

---

# Weblate Documentation

*Выпуск 0.8*

Michal Čihař

июл. 20, 2020



<b>1</b>	<b>About Weblate</b>	<b>3</b>
1.1	Project goals . . . . .	3
1.2	Project name . . . . .	3
1.3	Project website . . . . .	3
1.4	Authors . . . . .	3
<b>2</b>	<b>Usage guide</b>	<b>5</b>
2.1	Registration . . . . .	5
2.2	Profile information . . . . .	5
2.3	Projects structure . . . . .	5
2.4	Translation links . . . . .	6
2.5	Translating . . . . .	6
2.6	Dictionary . . . . .	6
2.7	Suggestions . . . . .	6
2.8	Machine translation . . . . .	6
2.9	Checks . . . . .	7
<b>3</b>	<b>Installation instructions</b>	<b>9</b>
3.1	Requirements . . . . .	9
3.2	Installation . . . . .	9
3.3	Running server . . . . .	9
3.4	Upgrading . . . . .	10
<b>4</b>	<b>Configuration</b>	<b>11</b>
<b>5</b>	<b>Administration</b>	<b>13</b>
5.1	Adding new resources . . . . .	13
5.2	Project . . . . .	13
5.3	Subproject . . . . .	14
5.4	Updating repositories . . . . .	14
5.5	Pushing changes . . . . .	14
5.6	Interacting with others . . . . .	14
5.7	Access control . . . . .	15
5.8	Lazy commits . . . . .	15
5.9	Customizing checks . . . . .	16
<b>6</b>	<b>Management commands</b>	<b>19</b>

<b>7</b>	<b>Frequently Asked Questions</b>	<b>21</b>
7.1	Requests sometimes fail with too many open files error . . . . .	21
7.2	Fulltext search is too slow . . . . .	21
7.3	Does Weblate support other VCS than Git? . . . . .	21
<b>8</b>	<b>Changes</b>	<b>23</b>
8.1	weblate 0.8 . . . . .	23
8.2	weblate 0.7 . . . . .	24
8.3	weblate 0.6 . . . . .	24
8.4	weblate 0.5 . . . . .	24
8.5	weblate 0.4 . . . . .	24
8.6	weblate 0.3 . . . . .	25
8.7	weblate 0.2 . . . . .	25
8.8	weblate 0.1 . . . . .	25
<b>9</b>	<b>License</b>	<b>27</b>
<b>10</b>	<b>Indices and tables</b>	<b>29</b>
	<b>Алфавитный указатель</b>	<b>31</b>

Contents:



## 1.1 Project goals

Minimalistic web based translation with direct commit to git on each translation made. 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 <http://demo.weblate.org/>. This documentation can be browsed on <http://weblate.readthedocs.org/>.

## 1.4 Authors

This tool was written by Michal Čihař [michal@cihar.com](mailto:michal@cihar.com).





This document briefly covers how to translate application using Weblate.

## 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 Profile information

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

In preferences, you can choose user interface language, languages which you prefer to translate (list of these will be offered to you on main page) and secondary languages, whose translations will be shown to you while translating.

## 2.3 Projects structure

Each project can contain various subprojects. The reason for this structure is that all subprojects in a project are expected to have a lot in common. Whenever translation is made in single subproject, it is automatically propagated to others within same project (this is especially useful when translating more version of same project).

## 2.4 Translation links

Once you navigate to 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.

## 2.5 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.

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.

## 2.6 Dictionary

Each project can have assigned dictionary 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.

## 2.7 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.

## 2.8 Machine translation

Based on configuration and your language, Weblate provides buttons for following machine translation tools.

### 2.8.1 MyMemory

Huge translation memory with machine translation.

**См.также:**

<http://mymemory.translated.net/>

### 2.8.2 Apertium

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

**См.также:**

<http://www.apertium.org/>

### 2.8.3 Microsoft Translator

Machine translation service provided by Microsoft.

**См.также:**

<http://www.microsofttranslator.com/>

## 2.9 Checks

Weblate does wide range of consistency checks on translated 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.

### 2.9.1 Not translated

The source and translated strings are same at least in one of plural forms. This checks ignores some strings which are quite usually same in all languages.

### 2.9.2 Starting newline

Source and translated do not both start with a newline.

### 2.9.3 Trailing newline

Source and translated do not both end with a newline.

### 2.9.4 Trailing space

Source and translated do not both end with a space.

### 2.9.5 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).

### 2.9.6 Trailing colon

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

### 2.9.7 Trailing question

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

### 2.9.8 Trailing exclamation

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

### 2.9.9 Python format

Python format string does not match source.

### 2.9.10 PHP format

PHP format string does not match source.

### 2.9.11 C format

C format string does not match source.

### 2.9.12 Missing plurals

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

### 2.9.13 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.

## 3.1 Requirements

**Django** <https://www.djangoproject.com/>

**Translate-toolkit** <http://translate.sourceforge.net/wiki/toolkit/index>

**GitPython (>= 0.3)** <https://github.com/gitpython-developers/GitPython>

**Django-registration** <https://bitbucket.org/ubernostrum/django-registration/>

**Whoosh** <http://bitbucket.org/mchaput/whoosh/>

## 3.2 Installation

Install all required components (see above), adjust `settings.py` and then run `./manage.py syncdb` to create database structure. Now you should be able to create translation projects using admin interface. You probably also want to run `./manage.py setuplang` to get default list of languages and `./manage.py setupgroups` to initialize default groups.

**См.также:**

*Access control*

## 3.3 Running server

Running Weblate is not different from running any other Django based application.

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

`/media` Serves `media` directory from Weblate.

`/static/admin` Serves media files for Django admin interface (eg. `/usr/share/pyshared/django/contrib/admin/media/`).

Additionally you should setup rewrite rule to serve `media/favicon.ico` as `favicon.ico`.

**См.также:**

<https://docs.djangoproject.com/en/1.3/howto/deployment/>

### 3.3.1 Sample configuration for Lighttpd

The configuration for Lighttpd web server might look like following:

```
fastcgi.server = (
    "/weblate.fcgi" => (
        "main" => (
            "socket" => "/var/run/django/weblate.socket",
            "check-local" => "disable",
        )
    ),
)
alias.url = (
    "/media" => "/var/lib/django/weblate/media/",
    "/static/admin" => "/usr/share/pyshared/django/contrib/admin/static/admin/",
)
url.rewrite-once = (
    "^(/*media.*)$" => "$1",
    "^(/*static.*)$" => "$1",
    "^/*favicon\\.ico$" => "/media/favicon.ico",
    "^/*robots\\.txt$" => "/media/robots.txt",
    "^(/.*)$" => "/weblate.fcgi$1",
)
expire.url = (
    "/media/" => "access 1 months",
    "/static/" => "access 1 months",
    "/favicon.ico" => "access 1 months",
)
```

## 3.4 Upgrading

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.

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.

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.

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

**CHECK\_LIST**

List of consistency checks to perform on translation.

**См.также:**

*Checks, Customizing checks*

**COMMIT\_MESSAGE**

Message used on each commit Weblate does.

**ENABLE\_HOOKS**

Whether to enable anonymous remote hooks.

**См.также:**

*Interacting with others*

**GIT\_ROOT**

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

**LAZY\_COMMITS**

Delay creating Git 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*

**MT\_APERTIUM\_KEY**

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

**MT\_MICROSOFT\_KEY**

API key for Microsoft Translator service, you can register at <http://www.bing.com/developers/createapp.aspx>

**NEARBY\_MESSAGES**

How many messages around current one to show during translating.

**SIMILAR\_MESSAGES**

Number of similar messages to lookup. This is not a hard limit, just a number Weblate tries to find if it is possible.

**SITE\_TITLE**

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

**WHOOSH\_INDEX**

Directory where Whoosh fulltext indices will be stored. Defaults to `whoosh-index` subdirectory.

**См.также:**

<https://docs.djangoproject.com/en/1.3/ref/settings/>



Administration of Weblate is done through standard Django admin interface, which is available under `/admin/` URL.

## 5.1 Adding new resources

All translation resources need to be available as Git repositories and are organized as `project/subproject` structure.

Weblate supports wide range of translation formats supported by translate toolkit, for example:

- GNU Gettext
- XLIFF
- Java properties
- Windows RC files
- Qt Linguist `.ts`
- Symbian localization files
- CSV
- INI

**См.также:**

<http://translate.sourceforge.net/wiki/toolkit/formats>

## 5.2 Project

To add new resource to translate, you need to create translation project first. The project is sort of shelf, in which real translations are folded. All subprojects in same project share suggestions and dictionary, also the

translations are automatically propagated through the all subproject in single project.

## 5.3 Subproject

Subproject is real resource for translating. You enter Git repository location and file mask which files to translate and Weblate automatically fetches the Git and finds all translated files.

---

**Примечание:** As setup of translation project includes fetching Git repositories, you might want to preseed these, repos are stored in path defined by `GIT_ROOT` in `settings.py` in `<project>/<subproject>` directories.

---

## 5.4 Updating repositories

You should set up some way how backend repositories are updated from their source. You can either use hooks (see *Interacting with others*) or just regularly run `./manage.py updategit --all`.

With Gettext po files, you might be often bitten by conflict in PO file headers. To avoid it, you can use shipped merge driver (`scripts/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/scripts/git-merge-gettext-po %0 %A %B
```

And enable it's use by defining proper attributes in given repository (eg. in `.git/info/attribute`):

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

---

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

---

**См.также:**

<http://www.no-ack.org/2010/12/writing-git-merge-driver-for-po-files.html>

## 5.5 Pushing changes

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

I case you will use SSH for pushing, you need to have key without passphrase (or use ssh-agent for Django) and the remote server needs to be verified by you first, otherwise push will fail.

## 5.6 Interacting with others

You can trigger update of underlaying git repository for every subproject or project by accessing URL `/hooks/update/project/subproject/` or `/hooks/update/project/`.

For GitHub, there is a special URL `/hooks/github/`, which parses GitHub notifications and updates related projects automatically.

---

**Примечание:** The GitHub notification relies on Git repository urls you use to be in form `git://github.com/owner/repo.git`, otherwise automatic detection of used repository will fail.

---

## 5.7 Access control

Weblate uses privileges system based on Django. It defines following extra privileges:

- Can upload translation [Users, Managers]
- Can overwrite with translation upload [Users, Managers]
- Can define author of translation upload [Managers]
- Can force committing of translation [Managers]
- Can update translation from git [Managers]
- Can push translations to remote git [Managers]
- Can do automatic translation using other project strings [Managers]
- Can save translation [Users, Managers]
- Can accept suggestion [Users, Managers]
- Can accept suggestion [Users, Managers]
- Can import dictionary [Users, Managers]
- Can add dictionary [Users, Managers]
- Can change dictionary [Users, Managers]
- Can delete dictionary [Users, Managers]

The default setup (after you run `./manage.py setupgroups`) consists of two groups *Users* and *Managers* which have privileges as described above. All new users are automatically added to *Users* group.

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.

## 5.8 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 do implicit sync to get Git repository in sync (you can do this in admin interface).

The changes are in this mode committed once one of following conditions happen:

- somebody else works on the translation
- merge from upstream occurs
- import of translation happens

- translation for a language is completed

## 5.9 Customizing checks

Weblate comes with wide range of consistency checks (see *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.

### 5.9.1 Checking translation text does not contain «foo»

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

```
from trans.checks import Check
from django.utils.translation import ugettext_lazy as _

class FooCheck(Check):

    # 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, flags, language, unit):
        return 'foo' in target
```

### 5.9.2 Checking Czech translation text plurals differ

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

```
from trans.checks import Check
from django.utils.translation import ugettext_lazy as _

class PluralCzechCheck(Check):

    # 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(self, sources, targets, flags, language, unit):
```

(continues on next page)

(продолжение с предыдущей страницы)

```
if self.is_language(language, ['cs']):  
    return targets[1] == targets[2]  
return False
```



---

## Management commands

---

The `./manage.py` is extended with following commands:

**checkgit**

Prints current state of backend git repository.

You can either define which subproject to check (eg. `weblate/master`) or use `--all` to check all existing subprojects.

**commitgit**

Commits any possible pending changes to backend git repository.

You can either define which subproject to check (eg. `weblate/master`) or use `--all` to check all existing subprojects.

**cleanuptrans**

Cleanups orphaned checks and translation suggestions.

**loadpo**

Reloads translations from disk (eg. in case you did some updates in Git repository).

**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.

**setupgroups**

Configures default groups and (if called with `--move`) assigns all users to default group.

**См. также:**

*Access control*

**setuplang**

Setups list of languages (it has own list and all defined in `translate-toolkit`).

**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 subproject to update (eg. `weblate/master`) or use `--all` to update all existing subprojects.

**updategit**

Fetches remote Git repositories and updates internal cache.

You can either define which project or subproject to update (eg. `weblate/master`) or use `--all` to update all existing subprojects.



---

## Frequently Asked Questions

---

### 7.1 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:

```
cd repos
for d in */* ; do
    pushd $d
    git gc
    popd
done
```

### 7.2 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 rebuild it from scratch time to time:

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

### 7.3 Does Weblate support other VCS than Git?

Not currently. Weblate requires distributed VCS and could be probably adjusted to work with anything else than Git, but somebody has to implement this support.



## 8.1 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.

## 8.2 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.

## 8.3 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.

## 8.4 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).

## 8.5 weblate 0.4

Released on February 8th 2012.

- Added usage guide to documentation.
- Fixed API hooks not to require CSRF protection.

## 8.6 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.

## 8.7 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.

## 8.8 weblate 0.1

Released on February 6th 2012.

- Initial release.



Copyright (C) 2012 Michal Čihař <[michal@cihar.com](mailto: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/>>.





- `genindex`
- `modindex`
- `search`



## СИМВОЛЫ

Опция командной строки `./manage.py`

- `checkgit`, 19
- `cleanuptrans`, 19
- `commitgit`, 19
- `loadpo`, 19
- `rebuild_index`, 19
- `setupgroups`, 19
- `setuplang`, 19
- `updatechecks`, 19
- `updategit`, 20

переменная окружения

- `CHECK_LIST`, 11, 16
- `COMMIT_MESSAGE`, 11
- `ENABLE_HOOKS`, 11
- `GIT_ROOT`, 11, 14
- `LAZY_COMMITS`, 11, 15
- `MT_APERTIUM_KEY`, 11
- `MT_MICROSOFT_KEY`, 11
- `NEARBY_MESSAGES`, 11
- `SIMILAR_MESSAGES`, 12
- `SITE_TITLE`, 12
- `WHOOSH_INDEX`, 12

## C

`CHECK_LIST`, 16

`checkgit`

Опция командной строки `./manage.py`, 19

`cleanuptrans`

Опция командной строки `./manage.py`, 19

`commitgit`

Опция командной строки `./manage.py`, 19

## G

`GIT_ROOT`, 14

## L

`LAZY_COMMITS`, 15

`loadpo`

Опция командной строки `./manage.py`, 19

## R

`rebuild_index`

Опция командной строки `./manage.py`, 19

## S

`setupgroups`

Опция командной строки `./manage.py`, 19

`setuplang`

Опция командной строки `./manage.py`, 19

## U

`updatechecks`

Опция командной строки `./manage.py`, 19

`updategit`

Опция командной строки `./manage.py`, 20