manage.py dumpuserdata`

Dumps userdata to file for later use by *`importuserdata`*

This is useful when migrating or merging Weblate instances.

4.14.8 import_project <project> <gitrepo> <branch> <filemask>

`manage.py import_project`

Batch imports components into project based on file mask.

<project> names an existing project, into which the components should be imported.

The <gitrepo> defines URL of Git repository to use, and <branch> the git branch. To import additional translation components, from an existing Weblate component, use a *`weblate://<project>/<component>`* URL for the <gitrepo>.

The repository is searched for directories matching a double wildcard (`**`) in the <filemask>. Each of these is then added as a component, named after the matched directory. Existing components will be skipped.

To customise the component's name, use the `--name-template` option. Its parameter is a python formatting string, which will expect the match from <filemask>.

By format string passed by the `--base-file-template` option you can customize base file for monolingual translations.

You can also specify file format to use (see *[Supported formats](#)*) by the `--file-format` parameter. The default is autodetection.

In case you need to specify version control system to use, you can do this using `--vcs` parameter. The default version control is Git.

You can override parsing of component name from matched files by `--component-regexp`. This is a regular expression which will be matched against file name (as matched by <filemask>) and has to contain named group *name*. This can be also used for excluding files in case they do not match this expression. For example: `.*/(?P<name>[^-]*)\.po`

To give you some examples, let's try importing two projects.

As first we import The Debian Handbook translations, where each language has separate folder with translations of each chapter:

```
./manage.py import_project \
  debian-handbook \
  git://anonscm.debian.org/debian-handbook/debian-handbook.git \
  squeeze/master \
  '*/**.po'
```

Another example can be Tanaguru tool, where we need to specify file format, base file template and has all components and translations located in single folder:

```
./manage.py 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
```

Example of more complex parsing of filenames to get correct component and language out of file name like `src/security/Numerous_security_holes_in_0.10.1.de.po`:

```
./manage.py import_project \
  --component-regex 'wiki/src/security/(?P<name>.*)\.([^.]*)\.po$' \
  tails \
  git://git.tails.boum.org/tails master \
  'wiki/src/security/**/*.po'
```

4.14.9 importuserdata <file.json>

```
manage.py importuserdata
```

Imports userdata from file created by *dumpuserdata*

4.14.10 importusers --check <file.json>

```
manage.py importusers
```

Imports users from JSON dump of Django `auth_users` database.

With `--check` it will just check whether given file can be imported and report possible conflicts on usernames or emails.

You can dump users from existing Django installation using:

```
./manage.py dumpdata auth.User > users.json
```

4.14.11 list_ignored_checks

```
manage.py list_ignored_checks
```

Lists most frequently ignored checks. This can be useful for tuning your setup, if users have to ignore too many of consistency checks.

4.14.12 list_versions

```
manage.py list_versions
```

Lists versions of Weblate dependencies.

4.14.13 loadpo <project|project/component>

`manage.py loadpo`

Reloads translations from disk (eg. in case you did some updates in VCS repository).

You can use `--force` to force update even if the files should be up to date. Additionally you can limit languages to process with `--lang`.

You can either define which project or component to update (eg. `weblate/master`) or use `--all` to update all existing components.

4.14.14 lock_translation <project|project/component>

`manage.py lock_translation`

Locks given component for translating. This is useful in case you want to do some maintenance on underlying repository.

You can either define which project or component to update (eg. `weblate/master`) or use `--all` to update all existing components.

См.также:

`unlock_translation`

4.14.15 pushgit <project|project/component>

`manage.py pushgit`

Pushes committed changes to upstream VCS repository. With `--force-commit` it also commits any pending changes.

You can either define which project or component to update (eg. `weblate/master`) or use `--all` to update all existing components.

4.14.16 rebuild_index <project|project/component>

`manage.py rebuild_index`

Rebuilds index for fulltext search. This might be lengthy operation if you have huge set of translation units.

You can use `--clean` to remove all words from database prior updating.

With `--optimize` the index will not be processed again, only it's content will be optimized (removing stale entries and merging possibly split index files).

См.также:

`Fulltext search`

4.14.17 update_index

`manage.py update_index`

Updates index for fulltext search when *OFFLOAD_INDEXING* is enabled.

It is recommended to run this frequently (eg. every 5 minutes) to have index uptodate.

См.также:

Fulltext search, Running maintenance tasks, Enable indexing offloading

4.14.18 `unlock_translation` <project|project/component>

```
manage.py unlock_translation
```

Unlocks given component for translating. This is useful in case you want to do some maintenance on underlaying repository.

You can either define which project or component to update (eg. `weblate/master`) or use `--all` to update all existing components.

См.также:

lock_translation

4.14.19 `setupgroups`

```
manage.py setupgroups
```

Configures default groups and (if called with `--move`) assigns all users to default group.

The option `--no-privs-update` disables update of existing groups (only adds new ones).

См.также:

Access control

4.14.20 `setuplang`

```
manage.py setuplang
```

Sets up list of languages (it has own list and all defined in `translate-toolkit`).

The option `--no-update` disables update of existing languages (only adds new ones).

4.14.21 `updatechecks` <project|project/component>

```
manage.py updatechecks
```

Updates all check for all units. This could be useful only on upgrades which do major changes to checks.

You can either define which project or component to update (eg. `weblate/master`) or use `--all` to update all existing components.

4.14.22 `updategit` <project|project/component>

```
manage.py updategit
```

Fetches remote VCS repositories and updates internal cache.

You can either define which project or component to update (eg. `weblate/master`) or use `--all` to update all existing components.

4.15 Advertisement

Weblate allows you to place advertisements inside emails and web pages. You can add them in the admin interface, defining placement and timespan, when it should be shown.

Active advertisements are chosen randomly to be displayed inside defined placement.

When no advertisements are enabled, you can enable `SELF_ADVERTISEMENT` to show advertisements for Weblate.

Frequently Asked Questions

5.1 Configuration

5.1.1 How to create automatic workflow?

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

1. Set up your git repository to tell Weblate whenever there is any change, see *Notification hooks* for information how to do it.
2. Set push URL at your *Component configuration* in Weblate, this will allow Weblate to push changes to your repository.
3. Enable push on commit on your *Project configuration* in Weblate, this will make Weblate push changes to your repository whenever they are committed at Weblate.
4. Optionally setup cron job for *commit_pending*.

См.также:

Continuous translation

5.1.2 How to access repositories over SSH?

Please see *Private repositories* for information about setting up SSH keys.

5.1.3 How to fix merge conflicts in translations?

The easiest way is to solve all conflicts locally at your workstation - simply add Weblate as remote repository, merge it into upstream and fix conflicts. Once you push changes back, Weblate will be able to use merged version without any other special actions.

```
# Add remote
git remote add weblate git://git.weblate.org/debian-handbook.git

# Update remotes
git remote update

# Merge Weblate changes
git merge weblate/master

# Resolve conflicts
edit ....
git add ...
...
git commit

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

См.также:

How to export Git repository weblate uses?

5.1.4 How do I translate several branches at once?

Weblate supports pushing translation changes within one *Project configuration*. For every *Component configuration* which has it enabled (the default behavior), the change made is automatically propagated to others. This way the 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
```

5.1.5 How to export Git repository weblate uses?

There is nothing special about the repository, it lives under *DATA_DIR* directory and is named as *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 git server and let it serve the repository to outside world.

5.1.6 What are options of pushing changes back upstream?

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

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

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

5.1.7 How can I check if my Weblate is configured properly?

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

5.1.8 Why does registration contain example.com as domain?

Weblate uses Django sites framework and it defines site name inside the database.

См.также:

Set correct site name

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

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

The author of every commit (when underlaying VCS supports it) is still recorded correctly as an user who has made the translation.

См.также:

Component configuration

5.2 Usage

5.2.1 How do I review others translations?

- You can subscribe to any changes made in *Subscriptions* and then check other contributions in email.
- There is review tool available at bottom of translation view, where you can choose to browse translations made by others since given date.

5.2.2 How do I provide feedback on source string?

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

5.2.3 How can I use existing translations while translating?

Weblate provides you several ways to utilize existing translations while translating:

- You can use import functionality to load compendium as translations, suggestions or fuzzy translations. This is best approach for one time translation using compendium or similar translation database.
- You can setup *tmserver* with all databases you have and let Weblate use it. This is good for case when you want to use it for several times during translating.
- Another option is to translate all related projects in single Weblate instance, what will make it automatically pick up translation from other projects as well.

См.также:

Machine translation setup, Machine translation

5.2.4 Does Weblate update translation files besides translations?

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

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

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 on your own.

См.также:

Processing repository with scripts

5.2.5 Where do language definition come from and how can I add own?

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

You are free to define own language in administrative interface, you just need to provide information about it.

5.2.6 Can Weblate highlight change in a fuzzy string?

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

For Gettext PO files, you have to pass 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.

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

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

Примечание: It is usually good idea to merge changed 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 can want to do the update automatically, you can add custom script to handle this to `POST_UPDATE_SCRIPTS` and enable it in the *Component configuration*.

5.3 Troubleshooting

5.3.1 Requests sometimes fail with too many open files error

This happens sometimes when your Git repository grows too much and you have more 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
```

См.также:

DATA_DIR

5.3.2 Fulltext search is too slow

Depending on various conditions (frequency of updates, server restarts and other), fulltext index might get too fragmented over time. It is recommended to optimize it time to time:

```
./manage.py rebuild_index --optimize
```

In case it does not help (or if you have removed lot of strings) it might be better to rebuild it from scratch:

```
./manage.py rebuild_index --clean
```

См.также:

rebuild_index

5.3.3 I get «Lock Error» quite often while translating

This is usually caused by concurrent updates to fulltext index. In case you are running multi threaded server (eg. `mod_wsgi`), this happens quite often. For such setup it is recommended to enable *OFFLOAD_INDEXING*.

См.также:

Fulltext search

5.3.4 Rebuilding index has failed with «No space left on device»

Whoosh uses temporary directory to build indices. In case you have small /tmp (eg. using ramdisk), this might fail. Change used temporary directory by passing as `TEMP` variable:

```
TEMP=/path/to/big/temp ./manage.py rebuild_index --clean
```

См.также:

`rebuild_index`

5.3.5 Database operations fail with «too many SQL variables»

This can happen with SQLite database as it is not powerful enough for some relations used within Weblate. The only way to fix this is to use some more capable database, see [Use powerful database engine](#) for more information.

См.также:

[Use powerful database engine](#), Django's databases

5.4 Features

5.4.1 Does Weblate support other VCS than Git and Mercurial?

Weblate currently does not have native support for anything else than Git and Mercurial, but it is possible to write backends for other VCSes.

You can also use Git [remote helpers](#) for supporting other VCS as well, but this usually leads to smaller or bigger problems, so be prepared to debug them.

At this time, helpers for Bazaar and Mercurial are available within separate repositories on GitHub: [git-remote-hg](#) and [git-remote-bzr](#). You can download them manually and put somewhere in your search path (for example ~/bin). You also need to have installed appropriate version control programs as well.

Once you have these installed, you can use such remotes to specify repository in Weblate.

To clone `gnuhello` project from Launchpad with Bazaar use:

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

For `hello` repository from `selenic.com` with Mercurial use:

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

Предупреждение: Please be prepared to some inconvenience when using Git remote helpers, for example with Mercurial, the remote helper sometimes tends to create new tip when pushing changes back.

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

5.4.2 How does Weblate credit translators?

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

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

5.4.3 Why does Weblate force to have show all po files in single tree?

Weblate was designed in a way that every po file is represented as single component. This is beneficial for translators, that they know what they are actually translating. If you feel your project should be translated as one, consider merging these po files. It will make life easier even for translators not using Weblate.

Примечание: In case there will be big demand for this feature, it might be implemented in future versions, but it's definitely not a priority for now.

Supported formats

Weblate supports any translation format understood by Translate-toolkit, however each format being slightly different, there might be some issues with not well tested formats.

См.также:

Supported formats in translate-toolkit

Weblate does support both monolingual and bilingual formats. Bilingual formats store two languages in single file - source and translation (typical examples is *GNU Gettext*, *XLIFF* or *Apple OS X strings*). On the other side, monolingual formats identify the string by ID and each language file contains only mapping of those to given language (typically *Android string resources*). Some file formats are used in both variants, see detailed description below.

For correct use of monolingual files, Weblate requires access to file containing complete list of strings to translate with their source - this file is called *Monolingual base language file* within Weblate, though the naming might vary in your application.

6.1 Automatic detection

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

6.2 GNU Gettext

Most widely used format in translating free software. This was first format supported by Weblate and still has best support.

Weblate supports contextual information stored in the file, adjusting its headers or linking to corresponding source files.

The bilingual gettext PO file typically looks like:

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

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

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

См.также:

[Gettext on Wikipedia](#), [Gettext in translate-toolkit documentation](#)

6.2.1 Monolingual Gettext

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

The monolingual gettext PO file typically looks like:

```
#: 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"
```

6.3 XLIFF

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

XLIFF is usually used as bilingual.

См.также:

[XLIFF on Wikipedia](#), [XLIFF in translate-toolkit documentation](#)

6.4 Java properties

Native Java format for translations.

Java properties are usually used as monolingual.

This format supports creating new languages. When a new languages is created, a new empty file will be added to the repository. Only keys that are defined will be written to the newly created file. The Weblate maintainer needs to make sure that this is the expected behaviour with the framework in use.

Weblate supports ISO-8859-1, UTF-8 and UTF-16 variants of this format.

См.также:

[Java properties on Wikipedia](#), [Java properties in translate-toolkit documentation](#)

6.5 Qt Linguist .ts

Translation format used in Qt based applications.

Qt Linguist files are used as both bilingual and monolingual.

Примечание: Weblate does not support adding new language to the translations using this format. This has to be done manually in the VCS.

См.также:

[Qt Linguist manual](#), [Qt .ts in translate-toolkit documentation](#)

6.6 Android string resources

Android specific file format for translating applications.

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

См.также:

[Android string resources documentation](#), [Android string resources in translate-toolkit documentation](#)

Примечание: Android *string-array* structures are not currently supported. To work around this, you can break you 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 localized.

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

6.7 Apple OS X strings

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

Apple OS X strings are usually used as bilingual.

См.также:

[Apple Strings Files documentation](#), [Apple strings in translate-toolkit documentation](#)

Примечание: You need translate-toolkit 1.12.0 or newer for proper support of Apple OS X strings. Older versions might produce corrupted files.

6.8 PHP strings

PHP translations are usually monolingual, so it is recommended to specify base file with English strings.

Example file:

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

Примечание: Translate-toolkit currently has some limitations in processing PHP files, so please double check that your files won't get corrupted before using Weblate in production setup.

См.также:

[PHP files in translate-toolkit documentation](#)

6.9 JSON files

Добавлено в версии 2.0.

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

JSON translations are usually monolingual, so it is recommended to specify base file with English strings.

Примечание: Weblate currently supports only simple JSON files with key value mappings, more complex formats like the ones used by Chrome extensions are currently not supported by translate-toolkit and will produce invalid results.

Example file:

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

См.также:

[JSON in translate-toolkit documentation](#)

6.10 .Net Resource files

Добавлено в версии 2.3.

.Net Resource (.resx) file is a monolingual XML file format used in Microsoft .Net Applications.

Примечание: You need translate-toolkit 1.13.0 or newer to include support for this format.

6.11 CSV files

Добавлено в версии 2.4.

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

- Files with header defining fields (source, translation, location, ...)
- Files with two field - source and translation (in this order)
- Files with fields as defined by translate-toolkit: location, source, target, id, fuzzy, context, translator_comments, developer_comments

6.12 Others

As already mentioned, all Translate-toolkit formats are supported, but they did not (yet) receive deeper testing.

См.также:

[Supported formats in translate-toolkit](#)

7.1 Notification hooks

Notification hooks allow external applications to notify Weblate that VCS repository has been updated.

GET `/hooks/update/(string: project)/string: component/` Triggers update of a component (pulling from VCS and scanning for translation changes).

GET `/hooks/update/(string: project)/` 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.

Примечание: GitHub includes direct support for notifying Weblate, just enable Weblate service hook in repository settings and set URL to URL of your Weblate installation.

См.также:

Automatically receiving changes from GitHub For instruction on setting up GitHub integration <https://help.github.com/articles/creating-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.

См.также:

Automatically receiving changes from GitLab For instruction on setting up GitLab integration

http://doc.gitlab.com/ce/web_hooks/web_hooks.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.

См.также:

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

7.2 Exports

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

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

Query Parameters

- *indent* (*integer*) – pretty printed indentation
- *jsonp* (*string*) – JSONP callback function to wrap the data

Retrieves statistics for given component in JSON format. Optionally as JSONP when you specify the callback in the *jsonp* parameter.

You can get pretty-printed output by appending *?indent=1* to the request.

Example request:

```
GET /exports/stats/weblate/master/?indent=4 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 \u010ciha\u0159",
    "last_change": "2012-03-28T15:07:38+00:00",
    "name": "Czech",
    "total": 436,
```

(continues on next page)

(продолжение с предыдущей страницы)

```

    "total_words": 15271,
    "translated": 436,
    "translated_percent": 100.0,
    "translated_words": 3201,
    "url": "http://hosted.weblate.org/engage/weblate/cs/",
    "url_translate": "http://hosted.weblate.org/projects/weblate/master/cs/"
  },
  {
    "code": "nl",
    "failing": 21,
    "failing_percent": 4.8,
    "fuzzy": 11,
    "fuzzy_percent": 2.5,
    "last_author": null,
    "last_change": null,
    "name": "Dutch",
    "total": 436,
    "total_words": 15271,
    "translated": 319,
    "translated_percent": 73.2,
    "translated_words": 3201,
    "url": "http://hosted.weblate.org/engage/weblate/nl/",
    "url_translate": "http://hosted.weblate.org/projects/weblate/master/nl/"
  },
  {
    "code": "el",
    "failing": 11,
    "failing_percent": 2.5,
    "fuzzy": 21,
    "fuzzy_percent": 4.8,
    "last_author": null,
    "last_change": null,
    "name": "Greek",
    "total": 436,
    "total_words": 15271,
    "translated": 312,
    "translated_percent": 71.6,
    "translated_words": 3201,
    "url": "http://hosted.weblate.org/engage/weblate/el/",
    "url_translate": "http://hosted.weblate.org/projects/weblate/master/el/"
  },
]

```

Included data:

code language code

failing, failing_percent number and percentage of failing checks

fuzzy, fuzzy_percent number and percentage of fuzzy strings

total_words total number of words

translated_words number of translated words

last_author name of last author

last_change date of last change

name language name

`total` total number of strings

`translated, translated_percent` number and percentage of translated strings

`url` URL to access the translation (engagement URL)

`url_translate` URL to access the translation (real translation URL)

7.3 RSS feeds

Changes in translations are exported in RSS feeds.

GET `/exports/rss/(string: project)/`
`string: component/string: language/` Retrieves RSS feed with recent changes for a translation.

GET `/exports/rss/(string: project)/`
`string: component/` Retrieves RSS feed with recent changes for a component.

GET `/exports/rss/(string: project)/`
Retrieves RSS feed with recent changes for a project.

GET `/exports/rss/language/(string: language)/`
Retrieves RSS feed with recent changes for a language.

GET `/exports/rss/`
Retrieves RSS feed with recent changes for Weblate instance.

См.также:

<https://en.wikipedia.org/wiki/RSS>

8.1 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 chars.
- Added support for Bitbucket webhooks.
- Tigher 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.

8.2 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 OAuth2.
- 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.

8.3 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 file names.
- 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.

8.4 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.

8.5 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 possibilites.
- 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 chars.
- Support for exporting stats as JSONP.

8.6 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.

8.7 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 email instead of username.
- Documentation improvements.
- Improved source strings review.
- Searching across all units.
- Better tracking of source strings.
- Captcha protection for registration.

8.8 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 units.
- 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.

8.9 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.

8.10 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.

8.11 weblate 1.4

Released on January 23rd 2013.

- Fixed deleting of checks/comments on unit 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 emails 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 unit tests coverage.

8.12 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 (eg. 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.

8.13 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 email notifications of translators.
- List only used languages in preferences.
- Improved handling of not known languages when importing project.
- Support for locking translation by translator.
- Optionally maintain Language-Team header in po file.
- Include some statistics in about page.
- Supports (and requires) django-registration 0.8.
- Caching of counted units with failing checks.
- Checking of requirements during setup.
- Documentation improvements.

8.14 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.

8.15 weblate 1.0

Released on May 10th 2012.

- Improved validation while adding/saving component.
- Experimental support for Android component files (needs patched ttkit).
- Updates from hooks are run in background.
- Improved installation instructions.
- Improved navigation in dictionary.

8.16 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.

8.17 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.

8.18 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.

8.19 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.

8.20 weblate 0.5

Released on February 12th 2012.

- **Support for machine translation using following online services:**
 - Apertium
 - Microsoft Translator
 - MyMemory
- Several new translations.
- Improved merging of upstream changes.
- Better handle concurrent git pull and translation.
- Propagating works for fuzzy changes as well.
- Propagating works also for file upload.
- Fixed file downloads while using FastCGI (and possibly others).

8.21 weblate 0.4

Released on February 8th 2012.

- Added usage guide to documentation.
- Fixed API hooks not to require CSRF protection.

8.22 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.

8.23 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.

8.24 weblate 0.1

Released on February 6th 2012.

- Initial release.

There are dozens of ways to contribute to Weblate. We welcome any help, be it coding help, graphics design, documentation or sponsorship.

9.1 Code and development

Weblate is being developed on GitHub <<https://github.com/nijel/weblate>>. You are welcome to fork the code and open pull requests. Patches in any other form are welcome as well. The code should follow PEP-8 coding guidelines.

We do write testsuite for our code, so please add testcases for any new functionality and verify that it works. You can see current test results on Travis <<https://travis-ci.org/nijel/weblate>> and coverage on Codecov <<https://codecov.io/github/nijel/weblate>>.

9.2 Starting with our codebase

If you are looking for some bugs which should be good for starting with our codebase, you can find them labelled with *newbie* tag:

<https://github.com/nijel/weblate/labels/newbie>

9.3 Earning money by coding

We're using Bountysource to fund our development, you can participate on this as well by implementing issues with bounties:

<https://github.com/nijel/weblate/labels/bounty>

9.4 Translating

Weblate is being translated using Weblate on <<https://hosted.weblate.org/>>, feel free to join us in effort to make Weblate available in as many world languages as possible.

Copyright (C) 2012 - 2015 Michal Čihař <michal@cihar.com>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

Indices and tables

- `genindex`
- `search`

/exports

GET /exports/rss/, [124](#)
GET /exports/rss/(string:project)/, [124](#)
GET /exports/rss/(string:project)/(string:component)/,
[124](#)
GET /exports/rss/(string:project)/(string:component)/(string:language)/,
[124](#)
GET /exports/rss/language/(string:language)/,
[124](#)
GET /exports/stats/(string:project)/(string:component)/,
[122](#)

/hooks

GET /hooks/update/(string:project)/, [121](#)
GET /hooks/update/(string:project)/(string:component)/,
[121](#)
POST /hooks/bitbucket/, [122](#)
POST /hooks/github/, [121](#)
POST /hooks/gitlab/, [121](#)

СИМВОЛЫ

.Net Resource
 file format, 119
переменная окружения
 WL_FILE_FORMAT, 72
 WL_FILEMASK, 72
 WL_LANGUAGE, 72
 WL_PATH, 72
 WL_REPO, 72
 WL_VCS, 72

A

Android
 file format, 117
ANONYMOUS_USER_NAME
 setting, 81
Apple strings
 file format, 118
AUTO_LOCK
 setting, 81
AUTO_LOCK_TIME
 setting, 81
AUTOFIX_LIST
 setting, 82

B

BACKGROUND_HOOKS
 setting, 82
bilingual
 translation, 115

C

changesite
 django-admin command, 100
CHECK_LIST
 setting, 82
checkgit
 django-admin command, 101
cleanuptrans

 django-admin command, 101
Comma separated values
 file format, 119
commit_pending
 django-admin command, 101
commitgit
 django-admin command, 101
createadmin
 django-admin command, 102
CSV
 file format, 119

D

DATA_DIR
 setting, 83
DEFAULT_COMMITER_EMAIL
 setting, 83
DEFAULT_COMMITER_NAME
 setting, 83
django-admin command
 changesite, 100
 checkgit, 101
 cleanuptrans, 101
 commit_pending, 101
 commitgit, 101
 createadmin, 102
 dumpuserdata, 102
 import_project, 102
 importuserdata, 103
 importusers, 103
 list_ignored_checks, 103
 list_versions, 103
 loadpo, 104
 lock_translation, 104
 pushgit, 104
 rebuild_index, 104
 setupgroups, 105
 setuplang, 105
 unlock_translation, 105
 update_index, 104

- updatechecks, 105
- updategit, 105
- dumpuserdata
 - django-admin command, 102

E

- ENABLE_AVATARS
 - setting, 83
- ENABLE_HOOKS
 - setting, 84
- ENABLE_HTTPS
 - setting, 84
- ENABLE_SHARING
 - setting, 84

F

- file format
 - .Net Resource, 119
 - Android, 117
 - Apple strings, 118
 - Comma separated values, 119
 - CSV, 119
 - Gettext, 115
 - Java properties, 117
 - JSON, 119
 - PHP strings, 118
 - PO, 115
 - Qt, 117
 - RESX, 119
 - string resources, 117
 - TS, 117
 - XLIFF, 116

G

- Gettext
 - file format, 115
- GIT_ROOT
 - setting, 84
- GITHUB_USERNAME
 - setting, 84
- GOOGLE_ANALYTICS_ID
 - setting, 84

H

- HIDE_REPO_CREDENTIALS
 - setting, 84

I

- import_project
 - django-admin command, 102
- importuserdata
 - django-admin command, 103
- importusers

- django-admin command, 103
- iPad
 - translation, 118
- iPhone
 - translation, 118

J

- Java properties
 - file format, 117
- JSON
 - file format, 119

L

- LAZY_COMMITS
 - setting, 84
- list_ignored_checks
 - django-admin command, 103
- list_versions
 - django-admin command, 103
- loadpo
 - django-admin command, 104
- LOCK_TIME
 - setting, 85
- lock_translation
 - django-admin command, 104
- LOGIN_REQUIRED_URLS
 - setting, 85
- LOGIN_REQUIRED_URLS_EXCEPTIONS
 - setting, 85

M

- MACHINE_TRANSLATION_SERVICES
 - setting, 85
- monolingual
 - translation, 115
- MT_APERTIUM_KEY
 - setting, 86
- MT_GOOGLE_KEY
 - setting, 86
- MT_MICROSOFT_ID
 - setting, 86
- MT_MICROSOFT_SECRET
 - setting, 86
- MT_MYMEMORY_EMAIL
 - setting, 86
- MT_MYMEMORY_KEY
 - setting, 86
- MT_MYMEMORY_USER
 - setting, 87
- MT_TMSERVER
 - setting, 87

N

- NEARBY_MESSAGES

setting, 87

O

OFFLOAD_INDEXING

setting, 87

P

PHP strings

file format, 118

PIWIK_SITE_ID

setting, 87

PIWIK_URL

setting, 87

PO

file format, 115

POST_ADD_SCRIPTS

setting, 88

POST_COMMIT_SCRIPTS

setting, 88

POST_PUSH_SCRIPTS

setting, 89

POST_UPDATE_SCRIPTS

setting, 88

PRE_COMMIT_SCRIPTS

setting, 88

pushgit

django-admin command, 104

Q

Qt

file format, 117

R

rebuild_index

django-admin command, 104

REGISTRATION_CAPTCHA

setting, 89

REGISTRATION_OPEN

setting, 89

RESX

file format, 119

S

SELF_ADVERTISEMENT

setting, 89

setting

ANONYMOUS_USER_NAME, 81

AUTO_LOCK, 81

AUTO_LOCK_TIME, 81

AUTOFIX_LIST, 82

BACKGROUND_HOOKS, 82

CHECK_LIST, 82

DATA_DIR, 83

DEFAULT_COMMITER_EMAIL, 83

DEFAULT_COMMITER_NAME, 83

ENABLE_AVATARS, 83

ENABLE_HOOKS, 84

ENABLE_HTTPS, 84

ENABLE_SHARING, 84

GIT_ROOT, 84

GITHUB_USERNAME, 84

GOOGLE_ANALYTICS_ID, 84

HIDE_REPO_CREDENTIALS, 84

LAZY_COMMITS, 84

LOCK_TIME, 85

LOGIN_REQUIRED_URLS, 85

LOGIN_REQUIRED_URLS_EXCEPTIONS, 85

MACHINE_TRANSLATION_SERVICES, 85

MT_APERTIUM_KEY, 86

MT_GOOGLE_KEY, 86

MT_MICROSOFT_ID, 86

MT_MICROSOFT_SECRET, 86

MT_MYMEMORY_EMAIL, 86

MT_MYMEMORY_KEY, 86

MT_MYMEMORY_USER, 87

MT_TMSERVER, 87

NEARBY_MESSAGES, 87

OFFLOAD_INDEXING, 87

PIWIK_SITE_ID, 87

PIWIK_URL, 87

POST_ADD_SCRIPTS, 88

POST_COMMIT_SCRIPTS, 88

POST_PUSH_SCRIPTS, 89

POST_UPDATE_SCRIPTS, 88

PRE_COMMIT_SCRIPTS, 88

REGISTRATION_CAPTCHA, 89

REGISTRATION_OPEN, 89

SELF_ADVERTISEMENT, 89

SIMPLIFY_LANGUAGES, 89

SITE_TITLE, 89

SOURCE_LANGUAGE, 89

TTF_PATH, 90

URL_PREFIX, 90

WHOOSH_INDEX, 90

setupgroups

django-admin command, 105

setuplang

django-admin command, 105

SIMPLIFY_LANGUAGES

setting, 89

SITE_TITLE

setting, 89

SOURCE_LANGUAGE

setting, 89

string resources

file format, 117

T

translation
 bilingual, [115](#)
 iPad, [118](#)
 iPhone, [118](#)
 monolingual, [115](#)

TS
 file format, [117](#)

TTF_PATH
 setting, [90](#)

U

unlock_translation
 django-admin command, [105](#)

update_index
 django-admin command, [104](#)

updatechecks
 django-admin command, [105](#)

updategit
 django-admin command, [105](#)

URL_PREFIX
 setting, [90](#)

W

WHOOSH_INDEX
 setting, [90](#)

X

XLIFF
 file format, [116](#)