The Weblate Manual

Выпуск 4.2

Michal Čihař

авг. 18, 2020

Пользовательская документация

1 Пользовательская документация **1**

1.1 Основы Weblate . . . 1
1.2 Регистрация и профиль пользователя . . . 1
1.3 Перевод с помощью Weblate . . . 10
1.4 Скачивание и загрузка переводов . . . 20
1.5 Проверки и исправления . . . 22
1.6 Поиск . . . 38
1.7 Руководство разработчика приложений . . . 42
1.8 Рабочие процессы перевода . . . 63
1.9 Часто задаваемые вопросы . . . 66
1.10 Поддерживаемые форматы файлов . . . 74
1.11 Интеграция с системой контроля версий . . . 93
1.12 Weblate's REST API . . . 100
1.13 Клиент Weblate . . . 141
1.14 Python API Weblate'a . . . 145

2 Документация для администратора **147**

2.1 Инструкции по настройке . . . 147
2.2 Weblate deployments . . . 202
2.3 Upgrading Weblate . . . 202
2.4 Резервное копирование и перенос Weblate . . . 207
2.5 Аутентификация . . . 212
2.6 Управление доступом . . . 221
2.7 Переводческие проекты . . . 229
2.8 Определения языка . . . 245
2.9 Непрерывная локализация . . . 246
2.10 Лицензирование переводов . . . 255
2.11 Процесс перевода . . . 257
2.12 Проверки и исправления . . . 263
2.13 Машинный перевод . . . 271
2.14 Надстройки . . . 278
2.15 Память переводов . . . 287
2.16 Конфигурация . . . 289
2.17 Пример конфигурации . . . 312
2.18 Команды управления . . . 327
2.19 Объявления . . . 337
2.20 Списки компонентов . . . 339
2.21 Optional Weblate modules . . . 341
2.22 Customizing Weblate . . . 345
2.23 Интерфейс управления . . . 347
2.24 Getting support for Weblate . . . 354

2.25 Правовые документы . . . 355

3 Документация для участников **357**
3.1 Внесение вклада в Weblate . . . 357
3.2 Starting contributing code to Weblate . . . 358
3.3 Weblate source code . . . 362
3.4 Debugging Weblate . . . 363
3.5 Weblate internals . . . 364
3.6 Weblate frontend . . . 365
3.7 Reporting issues in Weblate . . . 366
3.8 Weblate testsuite and continuous integration . . . 366
3.9 Data schemas . . . 368
3.10 Releasing Weblate . . . 372
3.11 О Weblate . . . 372
3.12 Лицензия . . . 373

4 История изменений **374**
4.1 Weblate 4.2 . . . 374
4.2 Weblate 4.1.1 . . . 375
4.3 Weblate 4.1 . . . 375
4.4 Weblate 4.0.4 . . . 376
4.5 Weblate 4.0.3 . . . 377
4.6 Weblate 4.0.2 . . . 377
4.7 Weblate 4.0.1 . . . 377
4.8 Weblate 4.0 . . . 378
4.9 Серия Weblate 3.x . . . 378
4.10 Серия Weblate 2.x . . . 390
4.11 Серия Weblate 1.x . . . 401
4.12 Серия Weblate 0.x . . . 405

Содержание модулей Python **409**

HTTP Routing Table **410**

Алфавитный указатель **412**

ГЛАВА 1

Пользовательская документация

1.1 Основы Weblate

1.1.1 Структура проекта

Переводы в Weblate организованы по проектам и компонентам. Каждый проект может содержать несколько компонентов, а те — переводы на отдельные языки. Компоненту соответствует один переводимый файл (например, *GNU gettext* или *Строковые ресурсы Android*). Проекты служат для организации компонентов в логические наборы (например, для группировки всех переводов, используемых в одном приложении).

Внутри каждого проекта переводы общих строк по умолчанию распространяются на другие его компоненты. Это облегчает бремя повторного и много-версионного перевода. При отключении этого распространения в *настройках компонента* для результирующих переводов, кажущихся непоследовательными, всё равно будут генерироваться ошибки.

1.2 Регистрация и профиль пользователя

1.2.1 Регистрация

По умолчанию любой может просматривать проекты, смотреть или предлагать переводы. Право сохранять изменения имеют только зарегистрированные пользователи и они упоминаются как авторы каждого сделанного ими перевода.

Вы можете зарегистрироваться, выполнив несколько простых шагов:

1. Заполните форму регистрации своими учётными данными.
2. Активируйте регистрацию, пройдя по ссылке в полученном вами электронном письме.
3. Дополнительно настройте свой профиль, выбрав, какие языки вы знаете.

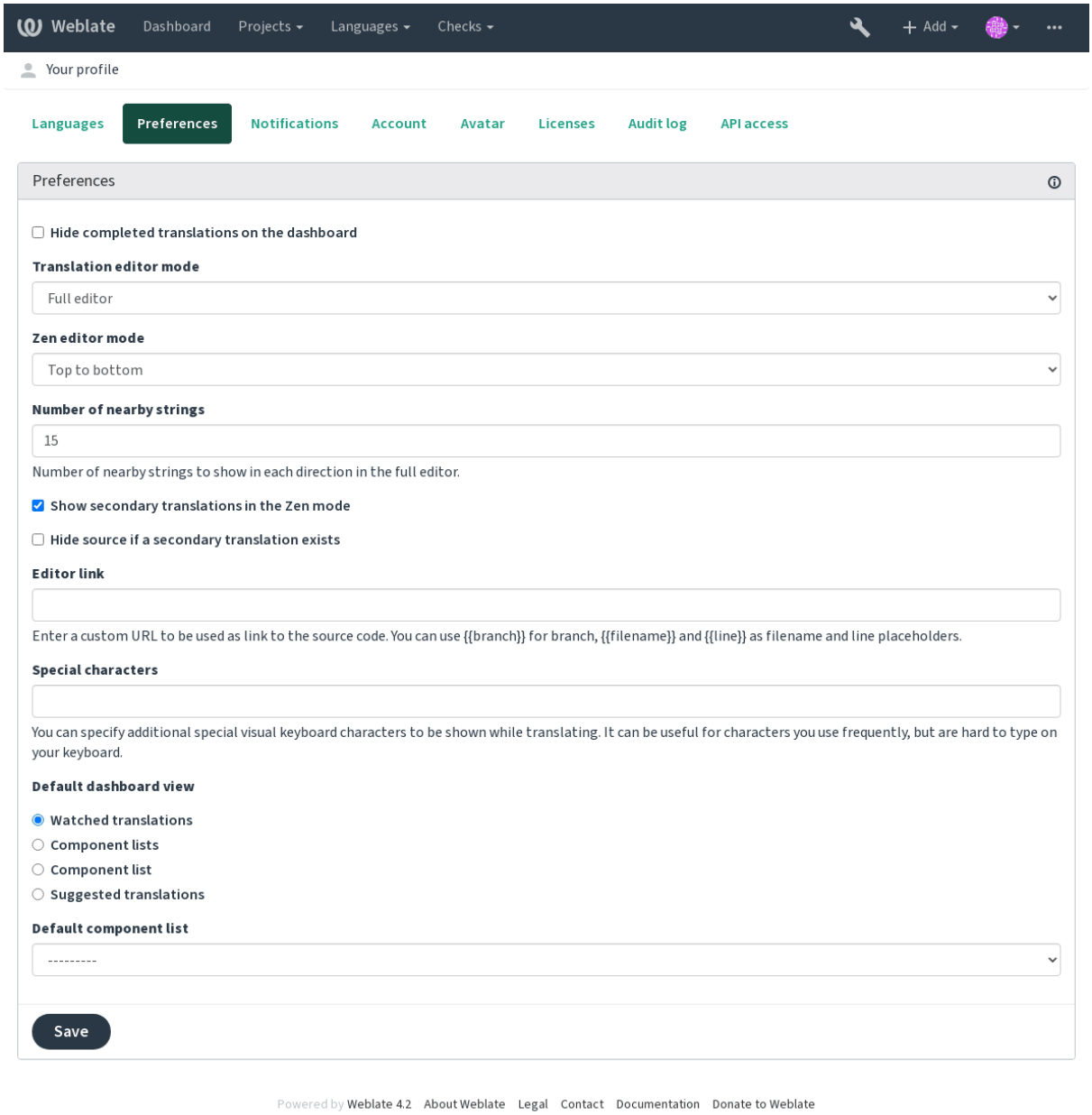
1.2.2 Панель управления

При входе в систему вы увидите сводку по проектам и их компонентам, а также информацию о завершённости их перевода.

Добавлено в версии 2.5.

По умолчанию показываются компоненты отслеживаемых вами проектов со ссылками на предпочитаемые вами языки.

Подсказка: При помощи вкладок навигации вы можете переключиться на другие представления.



Меню содержит следующие параметры:

- Ссылка *Проекты > Просмотр всех проектов* в главном меню показывает статус перевода каждого проекта на экземпляре Weblate.
- Выборе языка в главном меню *Языки* покажет статус перевода всех проектов, отфильтрованных по одному из ваших основных языков.

- На вкладке *Отслеживаемые переводы* панели управления показывается статус перевода только тех проектов, которые вы отслеживаете, отфильтрованных по вашим основным языкам.

Кроме того, в выпадающем списке может отображаться любое количество *списков компонентов*, наборов компонентов проекта, предварительно настроенных администратором Weblate, смотрите раздел *Списки компонентов*.

Вы можете настроить свой личный вид панели управления по умолчанию на вкладке *Настройки* параметров вашего профиля пользователя.

Примечание: Когда Weblate настроен для одного проекта при помощи параметра `SINGLE_PROJECT` в файле `settings.py` (смотрите раздел *Конфигурация*), панель управления показываться не будет, а пользователь вместо этого будет перенаправлен на один этот проект или компонент.

1.2.3 Профиль пользователя

Доступ к профилю пользователя осуществляется щелчком по иконке вашего пользователя в верхней правой части верхнего меню и выбором пункта *Параметры*.

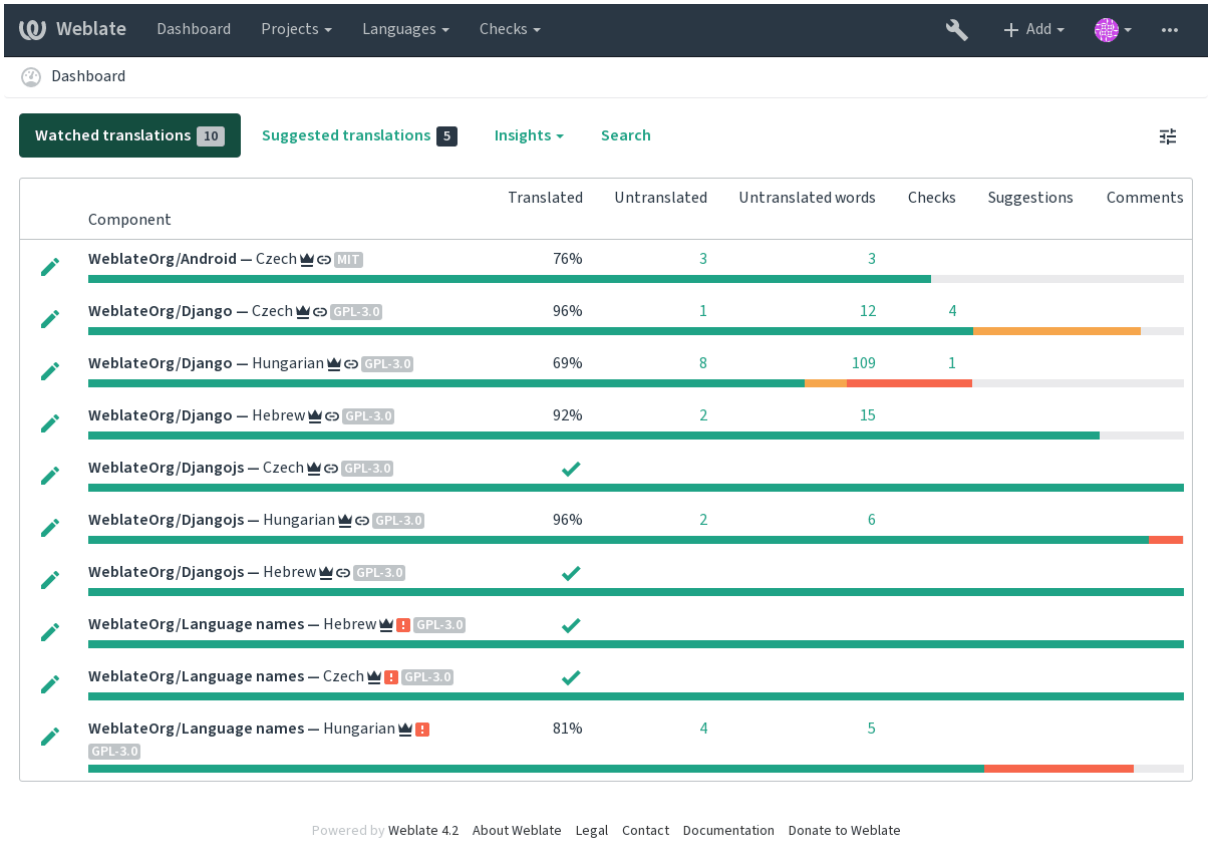
Профиль пользователя содержит ваши предпочтения. Имя и адрес электронной почты используются в коммитах системы контроля версий, поэтому держите эту информацию точной.

Примечание: Все пункты настроек, связанные с выбором языка, предлагают выбор только из языков, переводимых в настоящее время.

Подсказка: Чтобы сделать доступными и другие языки, на которые вы хотите переводить, запросите их добавление или добавьте их, нажав на соответствующую кнопку.

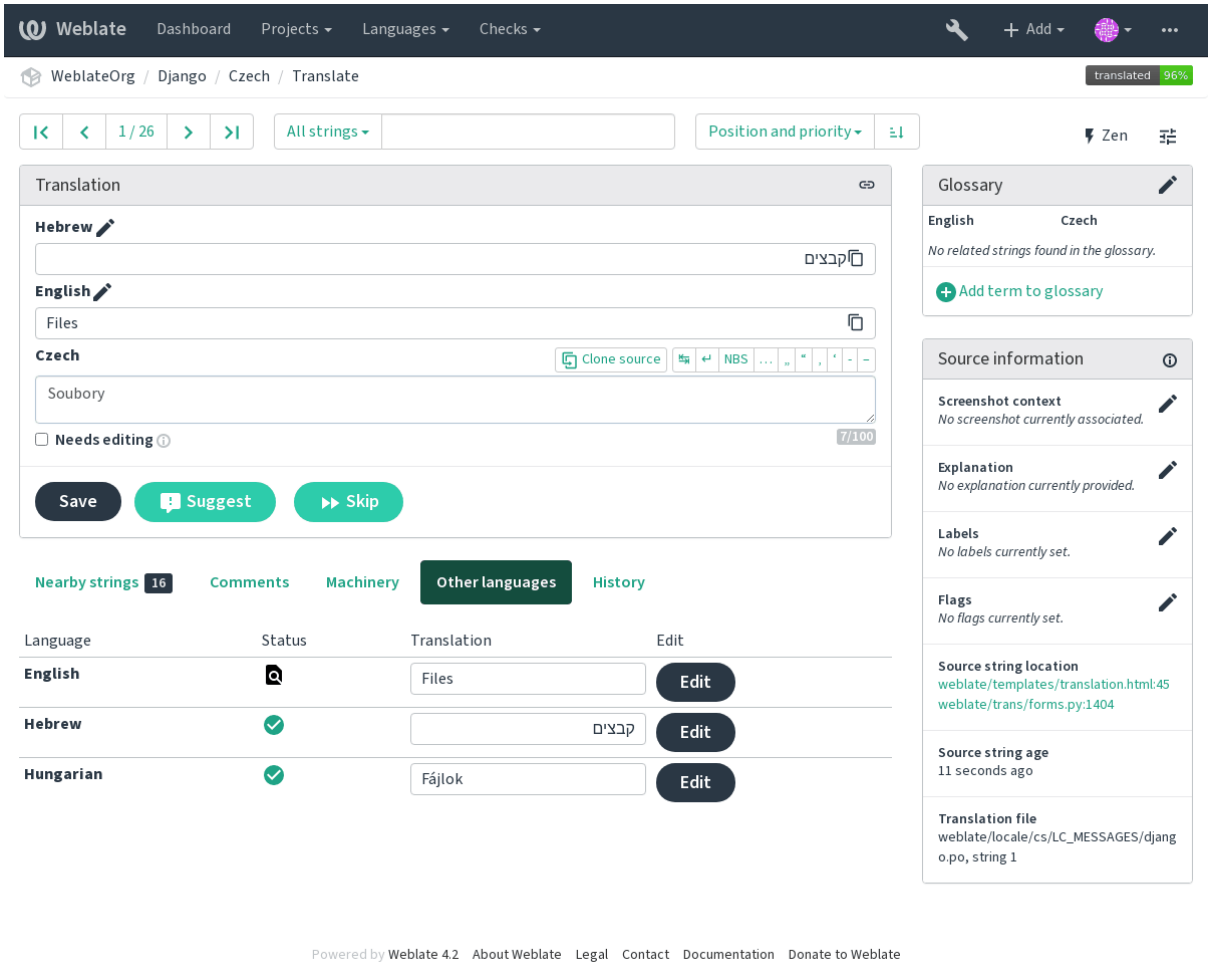
Языки перевода

Выберите, на какие языки вы предпочитаете переводить, и они будут показаны на главной странице отслеживаемых проектов, так что доступ ко всем переводам на каждый из этих языков для вас упростится.



Вспомогательные языки

Вы можете определить вспомогательные языки, которые будут вам показываться при переводе для ориентира. Пример показан на следующем рисунке, на котором вспомогательным языком выбран иврит:



Вид панели управления по умолчанию

На вкладке *Настройки* можно выбрать, какой из доступных видов панели управления будет показан по умолчанию. При выборе пункта *Список компонентов* необходимо будет выбрать из раскрывающегося списка *Список компонентов по умолчанию*, какой именно список компонентов будет отображаться.

См.также:

Списки компонентов

Аватар

Аватар может показываться для каждого пользователя (это зависит от параметра `ENABLE_AVATARS`). Эти изображения получаются от сервиса https://gravatar.com/.

Ссылка на редактор

Ссылка на исходный код показывается в web-браузере, настроенном по умолчанию в разделе *Конфигурация компонента*.

Подсказка: Установив *Ссылку на редактор*, вы будете использовать свой локальный редактор для открытия исходного кода файла из системы контроля версий с переведёнными строками. Вы можете использовать *шаблонную разметку*.

Обычно хорошим выбором является что-то вроде `editor://open/?file={{filename}}&line={{line}}`.

См.также:

Более подробную информацию о регистрации пользовательских URL-протоколов для редактора вы можете найти в документации к Nette (на английском языке).

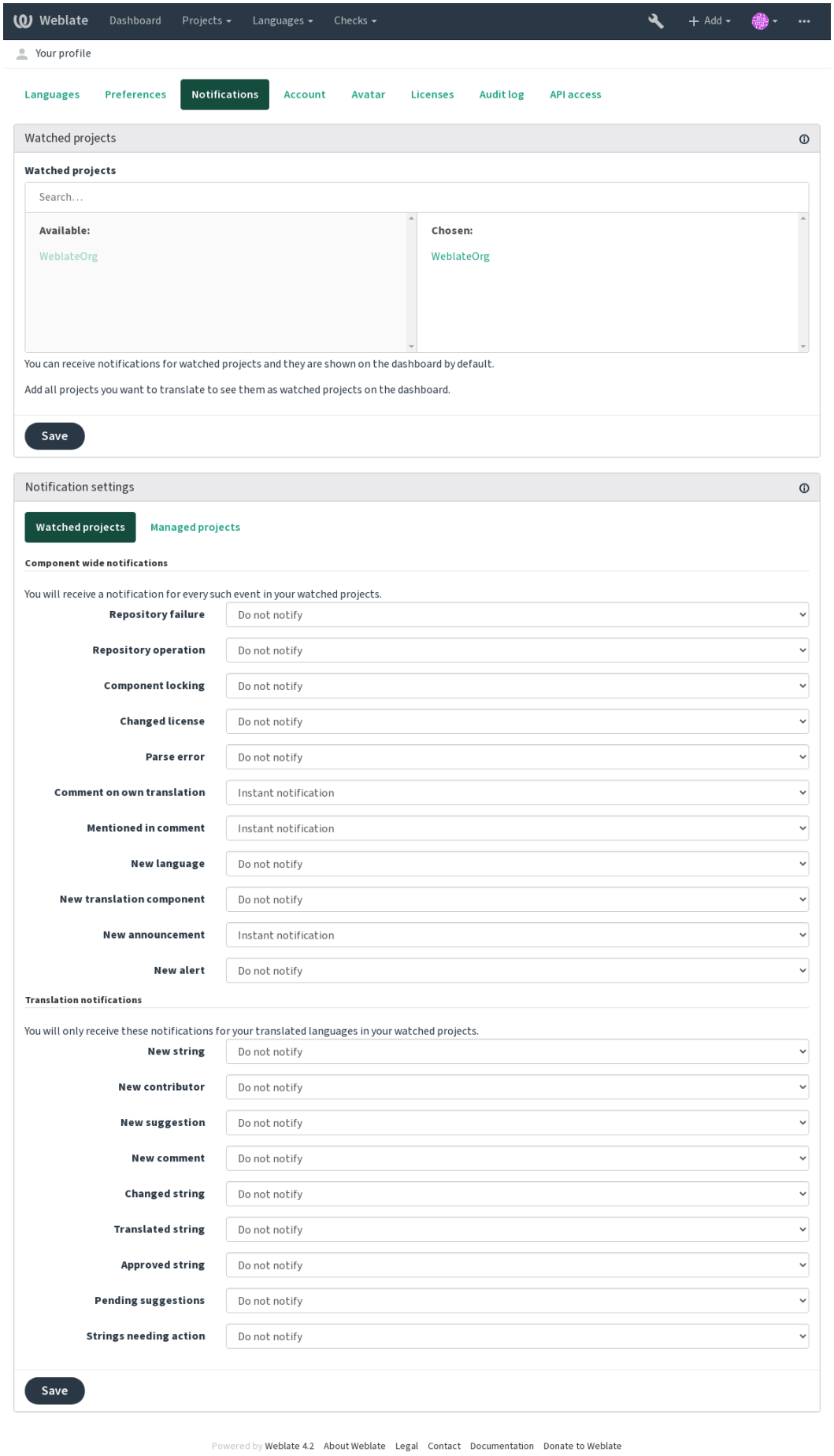
1.2.4 Уведомления

На вкладке *Уведомления* можно подписаться на различные уведомления. Уведомления по выбранным событиям для отслеживаемых или администрируемых проектов будут высылаться вам по электронной почте.

Некоторые уведомления отправляются только для событий, связанных с вашим языком (например, уведомление о новых строках для перевода), в то время как другие срабатывают на уровне компонента (например, уведомление об ошибках слияния). Эти две группы уведомлений в настройках визуально разделены.

Вы можете включать или отключать уведомления для отслеживаемых и управляемых проектов, а в дальнейшем их можно настроить (или полностью отключить) для каждого проекта и компонента. Посетите страницу компонента и выберите соответствующий пункт в меню *Отслеживание*.

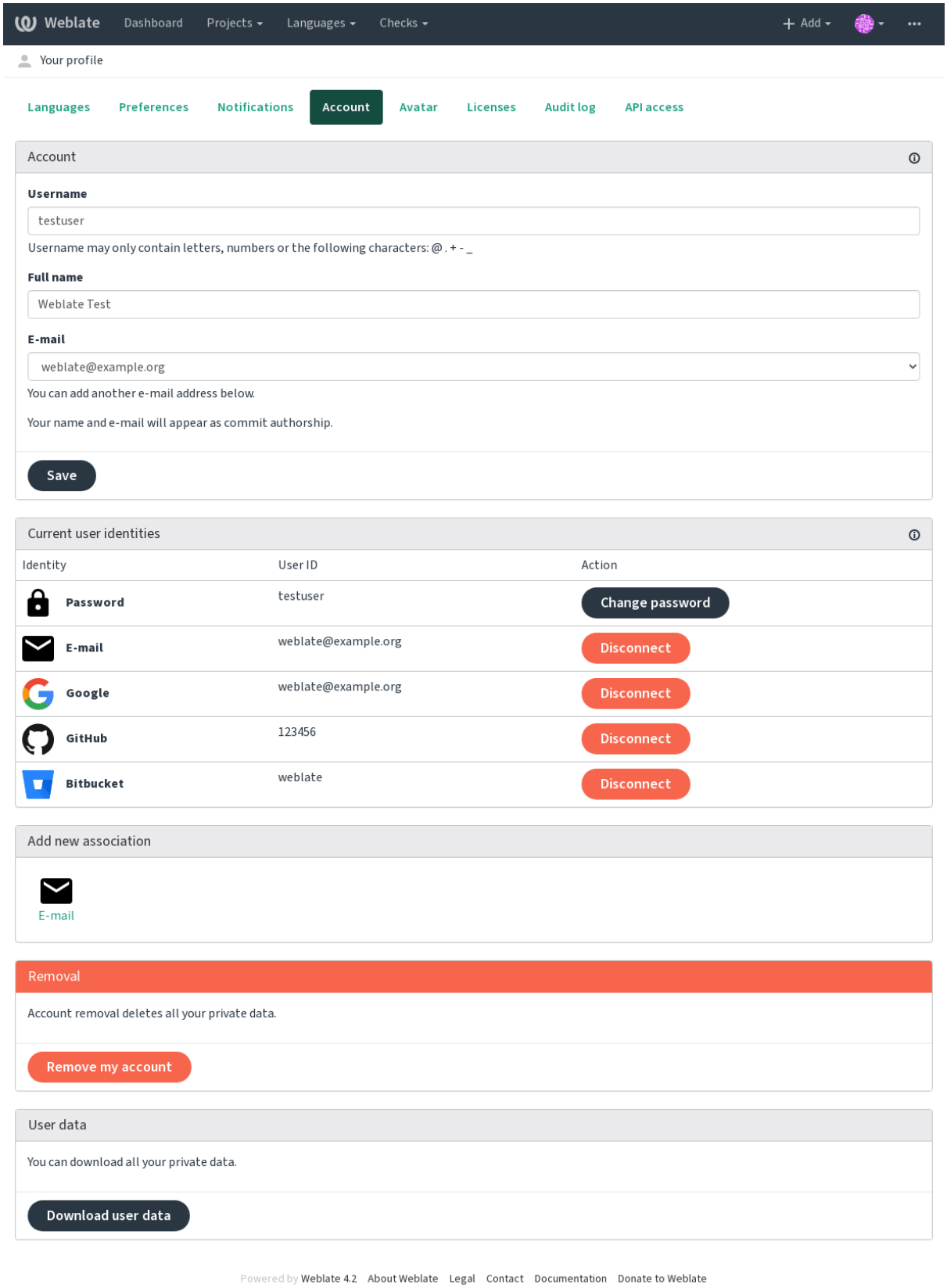
Примечание: Вы не будете получать уведомления о собственных действиях.


Weblate
Dashboard
Projects
Languages
Checks
Add
Your profile
Languages
Preferences
Notifications
Account
Avatar
Licenses
Audit log
API access
Watched projects
Watched projects
Search...
Available:
WeblateOrg
Chosen:
WeblateOrg
You can receive notifications for watched projects and they are shown on the dashboard by default.
Add all projects you want to translate to see them as watched projects on the dashboard.
Save
Notification settings
Watched projects
Managed projects
Component wide notifications
You will receive a notification for every such event in your watched projects.
Repository failure
Repository operation
Component locking
Changed license
Parse error
Comment on own translation
Mentioned in comment
New language
New translation component
New announcement
New alert
Do not notify
Instant notification
Translation notifications
You will only receive these notifications for your translated languages in your watched projects.
New string
New contributor
New suggestion
New comment
Changed string
Translated string
Approved string
Pending suggestions
Strings needing action
Save
Powered by Weblate 4.2
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.2.5 Учётная запись

The *Account* tab lets you set up basic account details, connect various services you can use to sign in into Weblate, completely remove your account, or download your user data (see *Weblate user data export*).

Примечание: Список сервисов зависит от вашей конфигурации Weblate и может включать такие популярные сайты, как GitLab, GitHub, Google, Facebook или Bitbucket, а также других провайдеров OAuth 2.0.


Weblate
Dashboard
Projects
Languages
Checks
Add
Your profile
Languages
Preferences
Notifications
Account
Avatar
Licenses
Audit log
API access
Account
Username
testuser
Username may only contain letters, numbers or the following characters: @ . + - _
Full name
Weblate Test
E-mail
weblate@example.org
You can add another e-mail address below.
Your name and e-mail will appear as commit authorship.
Save
Current user identities
Identity
User ID
Action
Password
testuser
Change password
E-mail
weblate@example.org
Disconnect
Google
weblate@example.org
Disconnect
GitHub
123456
Disconnect
Bitbucket
weblate
Disconnect
Add new association
E-mail
Removal
Account removal deletes all your private data.
Remove my account
User data
You can download all your private data.
Download user data
Powered by Weblate 4.2
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.2.6 Журнал аудита

Журнал аудита отслеживает действия, произведённые с вашей учётной записи. Для каждого важного действия с вашей учётной записью в нём регистрируются IP-адрес и браузер. Критические действия вдобавок инициируют уведомление на основной адрес электронной почты.

См.также:

Работа за обратным прокси

1.3 Перевод с помощью Weblate

Благодарим вас за проявленный интерес к переводу с помощью Weblate. Проекты могут быть настроены либо на прямой перевод, либо на принятие предложений, внесённых пользователями без учётных записей.

В целом, существует два режима перевода:

- Проект принимает прямые переводы
- Проект принимает только предложения, которые автоматически подтверждаются по достижении определённого числа голосов

Для получения более подробной информации о процессе перевода, пожалуйста, ознакомьтесь с разделом *Рабочие процессы перевода*.

Варианты видимости проекта перевода:

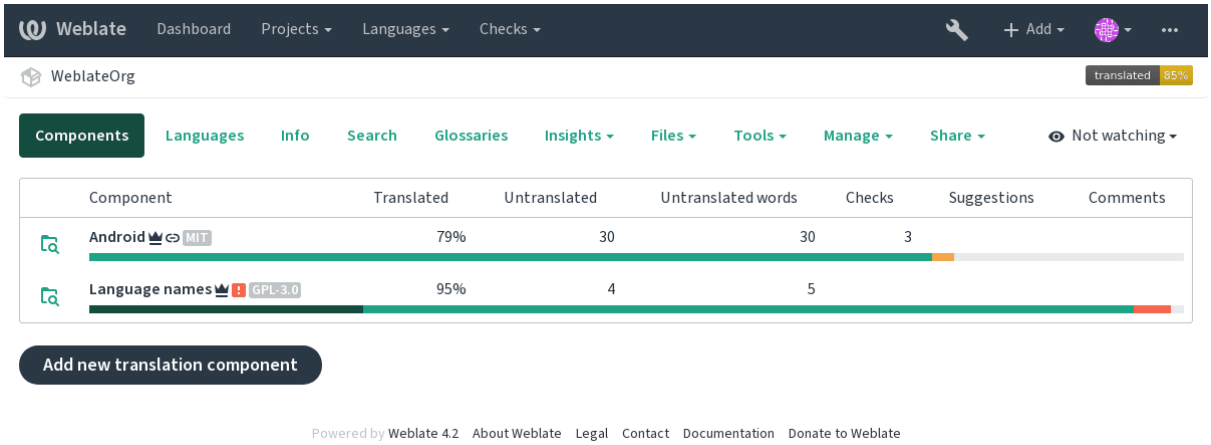
- Видимый публично, в который внести свой вклад может любой желающий
- Видимый только для определённой группы переводчиков

См.также:

Управление доступом, *Рабочие процессы перевода*

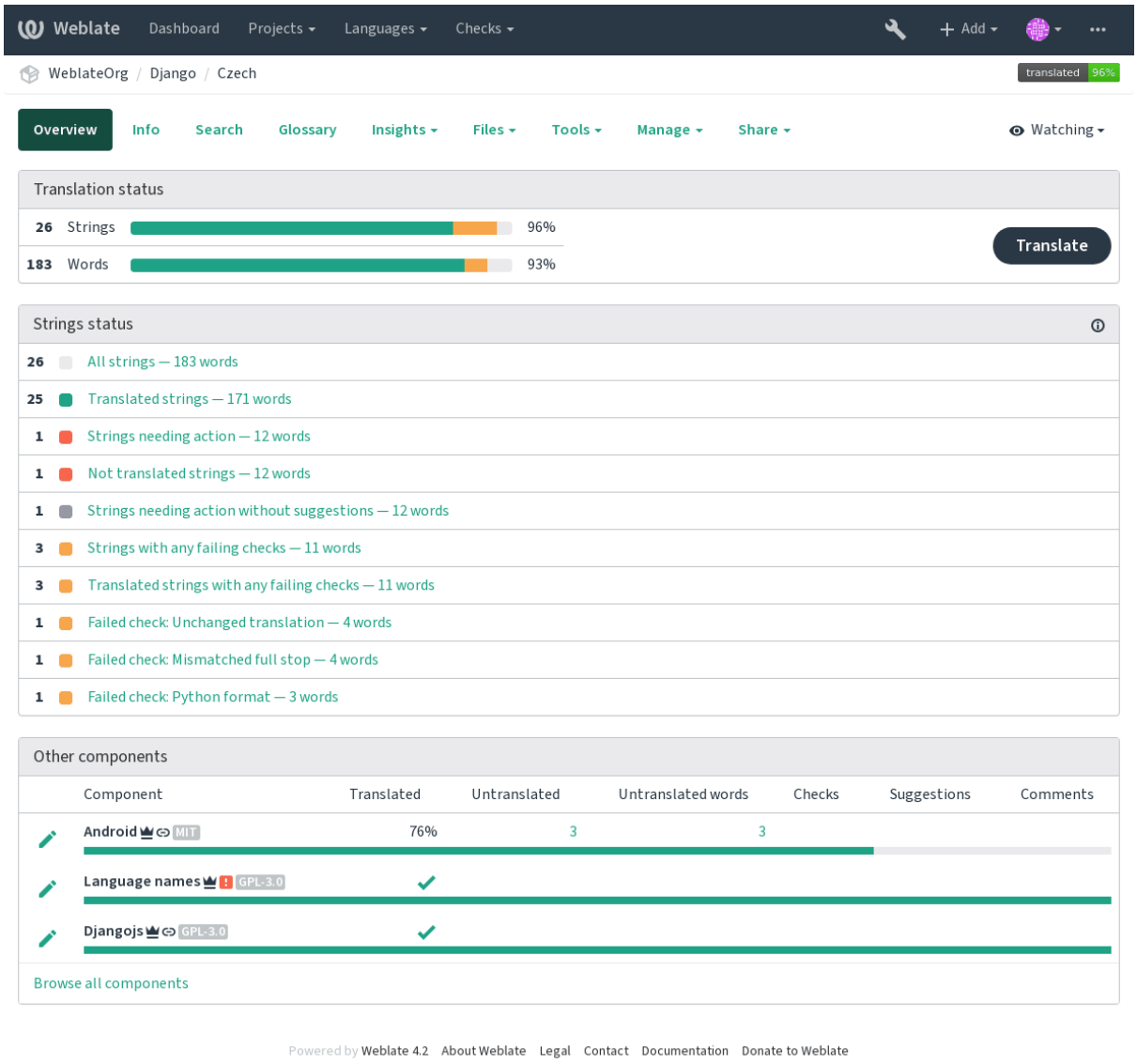
1.3.1 Переводческие проекты

В проектах переводов содержатся относящиеся к ним компоненты, связанные с одним и тем же программным обеспечением, книгой или проектов.



1.3.2 Ссылки на перевод

В компоненте находится набор ссылок, которые ведут к реальному переводу. Далее перевод подразделяется на отдельные проверки, такие как *Непереведённые строки* или *Строки, требующие действий*. Если весь проект переведён без ошибок, у него всё равно доступна ссылка *Все строки*. В качестве альтернативы вы можете воспользоваться полем поиска для поиска определённой строки или термина.



1.3.3 Предложения

Примечание: Фактические разрешения могут отличаться в зависимости от конфигурации Weblate.

Анонимные пользователи могут только (если это разрешено) вносить предложения. При этом, в случае возникновения в переводе неопределённости, он всё равно будет доступен для зарегистрированных пользователей, что подтолкнёт другого переводчика его отрецензировать.

Предложения ежедневно сканируются с целью удаления дубликатов или предложений, совпадающих с текущим переводом.

1.3.4 Комментарии

Комментарии могут быть размещены ко двум объектам — к исходной строке или к переводу. Выберите тот, который соответствует теме, которую вы хотите обсудить. Комментарии к исходной строке хороши для того, чтобы дать обратную связь по изначальной строке, например, что её нужно перефразировать или что она вводит в заблуждение.

В комментариях вы можете использовать синтаксис Markdown и упоминать других пользователей, используя синтаксис `@упоминаний`.

1.3.5 Варианты

Варианты используются для группировки вариантов строки различной длины. Пользовательский интерфейс может использовать различные строки в зависимости от размеров экрана или окна.

См.также:

Варианты строк

1.3.6 Метки

Метки используются для категоризации строк в проекте. Их можно использовать для дальнейшей настройки рабочего процесса локализации, например, для определения категорий строк.

См.также:

Метки строк

1.3.7 Перевод

На странице перевода отображаются исходная строка и область редактирования для перевода. Если переводится строка со множественным числом, то показываются несколько исходных строк и областей редактирования, каждая из которых описывает свою форму множественного числа.

Все специальные пробельные символы подчёркиваются красным цветом и обозначаются серыми видимыми символами-заменителями. Более одного последовательного пробела также подчёркиваются красным цветом, чтобы предупредить переводчика о потенциальной проблеме форматирования.

На этой странице также могут быть показаны различные кусочки дополнительной информации, большая часть которой берётся из исходного кода проекта (например, контекст, комментарии разработчика или место, где используется сообщение). Если вы в своих предпочтениях выбрали дополнительные языки, над исходной строкой будет показан перевод на эти языки (смотрите раздел *Вспомогательные языки*).

Под переводом будут показаны все предложения, внесённые другими пользователями, которые вы, в свою очередь, можете принять, принять с изменениями или удалить.

Формы множественного числа

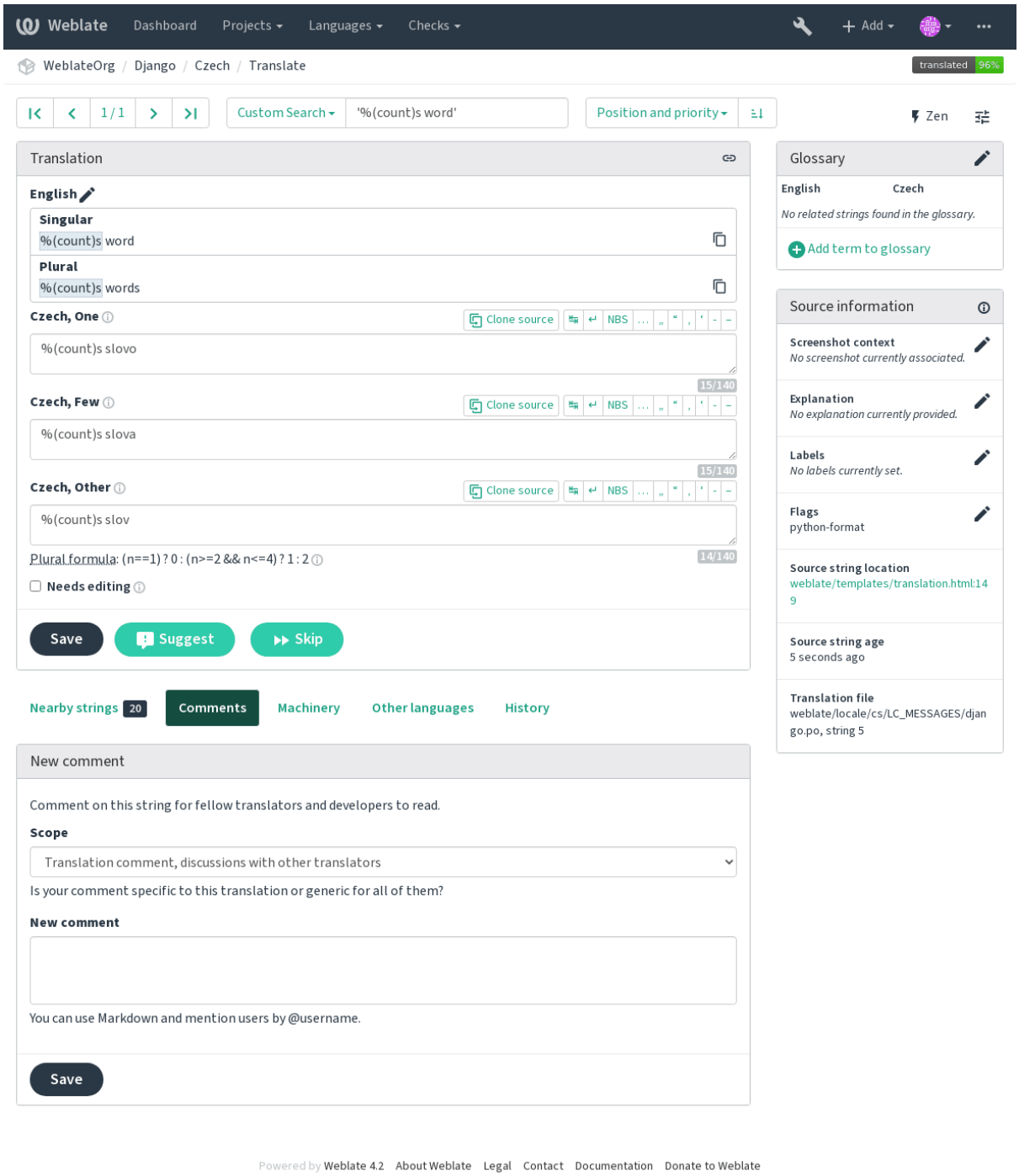
Слова, изменяющие свою форму в связи с их числовым обозначением, называются формами множественного числа. Каждый язык имеет своё определение множественного числа. В английском языке, например, существует одна форма множественного числа. В единичном определении, к примеру «автомобиля» — «car» — неявно подразумевается одна машина, а во множественном определении «автомобили» — «cars» — подразумеваются уже две или более машины, либо же понятие автомобилей как существительное. Такие языки, как, например, чешский, арабский или русский, имеют больше форм множественного числа, а их правила для этих форм отличаются от правил для английского языка.

Weblate полностью поддерживает каждую из этих форм, на каждом соответствующем языке, путём отдельного перевода каждой формы множественного числа. Количество полей и способ их использования в переводе

зависит от настроенной формулы множественного числа. Weblate показывает только основную информацию, но вы можете найти более подробное описание в языковых правилах для форм множественного числа от Консорциума Юникода.

См.также:

Формула множественного числа



Горячие клавиши

Изменено в версии 2.18: Сочетания клавиш в версии 2.18 были обновлены, чтобы снизить вероятность пересечения с умолчательными настройками браузера или системы.

Во время перевода можно использовать следующие сочетания клавиш:

Alt+Home Переход к первому переводу в текущем поиске.

Alt+End Переход к последнему переводу в текущем поиске.

Alt+PageUp Переход к предыдущему переводу в текущем поиске.

Alt+PageDown Переход к последующему переводу в текущем поиске.

Alt+Enter, Ctrl+Enter, или Cmd+Enter Сохранение текущего перевода.

Ctrl+Shift+Enter или Cmd+Shift+Enter Снятие с перевода пометки «на правку» и его сохранение.

Ctrl+E или Cmd+E Фокусировка на редакторе перевода.

Ctrl+U или Cmd+U Фокусировка на редакторе комментариев.

Ctrl+M или Cmd+M Показ вкладки автоматизированного перевода.

Ctrl+<ЧИСЛО> или Cmd+<ЧИСЛО> Копирование из исходной строки кусочка под соответствующим номером.

Ctrl+M <ЧИСЛО> или Cmd+M <ЧИСЛО> Копирование в текущий перевод машинного перевода с заданным номером.

Ctrl+I <ЧИСЛО> или Cmd+I <ЧИСЛО> Игнорирование одного элемента в списке проваленных проверок.

Ctrl+J или Cmd+J Shows the *Nearby strings* tab.

Ctrl+S или Cmd+S Показ вкладки поиска.

Ctrl+O или Cmd+O Копирование исходной строки.

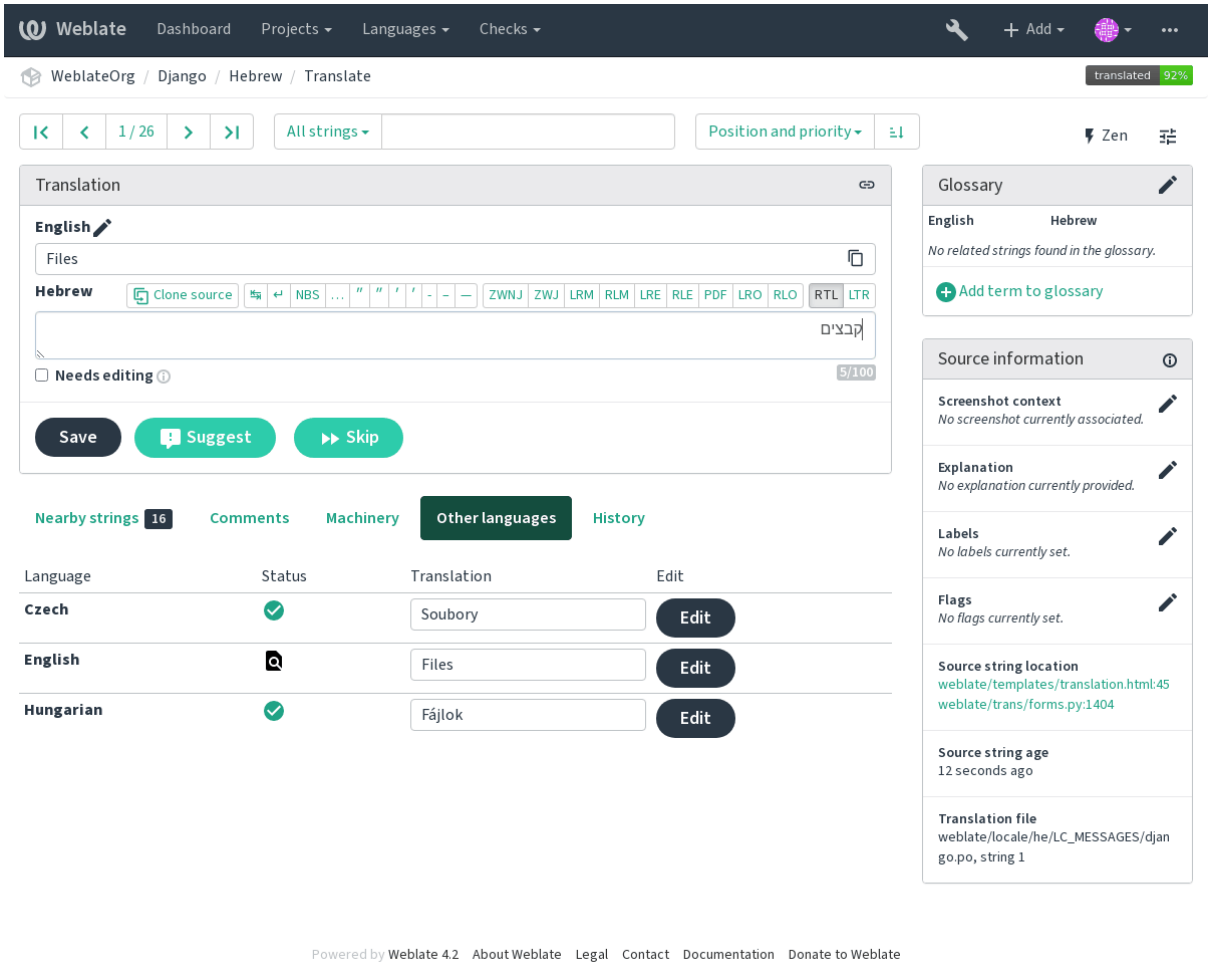
Ctrl+Y или Cmd+Y Toggles the *Needs editing* flag.

Визуальная клавиатура

Чуть выше поля перевода показывается небольшая визуальная клавиатура. Она может быть полезна для ввода символов, которые обычно не встречаются в тексте или которые трудно набирать.

Показываемые символы разделены на три категории:

- Пользовательские символы, определённые в его *профиле*
- Предоставляемые Weblate символы для конкретного языка (например, кавычки или символы, специфичные для текста справа-налево)
- Символы, настроенные через параметр `SPECIAL_CHARS`



Контекст перевода

Это контекстное описание содержит информацию, относящуюся к текущей строке.

Атрибуты строки Такие вещи, как идентификатор сообщения, его контекст (`msgctxt`) или местоположение в исходном коде.

Снимки экрана В Weblate могут быть загружены снимки экрана, чтобы лучше информировать переводчиков о том, где и как используется строка, смотрите раздел *Visual context for strings*.

Соседние строки Отображает соседние сообщения из файла перевода. Обычно они также используются в аналогичном контексте и оказываются полезными для поддержания согласованности перевода.

Другие вхождения В случае, если сообщение появляется в нескольких местах (например, в нескольких компонентах), эта вкладка показывает их все, если в них найдены противоречия (смотрите раздел *Противоречия*). Вы можете выбрать, какую из строк использовать.

Память переводов Показывает похожие строки, переведённые в прошлом, смотрите раздел Memory Management.

Глоссарий Отображает термины из глоссария проекта, используемые в текущем сообщении.

Недавние изменения Список людей, которые недавно изменили это сообщение, используя Weblate.

Проект Информация о проекте, например, инструкции для переводчиков или информация о репозитории его системы контроля версий.

Также, если формат перевода поддерживает такую возможность, вы можете перейти по предоставленным ссылкам к соответствующему исходному коду, содержащему исходную строку.

История переводов

Каждое изменение по умолчанию (если это не отключено в настройках компонента) сохраняется в базе данных и может быть отменено. Также при необходимости через нижележащую систему управления версиями можно отменить всё, что угодно.

Длина переведённой строки

Weblate может ограничивать длину перевода несколькими способами, чтобы гарантировать, что переведённая строка не будет слишком длинной:

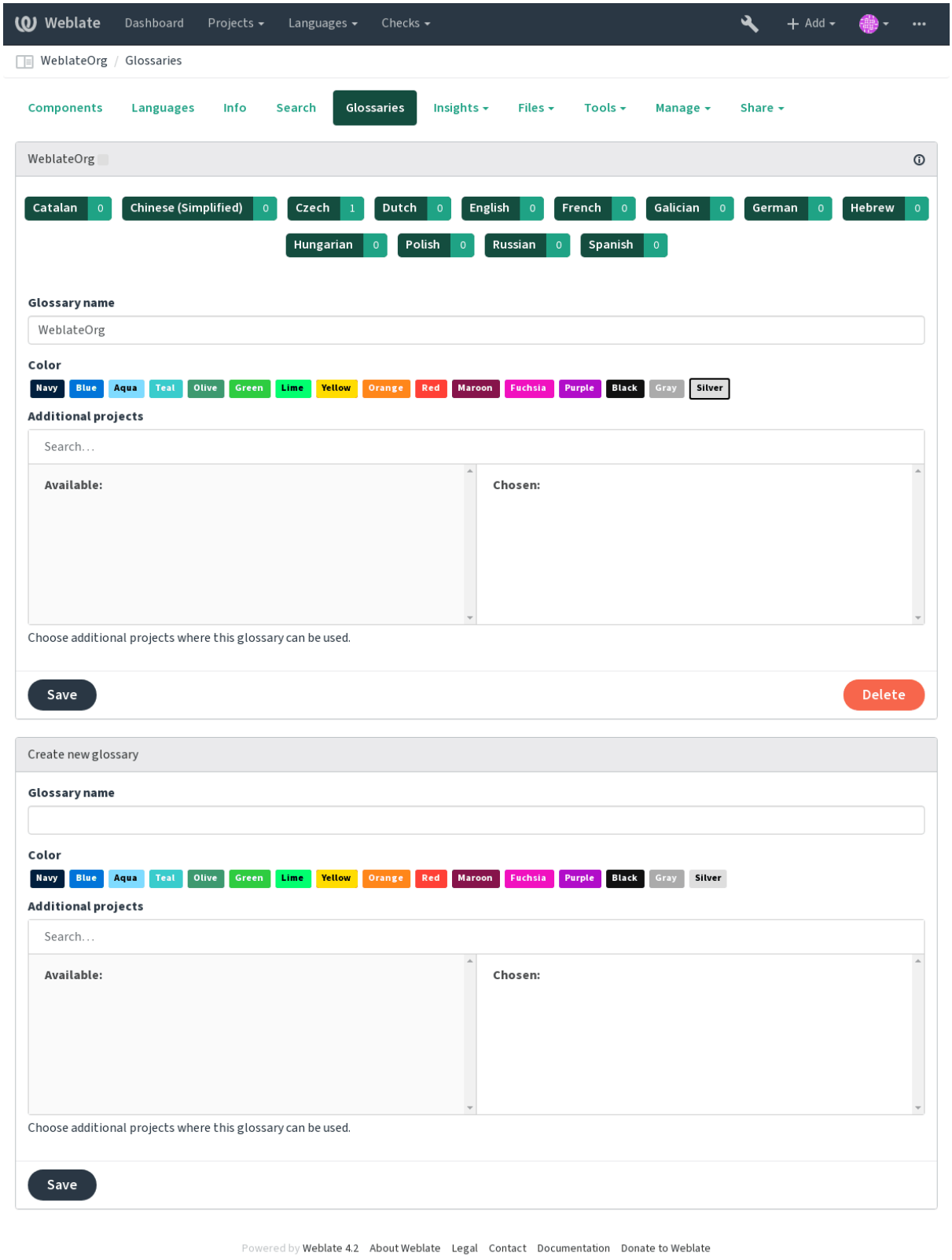
- Ограничение по умолчанию разрешает переводу быть в десять раз длиннее исходной строки. Это поведение можно включить с помощью параметра `LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH`. Если вы упёрлись в это ограничение, оно может быть вызвано тем, что одноязычный перевод был сконфигурирован как двуязычный, что заставило Weblate думать, что исходной строкой вместо настоящей исходной строки является ключ перевода. Дополнительную информацию смотрите в разделе *Двуязычные и одноязычные форматы*.
- Максимальная длина в символах, определяемая в файле перевода или флагом, смотрите раздел *Максимальная длина перевода*.
- Максимальный отрисовываемый размер текста в пикселях, определяемый флагами, смотрите раздел *Максимальный размер перевода*.

1.3.8 Глоссарий

К каждому проекту к любому языку может быть прикреплён глоссарий, хранящий сокращения для используемой терминологии. Таким образом облегчается поддержание согласованности перевода. Термины из переводимой в данный момент строки могут быть отображены в нижних вкладках.

Управление глоссариями

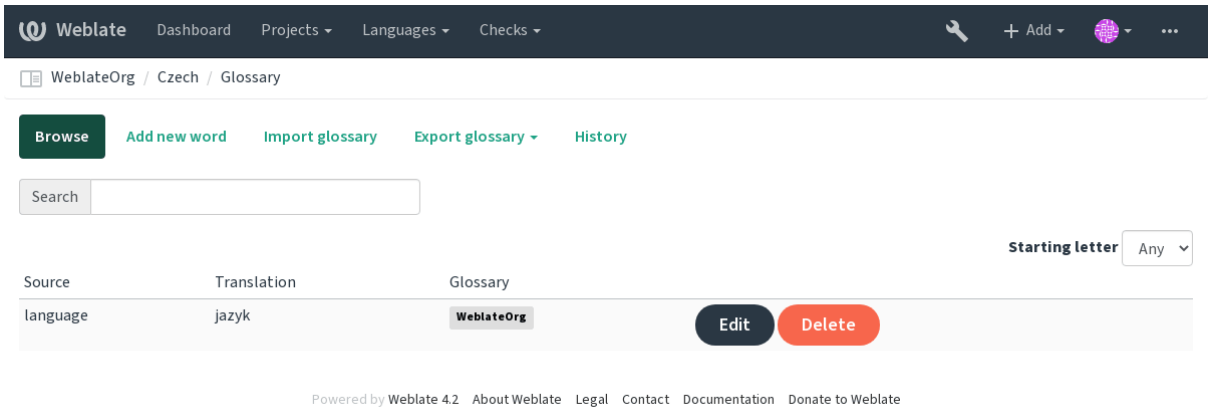
На вкладке *Глоссарии* любого проекта вы можете редактировать существующие глоссарии.



Пустой глоссарий для проекта создаётся автоматически при создании проекта. Глоссарии используются совместно всеми компонентами одного и того же проекта, кроме того, вы можете использовать их совместно с другими проектами. Это можно сделать только для проектов, которые вы можете администрировать.

В этом списке можно выбрать, каким глоссарием управлять (показываются все языки, используемые в текущем проекте). Проследовав по одной из языковых ссылок, вы попадёте на страницу, на которой можно редактировать, импортировать или экспортировать выбранный глоссарий, либо просматривать его историю

редактирования:



1.3.9 Машинный перевод

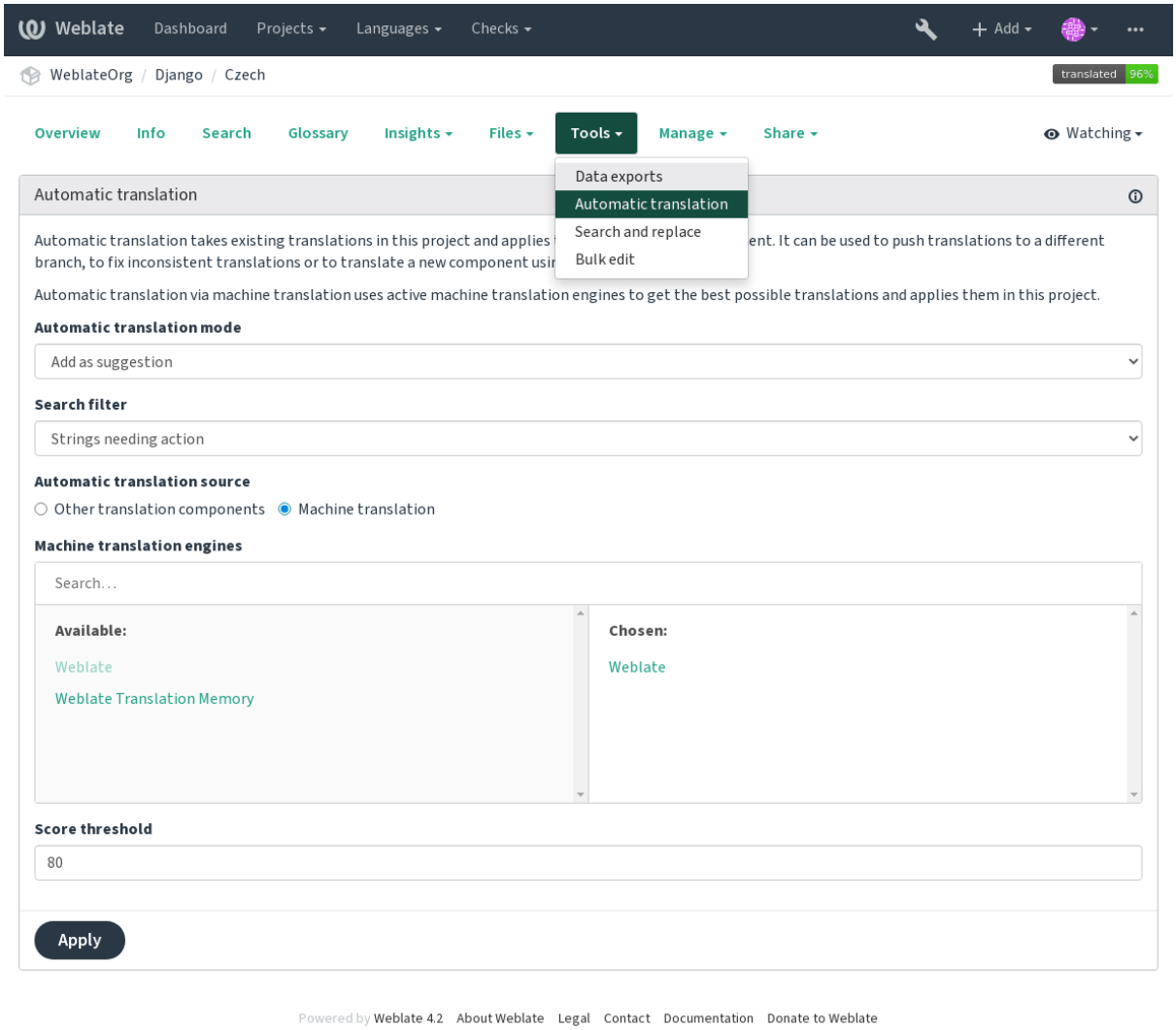
Основываясь на конфигурации и вашем языке перевода, Weblate предоставляет вам предложения от нескольких инструментов машинного перевода. Все машинные переводы доступны на единой вкладке каждой страницы перевода.

См.также:

Список поддерживаемых инструментов можно найти в разделе *Машинный перевод*.

1.3.10 Автоматический перевод

Вы можете использовать автоматический перевод для начального перевода на основе внешних источников. Этот инструмент называется *Автоматический перевод* и доступен в меню *Инструменты* после выбора компонента и языка:



Возможны два режима работы этого инструмента:

- Использование в качестве источника для переводов других компонентов Weblate.
- Использование выбранных сервисов машинного перевода с переводами выше определённого порога качества.

Вы также можете выбрать, какие строки должны быть переведены автоматически.

> **Предупреждение:** Имейте в виду, что если использовать широкие фильтры, вроде фильтра *Все строки*, это приведёт к перезаписи существующих переводов.

Инструмент полезен в ряде ситуаций, например, при объединении переводов различных компонентов (к примеру, веб-сайта и приложения) или при начальном переводе нового компонента с использованием существующих переводов (памяти переводов).

См.также:

Keeping translations same across components

1.3.11 Ограничение частоты запросов

Чтобы избежать злоупотребления интерфейсом, существует ограничение на частоту запросов, применяемое к некоторым операциям, таким как поиск, отправка контактной формы или перевод. В случае, если вы натолкнулись на это ограничение, вы будете на определённый период заблокированы, по истечению которого вы сможете заново выполнить операцию.

Ограничения по умолчанию описаны в руководстве по администрированию в разделе *Ограничение частоты запросов*, но они могут быть изменены в конфигурации.

1.3.12 Массовая правка

Массовая правка позволяет выполнять операцию одновременно с несколькими строками. Вы определяете критерии поиска строк и операцию для выполнения, и все найденные строки обновляются. Поддерживаются следующие операции:

- Изменение состояния строки (например, для одобрения всех строк, находящихся на рецензировании)
- Корректировка флагов перевода (смотрите раздел *Настройка поведения*)
- Корректировка меток строк (смотрите раздел *Метки строк*)

Подсказка: Этот инструмент называется *Массовая правка* и доступен он из меню *Инструменты* проекта, компонента или перевода.

См.также:

Надстройка «Массовая правка»

1.4 Скачивание и загрузка переводов

Вы можете экспортировать файлы из перевода, внести в них изменения и снова их импортировать. Это позволяет поработать над переводом в автономном режиме, а затем влить изменения обратно в существующий перевод. Это работает, даже если существующий перевод за это время был изменён.

Примечание: Доступные варианты могут быть ограничены *списками доступа*.

1.4.1 Скачивание переводов

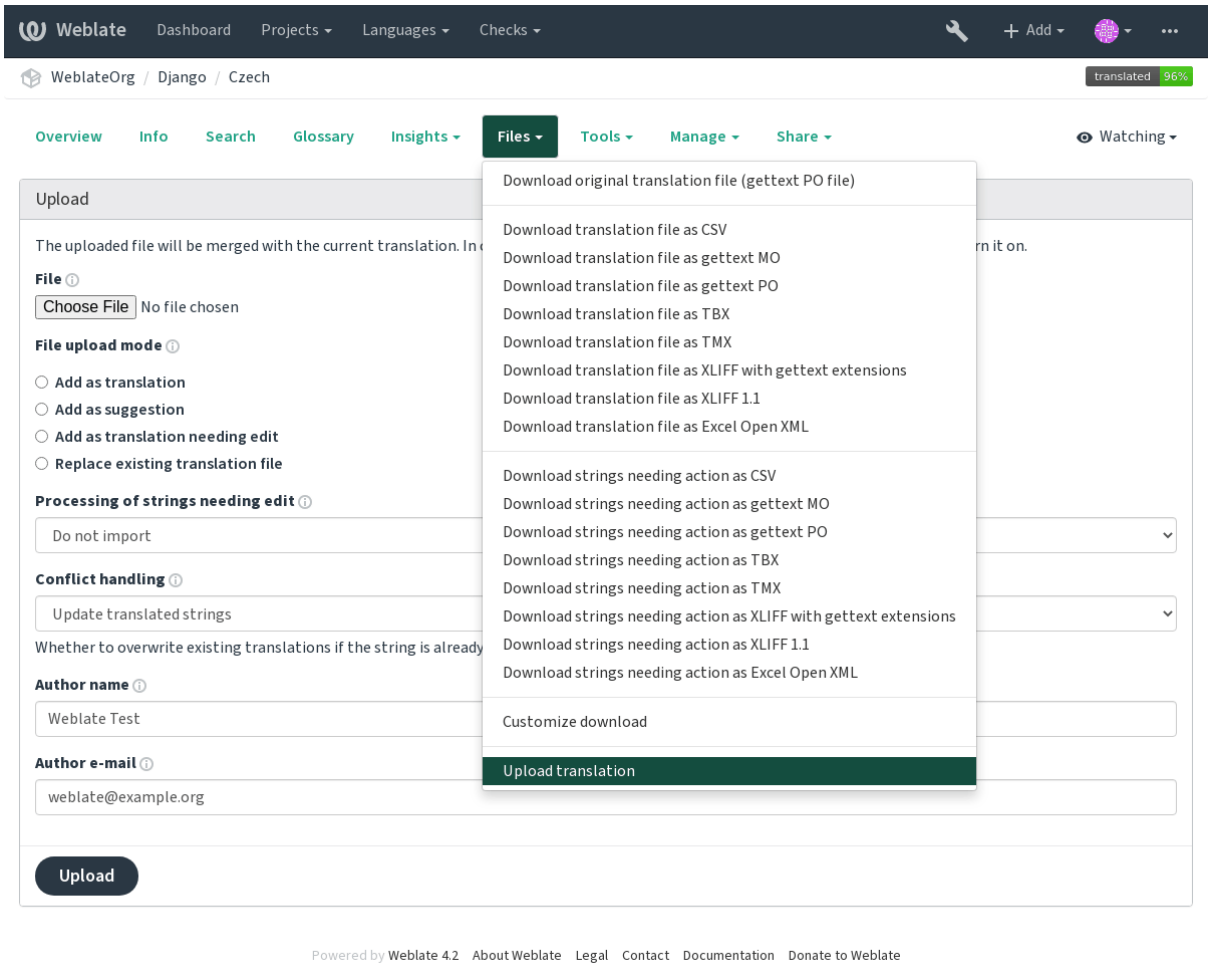
Из панели управления проекта или компонента переводимые файлы можно скачать через пункт *Скачать оригинальный файл перевода* меню *Файлы*, который создаёт копию оригинального файла в том виде, в котором он хранится в вышестоящей системе контроля версий.

Также вы можете скачать перевод, преобразованный в один из широко распространённых форматов локализации. Преобразованные файлы будут обогащены данными, предоставленными Weblate, такими как дополнительный контекст, комментарии или флаги.

В меню *Файлы* доступно несколько форматов файлов, в том числе скомпилированные файлы для использования в выбранном вами приложении (например, файлы `.mo` для GNU Gettext).

1.4.2 Загрузка переводов

После того, как вы внесли в перевод свои изменения, воспользуйтесь пунктом *Загрузить перевод* меню *Файлы*.



Поддерживаемые форматы файлов

Можно загрузить любой файл в поддерживаемом формате, но всё же рекомендуется использовать тот же самый формат файла, который используется для перевода, в противном случае некоторые возможности могут быть перенесены неправильно.

См.также:

Поддерживаемые форматы файлов

Загруженный файл объединяется с существующим для обновления перевода, по умолчанию перезаписывая существующие записи (это можно отключить или включить в диалоге загрузки).

Способы импортирования

При загрузке файлов перевода вы можете выбирать из следующих вариантов:

Добавить как перевод (`translate`) Импортированные переводы добавляются как переводы. Это наиболее распространённое использование и поведение по умолчанию.

Добавить как предложение (`suggest`) Импортированные переводы добавляются как предложения, делайте это тогда, когда хотите, чтобы загруженные строки были отрецензированы.

Добавить перевод как «На правку» (`fuzzy`) Импортированные переводы добавляются как переводы, требующие правки. Это может быть полезно, когда вы хотите, чтобы переводы не только использовались, но также были и отрецензированы.

Заменить существующий файл перевода (`replace`) Существующий файл заменяется новым содержимым. Это может привести к потере существующих переводов, используйте этот вариант с осторожностью.

Обновить исходные строки (`source`) Обновляет исходные строки в двуязычном файле перевода. Это поведение похоже на то, что делает надстройка *Обновлять PO-файлы до соответствия POT (msgmerge)*.

См.также:

`POST /api/translations/(string:project)/(string:component)/(string:language)/file/`

Разрешение конфликтов

Определяет, что делать с загружаемыми строками, которые уже переведены.

Строки, отмеченные как «На правку»

Также существует параметр, контролирующий то, что делать со строками в импортируемом файле, нуждающимися в редактировании . Такие строки можно обработать одним из трёх следующих способов: «Не импортировать», «Импортировать строки, как отмеченные „На правку“» или «Импортировать как переведённое».

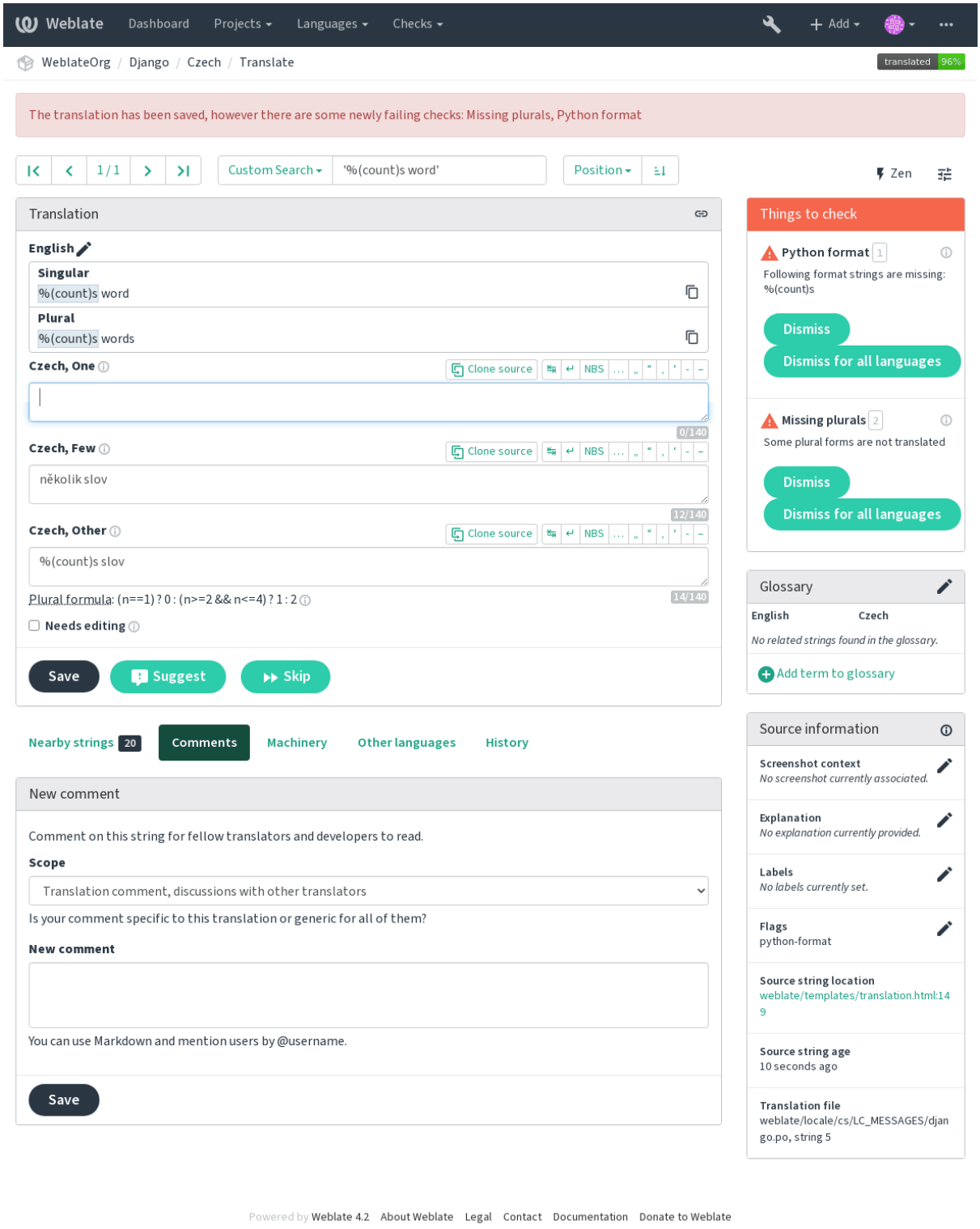
Переопределение авторства

С разрешениями администратора вы также можете указать авторство загружаемого файла. Это может быть полезно в случае, если вы получили файл другим способом и хотите объединить переводы из него с существующими с правильным указанием фактического автора.

1.5 Проверки и исправления

Проверки качества помогают выявить распространённые ошибки переводчиков, что гарантирует хорошее состояние перевода. В случае ложно-положительных срабатываний проверки могут быть проигнорированы.

После отправки перевода с неудачной проверкой, она сразу же показывается пользователю:


Weblate
Dashboard
Projects
Languages
Checks
Add
WeblateOrg / Django / Czech / Translate
translated 96%
The translation has been saved, however there are some newly failing checks: Missing plurals, Python format
1 / 1
Custom Search
'%(count)s word'
Position
Zen
Translation
English
Singular
%(count)s word
Plural
%(count)s words
Czech, One
Clone source
NBS
0/140
Czech, Few
Clone source
NBS
několik slov
12/140
Czech, Other
Clone source
NBS
%(count)s slov
Plural formula: (n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2
14/140
Needs editing
Save
Suggest
Skip
Nearby strings 20
Comments
Machinery
Other languages
History
New comment
Comment on this string for fellow translators and developers to read.
Scope
Translation comment, discussions with other translators
Is your comment specific to this translation or generic for all of them?
New comment
You can use Markdown and mention users by @username.
Save
Things to check
Python format 1
Following format strings are missing: %(count)s
Dismiss
Dismiss for all languages
Missing plurals 2
Some plural forms are not translated
Dismiss
Dismiss for all languages
Glossary
English
Czech
No related strings found in the glossary.
Add term to glossary
Source information
Screenshot context
No screenshot currently associated.
Explanation
No explanation currently provided.
Labels
No labels currently set.
Flags
python-format
Source string location
weblate/templates/translation.html:149
Source string age
10 seconds ago
Translation file
weblate/locale/cs/LC_MESSAGES/django.po, string 5
Powered by Weblate 4.2
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.5.1 Автоматические исправления

В дополнение к *проверкам качества*, Weblate может автоматически исправлять некоторые распространённые ошибки в переведённых строках. Используйте эту возможность с осторожностью, чтобы не допустить добавления новых ошибок.

См.также:

`AUTOFIX_LIST`

1.5.2 Проверки качества

Weblate использует широкий спектр проверок качества строк. В следующем разделе они все описываются более подробно. Также существуют проверки, специфичные для конкретных языков. Если какая-то проверка срабатывает ошибочно, пожалуйста, отправьте сообщение об ошибке.

См.также:

`CHECK_LIST`, *Настройка поведения*

1.5.3 Проверки перевода

Выполняется при каждом изменении перевода, помогая переводчикам поддерживать высокое качество перевода.

BBcode-разметка

BBcode в переводе не соответствует источнику

BBCode представляет собой простую разметку, полезную для, например, выделения важных частей сообщения жирным шрифтом или курсивом.

Эта проверка гарантирует, что они также будут присутствовать в переводе.

Примечание: На данный момент метод обнаружения тегов BBcode достаточно прост, поэтому данная проверка может привести к ложным срабатываниям.

Повторяющиеся слова

Одно и то же слово повторяется в тексте дважды подряд:

Добавлено в версии 4.1.

Проверяет, что в переводе не встречаются последовательные дубликаты слова. Обычно это указывает на ошибку в переводе.

Подсказка: Эта проверка включает в себя правила, специфичные для конкретных языков, чтобы избежать ложных срабатываний. В случае ложного срабатывания этой проверки в вашем случае, сообщите нам об этом. Смотрите раздел *Reporting issues in Weblate*.

Двойной пробел

Перевод содержит двойной пробел

Проверяет наличие в переводе двойного пробела, чтобы избежать ложных срабатываний на других проверках, связанных с пробелами.

Проверка возвращает ложь, если в исходной строке найден двойной пробел, что означает, что двойной пробел был внесён в неё преднамеренно.

Форматированные строки

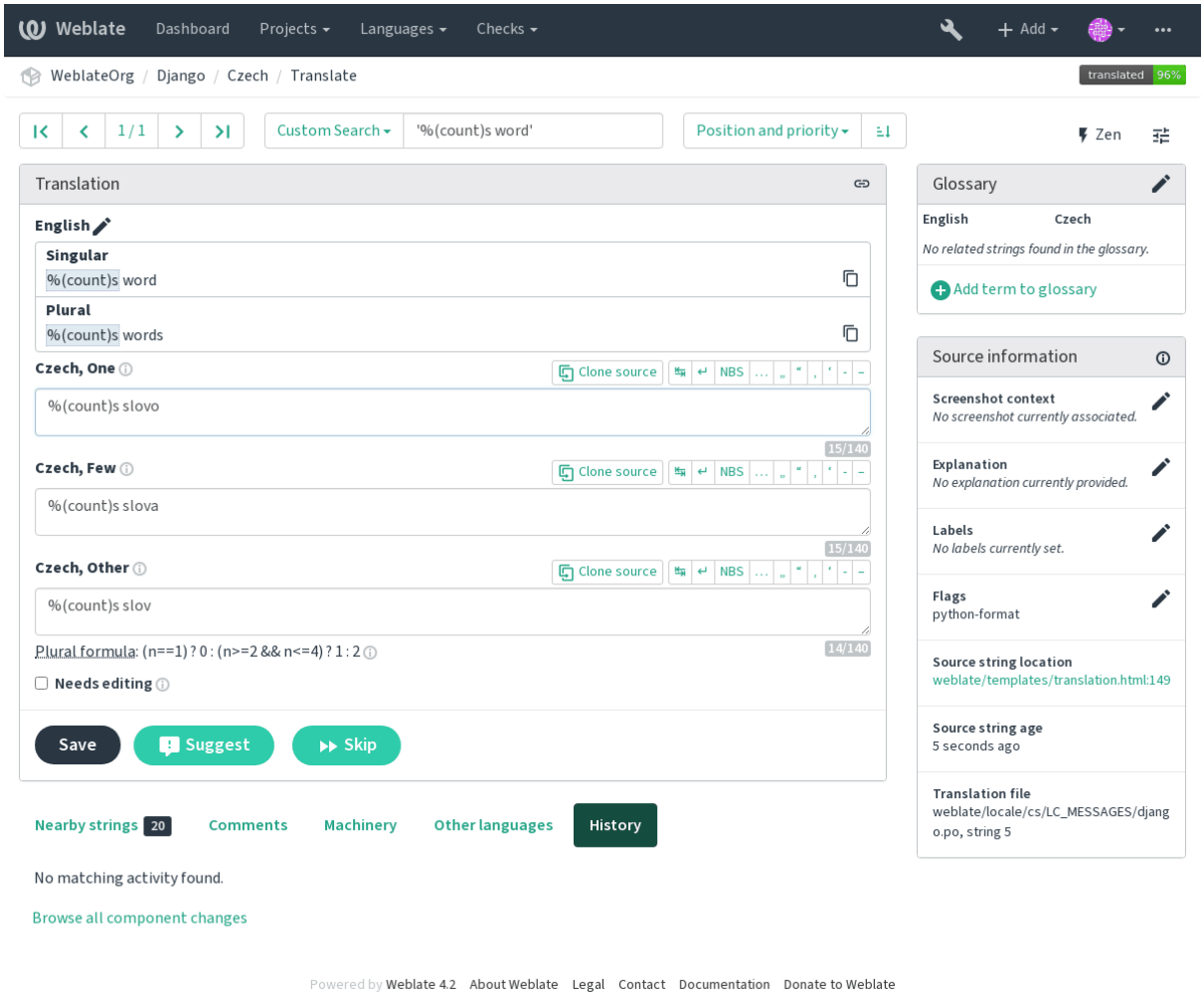
Проверяет, что форматирование в исходной строке и в переводе одинаково. Упущение в переводе строк формата обычно вызывает серьёзные проблемы, поэтому форматирование в переводе обычно должно соответствовать форматированию исходной строки.

Weblate поддерживает проверку строк формата на нескольких языках. Проверка не включается автоматически, только если строка соответствующим образом помечена (например, флагом *c-format* для C-формата). Gettext добавляет эту разметку автоматически, но для других форматов файлов или если ваши PO-файлы не сгенерированы программой **`xgettext`** вам, вероятно, придётся добавить её вручную.

Её можно добавлять как индивидуально в каждую единицу перевода (смотрите раздел *Additional info on source strings*), так и в *конфигурации компонента*. Определение формата для компонента целиком проще, но может привести к ложным срабатываниям, если строка не интерпретируется как строка формата, но при этом внутри неё используется синтаксис строки формата.

Подсказка: В случае, если конкретная проверка формата в Weblate отсутствует, вы можете использовать общую проверку *Символы-заполнители*.

Помимо проверки, эти флаги также включают подсветку строк формата, чтобы их было легко вставлять в перевод:



Интерполяционная строка AngularJS

Интерполяционные строки AngularJS не сооответствуют источнику

Именованная строка формата	`Your balance is {{amount}} {{ currency }}`
Флаг для включения	*angularjs-format*

См.также:

AngularJS: API: $interpolate

C-формат

Строка в формате C не соответствует источнику

Простая строка формата	`There are %d apples`
Позиционная строка формата	`Your balance is %1$d %2$s`
Флаг для включения	*c-format*

См.также:

Строки формата C, Синтаксис строки форматирования функции printf C

C#-формат

Строка в формате C# не соответствует источнику

Позиционная строка формата	`There are {0} apples`
Флаг для включения	*c-sharp-format*

См.также:

Строки формата C#

Шаблонные литералы ECMAScript

Шаблонные литералы ECMAScript не соответствует источнику

Интерполяция	`There are ${number} apples`
Флаг для включения	*es-format*

См.также:

Шаблонные литералы

Интерполяция i18next

Интерполяция i18next не соответствует источнику

Добавлено в версии 4.0.

Интерполяция	`There are {{number}} apples`
Вложение	`There are $t(number) apples`
Флаг для включения	*i18next-interpolation*

См.также:

Интерполяция i18next

Java-формат

Строка в формате Java не соответствует источнику

Простая строка формата	`There are %d apples`
Позиционная строка формата	`Your balance is %1$d %2$s`
Флаг для включения	*java-format*

См.также:

Строки в формате Java

Java MessageFormat

Строка в формате Java MessageFormat не соответствует источнику

Позиционная строка формата	`There are {0} apples`
Флаг для включения	Флаг *java-messageformat* включает эту проверку безусловно
	Флаг *auto-java-messageformat* включает проверку только если в исходной строке содержится строка формата

См.также:

Java MessageFormat

JavaScript-формат

Строка в формате JavaScript не соответствует источнику

Простая строка формата	`There are %d apples`
Флаг для включения	*javascript-format*

См.также:

Строки форматирования JavaScript

Символы-заполнители в процентной нотации

Символы-заполнители в процентной нотации не соответствуют источнику

Добавлено в версии 4.0.

Простая строка формата	`There are %number% apples`
Флаг для включения	*percent-placeholders*

Perl-формат

Строка в формате Perl'a не соответствует источнику

Простая строка формата	`There are %d apples`
Позиционная строка формата	`Your balance is %1$d %2$s`
Флаг для включения	*perl-format*

См.также:

Функция sprintf Perl'a, Строки формата Perl'a

PHP- формат

Строка в формате PHP не соответствует источнику

Простая строка формата	`There are %d apples`
Позиционная строка формата	`Your balance is %1$d %2$s`
Флаг для включения	*php-format*

См.также:

Документация по функции sprintf PHP, Строки формата PHP

Python-формат: фиг. скобки

Форматирование фигурных скобок в Python не соответствует источнику

Простая строка формата	`There are {} apples`
Именованная строка формата	`Your balance is {amount} {currency}`
Флаг для включения	*python-brace-format*

См.также:

Формат фигурных скобок в Python'е, Python'ьи строки формата

Python-формат

Строка в формате Python'а не соответствует источнику

Простая строка формата	`There are %d apples`
Именованная строка формата	`Your balance is %(amount) %(currency)`
Флаг для включения	*python-format*

См.также:

Форматирование строк в Python'е, Python'ьи строки формата

Qt-формат

Строка в формате Qt не соответствует источнику

Позиционная строка формата	`There are %1 apples`
Флаг для включения	*qt-format*

См.также:

Qt QString::arg()

Формат множественного числа Qt

Строка в формате множественного числа Qt не соответствует источнику

Строка в формате множественного числа	`There are %Ln apple(s)`
Флаг для включения	*qt-plural-format*

См.также:

Руководство Qt по интернационализации

Ruby-формат

Строка формата Ruby не соответствует источнику

Простая строка формата	`There are %d apples`
Позиционная строка формата	`Your balance is %1$f %2$s`
Именованная строка формата	`Your balance is %+.2<amount>f %<currency>s`
Именованная шаблонная строка	`Your balance is %{amount} %{currency}`
Флаг для включения	*ruby-format*

См.также:

Ruby Kernel#sprintf

Уже переведено

Эта строка уже переводилась

Означает, что строка уже переведена. Это может произойти, если переводы откатывались в системе контроля версий или иным образом терялись.

Противоречия

Эта строка имеет более одного перевода в этом проекте, либо же в некоторых компонентах она не переведена.

Weblate проверяет переводы одинаковых строк по всем переводам в проекте, чтобы помочь вам поддерживать переводы согласованными.

Проверка срабатывает, когда в проекте обнаруживаются различные переводы одной и той же строки. Также это может привести к несогласованности в отображаемых проверках. Другие переводы этой строки можно найти на вкладке *Другие вхождения*.

Примечание: Также эта проверка срабатывает в том случае, если строка в одном компоненте переведена, а в другом — нет. Она может использоваться, чтобы по-быстрому вручную обработать строки, не переведённые в некоторых компонентах; для этого надо просто на вкладке *Другие вхождения* нажать на кнопку *Использовать этот перевод*, отображаемую напротив каждой строки.

Для автоматизации перевода новых добавленных строк, которые уже переведены в другом компоненте, вы можете использовать надстройку *Автоматический перевод*.

См.также:

Keeping translations same across components

Используется кашида

Декоративные буквы кашиды не должны использоваться

Добавлено в версии 3.5.

Декоративные буквы кашида не должны использоваться в переводе. Также они известны как татвиль.

См.также:

Кашида в Википедии

Ссылки Markdown

Ссылки Markdown-разметки не соответствуют источнику

Добавлено в версии 3.5.

Ссылки Markdown-разметки не соответствуют источнику.

См.также:

Ссылки Markdown

Сноски Markdown

Ссылки на сноски Markdown не соответствуют источнику

Добавлено в версии 3.5.

Ссылки на сноски Markdown не соответствуют источнику.

См.также:

Ссылки Markdown

Синтаксис Markdown

Синтаксис Markdown-разметки не соответствует источнику

Добавлено в версии 3.5.

Синтаксис Markdown-разметки не соответствует источнику

См.также:

Строчные элементы Markdown

Максимальная длина перевода

Перевод не должен превышать заданную длину

Проверяет, что переводы имеют приемлемую длину, чтобы укладываться в доступное пространство. Эта проверка проверяет только длину символов перевода.

В отличие от других проверок, флаг для этой проверки должен быть установлен в виде пары `ключ:значение`, например, `max-length:100`.

Подсказка: Эта проверка смотрит на количество символов, что может быть не самой лучшей метрикой при использовании для отображения текста пропорциональных шрифтов. Проверка *Максимальный размер перевода* проверяет фактическую отрисовку текста.

Также для расширения позиций-заполнителей перед проверкой строки может быть полезен флаг `replacements:`.

Максимальный размер перевода

Отрисованный текст перевода не должен превышать заданный размер

Добавлено в версии 3.7.

Отрисованный текст перевода не должен превышать заданный размер. Эта проверка отрисовывает текст с переносом по строкам и проверяет, вписывается ли он в заданные границы.

Для этой проверки требуется один или два параметра — максимальная ширина и максимальное количество строк. В случае, если количество строк не указано, считается, что текст должен уместиться в одну строку.

Также директивами `font-*` вы можете настроить используемый шрифт (смотрите раздел *Настройка поведения*), например, следующие флаги перевода говорят, что текст, отрисовываемый шрифтом ubuntu размера 22, должен вписаться в две строки и 500 пикселей:

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

Подсказка: Возможно, вы захотите установить директивы `font-*` в *настройках компонента*, чтобы для всех строк компонента был настроен один и тот же шрифт. В случае, если вам нужно будет настроить их для отдельной строки, вы сможете переопределить их значения для этой строки.

Также для расширения позиций-заполнителей перед проверкой строки может быть полезен флаг `replacements:`.

См.также:

Управление шрифтами, *Настройка поведения*, *Максимальная длина перевода*

Несовпадающие n

Количество n в переводе не соответствует источнику

Экранированные символы новой строки обычно важны при форматировании вывода программы. Проверка срабатывает, если количество символов `\\n` в переводе не совпадает с их количеством в исходной строке.

Несовпадающие двоеточия

Источник и перевод не заканчиваются оба двоеточием

Проверяет, что двоеточия в исходной строке и в переводе одинаковы. Также проверяется наличие двоеточий для тех языков, в которых они не используются (китайский или японский).

См.также:

Двоеточие в Википедии

Несовпадающие многоточия

Источник и перевод не заканчиваются оба многоточием

Проверяет, что конечные многоточия в исходной строке и в переводе одинаковы. Эта проверка проверяет только настоящие многоточия (символы …), но не три подряд идущих точки (`...`).

Как правило, на печати многоточие отрисовывается лучше, чем три подряд стоящих точки и лучше звучит при преобразовании текста в речь.

См.также:

Многоточие в Википедии

Несовпадающие восклицательные знаки

Источник и перевод не заканчиваются оба восклицательным знаком

Проверяет, что восклицательные знаки в исходной строке и в переводе одинаковы. Также проверяется наличие восклицательных знаков для тех языков, в которых они не используются (китайский, японский, корейский, армянский, лимбу, бирманский или нко).

См.также:

Восклицательный знак в Википедии

Несовпадающие точки

Источник и перевод не заканчиваются оба точкой

Проверяет, что точки в исходной строке и в переводе одинаковы. Также проверяется наличие точек для тех языков, в которых они не используются (китайский, японский, деванагари или урду).

См.также:

Точка в Википедии

Несовпадающие вопросительные знаки

Источник и перевод не заканчиваются оба вопросительным знаком

Проверяет, что вопросительные знаки в исходной строке и в переводе одинаковы. Также проверяется наличие вопросительных знаков для тех языков, в которых они не используются (армянский, арабский, китайский, корейский, японский, эфиопский, ваи или коптский).

См.также:

Вопросительный знак в Википедии

Несовпадающие точки с запятой

Источник и перевод не заканчиваются оба точкой с запятой

Проверяет, что точки с запятой в конце предложений в исходной строке и в переводе одинаковы. Эта проверка может быть полезна, например, для сохранения форматирования в записях файлов `.desktop`.

См.также:

Точка с запятой в Википедии

Несовпадающие переводы строк

Количество символов новой строки в переводе не соответствует источнику

Символы новой строки обычно важны при форматировании вывода программы. Проверка срабатывает, если количество литералов``n`` в переводе не совпадает с их количеством в исходной строке.

Нет форм множественного числа

Некоторые формы множественного числа не переведены

Проверяет, что были переведены все формы множественного числа исходной строки. Сведения о том, как используется каждая форма множественного числа, можно найти в определении строки.

Отсутствие перевода в форме множественного числа в некоторых случаях приводит к тому, что при использовании формы множественного числа ничего не будет отображаться.

Символы-заполнители

В переводе пропущены некоторые символы-заполнители:

Добавлено в версии 3.9.

В переводе пропущены некоторые символы-заполнители. Эти символы либо извлекаются из файла перевода, либо определяются вручную с помощью флага ``placeholders`; если их несколько, они разделяются двоеточием, а строки с пробелами могут быть заключены в кавычки:

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

См.также:

Настройка поведения

Пробелы и пунктуация

Отсутствует неразрывный пробел перед знаком двойной пунктуации

Добавлено в версии 3.9.

Проверяет наличие неразрывного пробела перед знаком двойной пунктуации (восклицательным знаком, вопросительным знаком, точкой с запятой или двоеточием). Это правило используется только в некоторых избранных языках, вроде французского или бретонского, где по типографским правилам обязателен пробел перед знаком двойной пунктуации.

См.также:

Пробелы во французском и английском языках в (англоязычной) Википедии

Регулярное выражение

Перевод не соответствует регулярному выражению:

Добавлено в версии 3.9.

Перевод не соответствует регулярному выражению. Это выражение либо извлекается из файла перевода, либо определяется вручную с помощью флага `regex`:

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

Одинаковые формы множественного числа

Некоторые формы множественного числа переведены одинаково

Проверка срабатывает, если некоторые формы множественного числа в переводе переведены одинаково. В большинстве языков они должны быть разными.

Пустая строка в начале

Источник и перевод не начинаются оба с новой строки

Обычно символы новой строки в исходных текстах появляются по уважительной причине, их пропуск или добавление могут привести к проблемам с форматированием при использовании переведённого текста.

См.также:

Пустая строка в конце

Пробелы в начале

Источник и перевод не начинаются оба одинаковым числом пробелов

Пробелы в начале строки обычно используются для набивания отступов в интерфейсе, поэтому их важно сохранить.

Пустая строка в конце

Источник и перевод не заканчиваются оба новой строкой

Обычно символы новой строки в исходных текстах появляются по уважительной причине, их пропуск или добавление могут привести к проблемам с форматированием при использовании переведённого текста.

См.также:

Пустая строка в начале

Пробелы в конце

Источник и перевод не заканчиваются оба пробелом

Проверяет, что количество конечных пробелов в исходной строке и в переводе одинаково.

Конечные пробелы обычно используется для добавления отступа до соседних элементов, поэтому их удаление может поломать макет.

Перевод без изменения

Источник и перевод тождественны

Возникает, если исходная строка и соответствующая строка перевода идентичны, по крайней мере в одной из форм множественного числа. Некоторые строки, обычно встречающиеся во всех языках, игнорируются, а различная разметка удаляется. Это уменьшает количество ложных срабатываний.

Эта проверка может помочь найти ошибочно не переведённые строки.

Поведение этой проверки по умолчанию заключается в исключении из проверки слов из встроенного чёрного списка. Это такие слова, которые зачастую не переводятся. Это полезно для того, чтобы избежать ложных срабатываний на коротких строках, которые состоят только из одного слова, одинакового в нескольких языках. Этот чёрный список можно отключить, добавив в строку или в компонент флаг `strict-same`.

См.также:

Конфигурация компонента, *Настройка поведения*

Небезопасный HTML-код

Перевод использует небезопасную HTML-разметку

Добавлено в версии 3.9.

Перевод использует небезопасную HTML-разметку. Эта проверка должна быть включена с помощью флага `safe-html` (смотрите раздел *Настройка поведения*). Также имеется соответствующий автоматический исправляльщик, который может автоматически почистить разметку.

См.также:

Проверка HTML осуществляется с помощью разработанной Mozilla библиотеки **`Bleach <https://bleach.readthedocs.io/`_**.

URL-ы

Перевод не содержит URL

Добавлено в версии 3.5.

Перевод не содержит URL-адрес. Эта проверка срабатывает только в том случае, если единица перевода помечена как содержащая URL-адрес. В этом случае перевод должен быть корректным URL-адресом.

XML-разметка

XML-теги в переводе не соответствуют источнику

Обычно это означает, что полученный результат будет выглядеть по-другому. В большинстве случаев такое изменение из-за перевода не является желаемым результатом, но иногда именно это и надо.

Проверяет, что XML-теги в исходной строке и в переводе одинаковы.

Синтаксис XML

Перевод не является корректным XML

Добавлено в версии 2.8.

Разметка XML некорректна.

Пробел нулевой ширины

Перевод содержит лишние символы пробела нулевой ширины

Символы пробела нулевой ширины (<U+200B>) используются для разбиения сообщений по словам (переноса по словам).

Поскольку обычно они вставляются по ошибке, эта проверка срабатывает на их присутствие в переводе. В некоторых программах при использовании этого символа могут возникнуть проблемы.

См.также:

Пробел нулевой ширины в (англоязычной) Википедии

1.5.4 Проверки для исходных строк

Проверки для исходных строк могут помочь разработчикам улучшить качество исходных строк.

Многоточие

В этой строке используются три точки (…) вместо символа многоточия (…)

Эта проверка срабатывает, когда в строке используются три подряд стоящих точки (`...`), когда вместо них должен использоваться символ многоточия (…).

Использование юникодного символа в большинстве случаев является лучшим подходом; он лучше выглядит при отрисовке текста и может лучше звучать при преобразовании текста в речь.

См.также:

Многоточие в Википедии

Давно не переводилась

Строка давно не переводилась

Добавлено в версии 4.1.

Если строка долгое время не переводилась, это может указывать на проблему в исходной строке, затрудняющую её перевод.

Множество неудачных проверок

Переводы на несколько языков имеют неудачные проверки

Во многих переводах этой строки имеются неудачные проверки качества. Обычно это свидетельствует о том, что нужно что-то сделать для улучшения исходной строки.

Часто неудачная проверка может быть вызвана пропущенной точкой в конце предложения или подобными мелкими проблемами, которые переводчики склонны исправлять в переводе, в то время как лучше было бы их исправить в исходной строке.

Несколько переменных без названий

Строка содержит несколько неименованных переменных, что не позволяет переводчикам их переупорядочивать

Добавлено в версии 4.1.

Строка содержит несколько неименованных переменных, что не позволяет переводчикам их переупорядочивать.

Рассмотрите возможность использования именованных переменных вместо неименованных, чтобы позволить переводчикам менять их порядок в переводимом тексте.

Формы единственного числа

Строка содержит множественное число, но не использует формы множественного числа

Строка содержит множественное число, но не использует формы множественного числа. В случае, если ваша система перевода его поддерживает, вы должны использовать вариант, учитывающий множественное число.

Например, при использовании Python'а и Gettext'а это может выглядеть так:

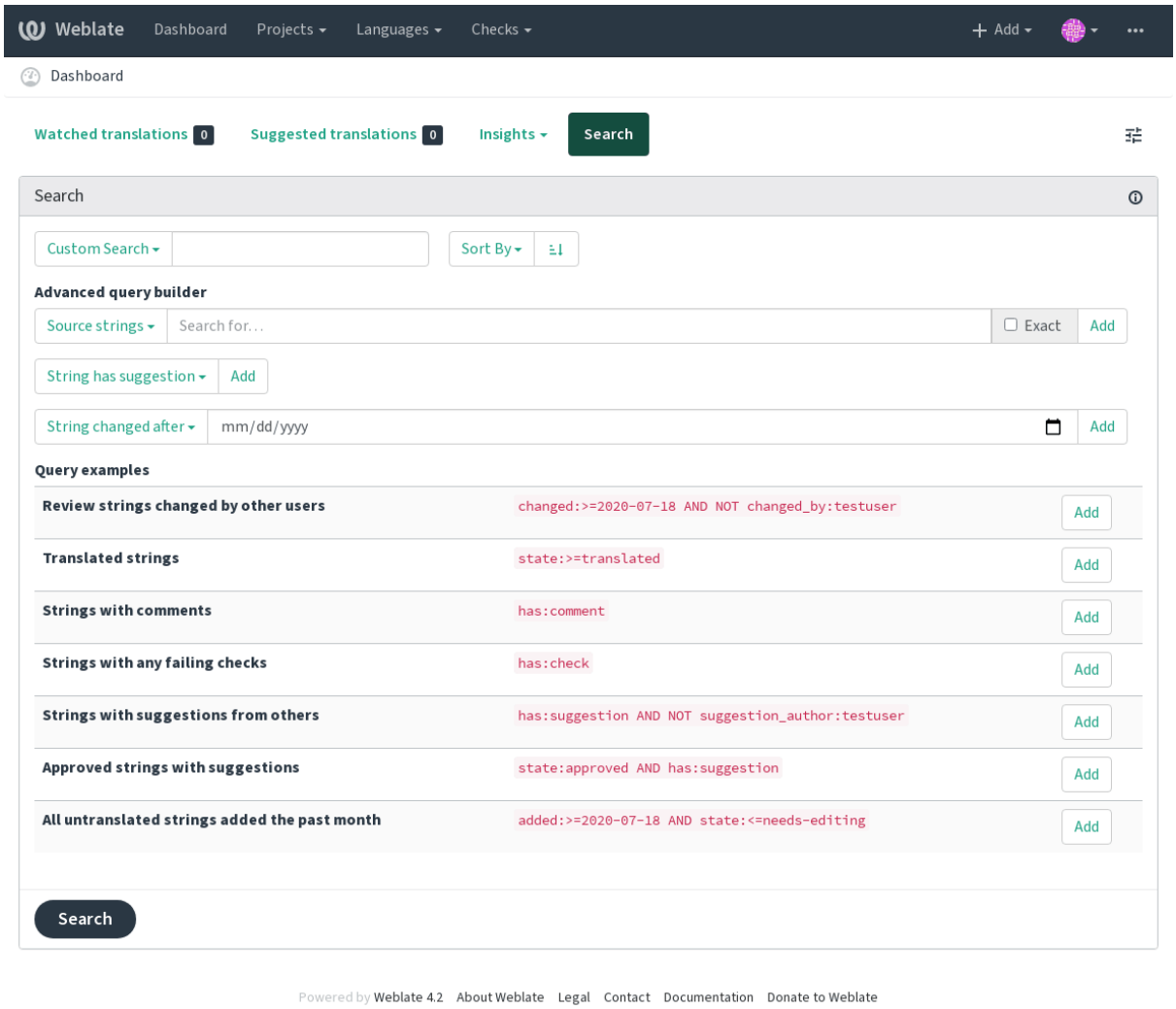
```
from gettext import ngettext

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

1.6 Поиск

Добавлено в версии 3.9.

Для поиска нужных строк можно использовать расширенные поисковые запросы, включающие в себя логические операции, круглые скобки или поиск по конкретным полям.

Если поле не определено, поиск проводится по полям *Источник*, *Перевод* и *Контекст*.



1.6.1 Простой поиск

Любая фраза, введённая в поле поиска, разбивается на слова. При поиске показываются строки, содержащие любое из этих слов. Чтобы найти точную фразу, поместите "фразу для поиска" в кавычки (работают как одинарные ('), так и двойные (") кавычки): `"это строка, заключённая в кавычки"` или `'другая закавыченная строка'`.

1.6.2 Поля

`source:ТЕКСТ` Регистронезависимый поиск по исходной строке.

`target:ТЕКСТ` Регистронезависимый поиск по целевой строке.

`context:ТЕКСТ` Регистронезависимый поиск по строке контекста.

`key:ТЕКСТ` Регистронезависимый поиск по строке ключа.

`note:ТЕКСТ` Регистронезависимый поиск по строке комментария.

`location:ТЕКСТ` Регистронезависимый поиск по строке местоположения.

`priority:ЧИСЛО` Приоритет строки.

`added:ДАТА_И_ВРЕМЯ` Метка времени, когда строка была добавлена в Weblate.

`state:ТЕКСТ` Поиск по состоянию (`approved`, `translated`, `needs-editing`, `empty`, `read-only`), поддерживает *Операторы поиска по полям*.

`pending:ЛОГИЧЕСКОЕ_ЗНАЧЕНИЕ` Строка, ожидающая сброса в систему контроля версий.

`has:ТЕКСТ` Поиск строк, имеющих атрибуты — `plural`, `context`, `suggestion`, `comment`, `check`, `dismissed-check`, `translation`, `variant`, `screenshot` (работает только для исходных строк).

`is:ТЕКСТ` Поиск по состояниям строки (`pending`, `translated`, `untranslated`).

`language:ТЕКСТ` Целевой язык строки.

`component:TEXT` Component slug, see *Плашка компонента*.

`project:TEXT` Project slug, see *Плашка проекта*.

`changed_by:ТЕКСТ` Строка была изменена автором с указанным именем пользователя.

`changed:ДАТА_И_ВРЕМЯ` Дата изменения строки, поддерживает *Операторы поиска по полям*.

`check:ТЕКСТ` Строка имеет неудачные проверки.

`dismissed_check:ТЕКСТ` Строка имеет скрытые проверки.

`comment:ТЕКСТ` Поиск в пользовательских комментариях.

`comment_author:ТЕКСТ` Фильтр по авторским комментариям.

`suggestion:ТЕКСТ` Поиск в предложениях.

`suggestion_author:ТЕКСТ` Фильтр по автору предложения.

1.6.3 Логические операторы

Вы можете комбинировать поисковые запросы, используя для формирования сложных запросов операторы `AND`, `OR` и `NOT` и круглые скобки. Например: `state:translated AND (source:hello OR source:bar)`

1.6.4 Операторы поиска по полям

Вы можете указать операторы, диапазоны или частичный поиск по дате или числам:

`state:>=translated` Состояние `translated` или лучше (`approved`).

`changed:2019` Изменено в 2019 году.

`changed:[2019-03-01 to 2019-04-01]` Изменено между двумя указанными датами.

1.6.5 Оператор точного поиска

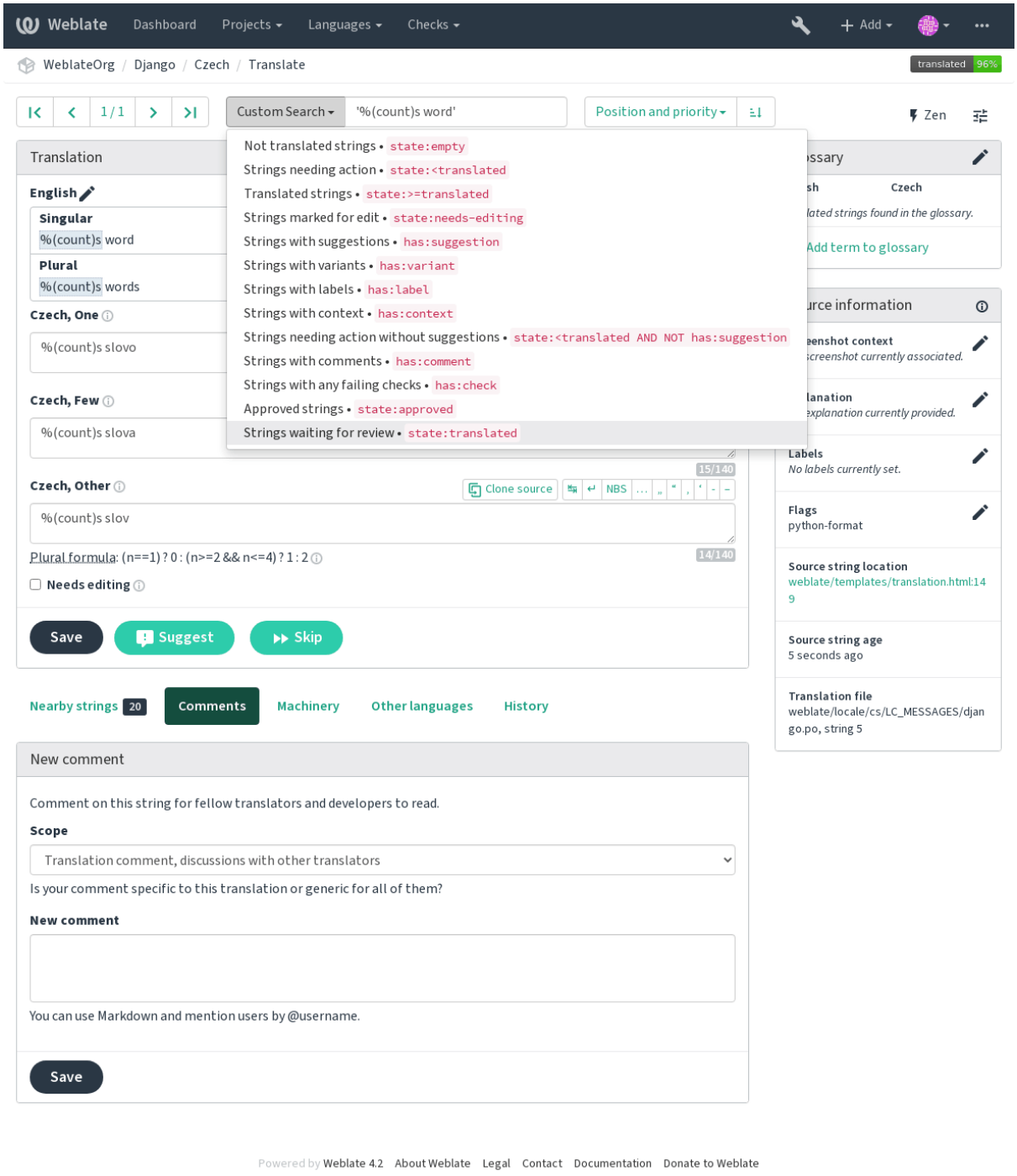
С помощью оператора = вы можете сделать запрос на точное совпадение различных строковых полей. Например, для нахождения всех исходных строк, точно равных строке `hello world`, используйте запрос `source:="hello world"`. При поиске односложных выражений кавычки можно опустить. Например, для нахождения всех исходных строк, точно равных строке `hello`, вы можете использовать запрос `source:=hello`.

1.6.6 Регулярные выражения

Везде, где принимается текст, вы также можете указать регулярное выражение в виде `r"выражение"`. Например, для поиска по исходным строкам, содержащим любую цифру от 2 до 5, используйте запрос `source:r"[2-5]"`

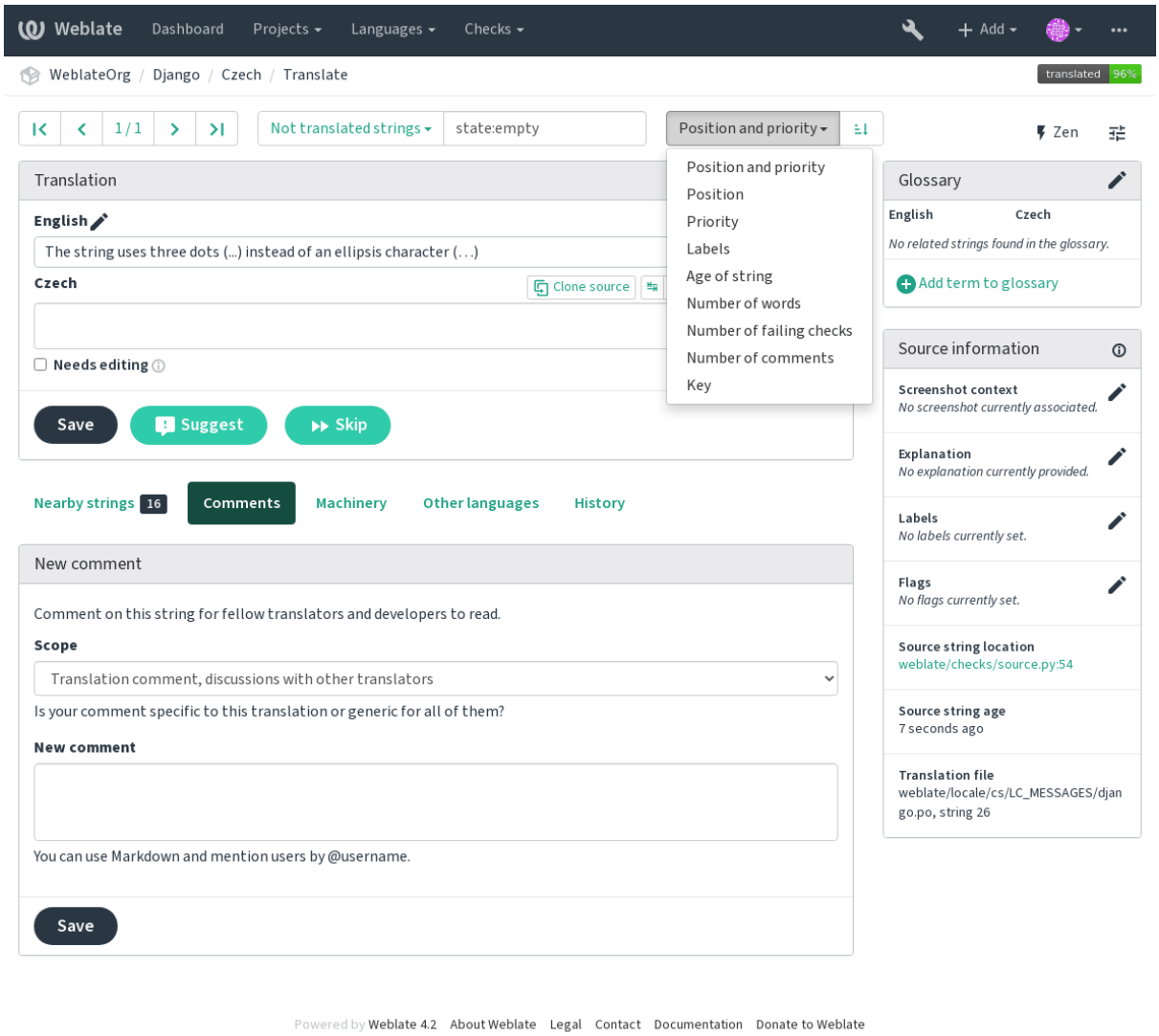
1.6.7 Предопределённые запросы

На странице поиска можно выбрать один из предопределённых запросов, которые позволяют вам быстро получить доступ к наиболее часто встречающимся поисковым запросам:


Weblate
Dashboard
Projects
Languages
Checks
Add
WeblateOrg / Django / Czech / Translate
translated 96%
1 / 1
Custom Search
'%(count)s word'
Position and priority
Zen
Translation
English
Singular
%(count)s word
Plural
%(count)s words
Czech, One
%(count)s slovo
Czech, Few
%(count)s slova
Czech, Other
%(count)s slov
Plural formula: (n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2
Needs editing
Save
Suggest
Skip
Not translated strings • state:empty
Strings needing action • state:<translated
Translated strings • state:>=translated
Strings marked for edit • state:needs-editing
Strings with suggestions • has:suggestion
Strings with variants • has:variant
Strings with labels • has:label
Strings with context • has:context
Strings needing action without suggestions • state:<translated AND NOT has:suggestion
Strings with comments • has:comment
Strings with any failing checks • has:check
Approved strings • state:approved
Strings waiting for review • state:translated
Clone source
NBS
15/140
14/140
Nearby strings 20
Comments
Machinery
Other languages
History
New comment
Comment on this string for fellow translators and developers to read.
Scope
Translation comment, discussions with other translators
Is your comment specific to this translation or generic for all of them?
New comment
You can use Markdown and mention users by @username.
Save
Czech
Add term to glossary
Labels
No labels currently set.
Flags
python-format
Source string location
weblate/templates/translation.html:149
Source string age
5 seconds ago
Translation file
weblate/locale/cs/LC_MESSAGES/django.po, string 5
Powered by Weblate 4.2
About Weblate
Legal
Contact
Documentation
Donate to Weblate

1.6.8 Упорядочивание результатов

Существует множество вариантов сортировки строк в соответствии с вашими потребностями:



1.7 Руководство разработчика приложений

Использование Weblate — это процесс, который приближает пользователей к вам путём приближения вас к вашим переводчикам. Вы сами решаете, какой объём его возможностей вы хотите использовать.

1.7.1 Знакомство с интернационализацией

У вас есть проект и вы хотите перевести его на несколько языков? Это руководство вам в этом поможет. В нём приведены несколько типичных ситуаций, но большинство примеров достаточно общи и могут быть применены также и к другим сценариям.

Перед тем, как переводить какое-либо программное обеспечение, вы должны осознать, что языки в мире сильно отличаются друг от друга, так что вы не должны делать никаких предположений, основанных на вашем опыте. Если вы попытаетесь собрать предложение из отдельно переведённых сегментов, в большинстве языков оно будет выглядеть очень странно. Также вы должны правильно обрабатывать множественные формы, поскольку во многих языках они строятся по сложным правилам, а используемый вами фреймворк интернационализации должен их поддерживать.

И последнее, но не менее важное: иногда в переведённую строку может понадобиться добавить некий контекст. Представьте себе, что переводчик получил для перевода строку `Sun`. Без контекста большинство людей перевели бы её как «Солнце», подразумевая, что речь идёт о нашей ближайшей звезде, но на самом деле она могла бы быть аббревиатурой для воскресенья (**Sun**day).

Выбор фреймворка интернационализации

Выбирайте тот фреймворк, который является стандартом для вашей платформы, постарайтесь не изобретать заново колесо, создавая свой собственный фреймворк для работы с локализациями. Weblate поддерживает большинство широко используемых фреймворков, подробнее смотрите раздел *Поддерживаемые форматы файлов* (особенно его подраздел *Возможности типов переводов*).

Наши личные рекомендации для некоторых платформ приведены в следующей таблице. Эти рекомендации основаны на нашем опыте, но он не всеобъемлющ, поэтому, когда делаете выбор, всегда учитывайте своё окружение.

Платформа	Рекомендуемый формат
Android	*Строковые ресурсы Android*
iOS	*Строки Apple iOS*
Qt	*Файлы .ts Qt Linguist*
Python	*GNU gettext*
PHP	*GNU gettext*[1]
C/C++	*GNU gettext*
C#	*.XML-файлы ресурсов*
Perl	*GNU gettext*
Ruby	*Файлы Ruby YAML*
Расширения браузеров	*JSON-файлы WebExtension*
Java	*XLIFF*[2]
JavaScript	*JSON -файлы i18next*[3]

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

- *Перевод программ с помощью GNU Gettext*
- *Перевод документации с помощью Sphinx*
- *Translating HTML and JavaScript using Weblate CDN*

Интеграция с Weblate

Получение из Weblate обновлений переводов

Для получения обновлённых строк из Weblate вы можете просто затянуть изменения из нижележащего репозитория (либо из файловой системы, либо его можно сделать доступным через *Git exporter*). До этого вы можете захотеть зафиксировать любые отложенные изменения (смотрите раздел *Lazy commits*). Это можно сделать либо через пользовательский интерфейс (в разделе *Обслуживание репозитория*), либо из командной строки при помощи *клиента Weblate*.

Эти действия можно автоматизировать, если вы предоставите Weblate доступ на отправку изменений в ваш репозиторий и настроите в *настройках компонента* *URL для отправки*.

См.также:

Непрерывная локализация

[1] Родная поддержка Gettext в PHP содержит множество ошибок и часто отсутствует в сборках под Windows, так что вместо неё мы рекомендуем использовать стороннюю библиотеку motranslator.

[2] Также, если множественные числа вам не нужны, вы можете использовать *Файлы свойств Java*.

[3] Также, если множественные числа вам не нужны, вы можете использовать простой *JSON-файлы*.

Отправка в Weblate изменений строк

Для отправки недавно обновлённые строки в Weblate, просто позвольте ему извлечь их из вышестоящего репозитория. Это можно сделать либо через пользовательский интерфейс (в разделе *Обслуживание репозитория*), либо из командной строки при помощи *клиента Weblate*.

Это действие можно автоматизировать путём установки в вашем репозитории веб-обработчика, запускающего Weblate по появлению нового коммита, подробнее смотрите в разделе *Updating repositories*.

См.также:

Непрерывная локализация

1.7.2 Перевод программ с помощью GNU Gettext

GNU Gettext является одним из наиболее широко используемых инструментов интернационализации свободного программного обеспечения. Он предоставляет простой, но гибкий способ локализации программного обеспечения. Он имеет отличную поддержку форм множественного числа, может добавлять к переводимым строкам дополнительный контекст и на его основе построено довольно большое количество инструментов. И конечно, он имеет отличную поддержку в Weblate (смотрите описание формата файла *GNU gettext*).

Примечание: Если вы собираетесь использовать его в несвободном программном обеспечении, пожалуйста, сначала ознакомьтесь с его лицензией, она может оказаться для вас непригодной.

GNU Gettext можно использовать со множеством различных языков (C, Python, PHP, Ruby, JavaScript и со многими другими), и обычно во фреймворках пользовательского интерфейса уже есть некоторая его поддержка. Стандартным его использованием является вызов функции *gettext()*, которая часто сокращается до своего псевдонима *_()*, чтобы сделать код более простым и лёгким для чтения.

Дополнительно он предоставляет вызовы *pgettext()*, позволяющий предоставлять дополнительный контекст для переводчиков, и *ngettext()*, который умеет обрабатывать формы множественного числа по правилам целевого языка.

Как широко распространённый инструмент, для него существует множество обёрток, значительно упрощающих его использование; вместо описанного ниже ручного вызова Gettext, вы, возможно, захотите попробовать одну из них, например, intltool.

Пример программы

Простая программа на C, использующая Gettext, может выглядеть следующим образом:

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

int main(void)
{
    int count = 1;
    setlocale(LC_ALL, "");
    bindtextdomain("hello", "/usr/share/locale");
    textdomain("hello");
    printf(
        ngettext(
            "Orangutan has %d banana.\n",
            "Orangutan has %d bananas.\n",
            count
        ),
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        count
    );
    printf("%s\n", gettext("Thank you for using Weblate."));
    exit(0);
}
```

Извлечение строк для перевода

После расстановки по коду вызовов gettext, вы можете воспользоваться программой **xgettext** для извлечения из кода сообщений и записи их в .pot-файлы:

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

Примечание: Существуют альтернативные программы для извлечения строк из кода, например pybabel.

После этого вызова создастся шаблонный файл, который можно использовать для начала новых переводов (используя программу **msginit**) или обновления существующих после изменения кода (для этого можно использовать программу **msgmerge**). В результате получается простой структурированный текстовый файл:

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

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

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

Каждая запись с `msgid` в начале определяет переводимую строку текста, специальная пустая запись в начале файла — это заголовок файла, содержащий метаданные о переводе.

Начало нового перевода

Имея шаблон, мы можем начать наш первый перевод:

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

Только что созданный файл `cs.po` уже содержит некоторую информацию. Самое главное заключается в том, что в нём правильно определены формы множественного числа для выбранного языка, и вы можете видеть, что, отражая этот факт, количество форм множественного числа изменилось:

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

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

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

Этот файл компилируется в оптимизированную двоичную форму — файл .mo, используемый функциями GNU Gettext времени выполнения.

Обновление строк

После добавления в вашу программу ещё нескольких строк или после изменения старых, вы снова выполняете команду **xgettext**, которая пересоздаёт файл шаблона:

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

Затем вы можете обновить отдельные файлы перевода, чтобы они соответствовали вновь созданным шаблонам (обновление, в том числе, переупорядочивает строки для соответствия их новому шаблону):

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

Импорт в Weblate

Для импорта такого перевода в Weblate, всё, что вам нужно — это определить следующие поля при создании компонента (подробное описание полей смотрите в разделе в *Конфигурация компонента*):

Поле	Значение
Репозиторий исходного кода	URL-адрес репозитория системы контроля версий с вашим проектом
Маска файла	`po/*.po`
Шаблон для новых переводов	`po/hello.pot`
Формат файла	Выберите *gettext PO-файл*
Новый язык	Выберите *Создать новый язык*

Вот и всё, теперь вы готовы начать переводить свою программу!

См.также:

Пример использования Gettext со многими языками можно найти в проекте Weblate Hello на GitHub'е: <https://github.com/WeblateOrg/hello>.

1.7.3 Перевод документации с помощью Sphinx

Sphinx — это инструмент для создания красивой документации. Он использует простой синтаксис reStructuredText и может генерировать вывод во многих форматах. Если вы ищете пример, то эта документация также собирается с его помощью. Очень полезным спутником при использовании Sphinx является сервис Read the Docs, который бесплатно будет собирать и публиковать вашу документацию.

Я не буду заострять внимание на написание самой документации, если вы нуждаетесь в консультации, просто следуйте инструкциям на сайте Sphinx'а. После того, как ваша документация будет написана, её перевод будет довольно простым делом, поскольку Sphinx уже поддерживает для этого всё необходимое, что хорошо описано в его документации по интернационализации. Это буквально вопрос настройки нескольких параметров и вызова инструмента `sphinx-intl`.

Если вы пользуетесь сервисом Read the Docs, вы запустить сборку переведённой документации на Read the Docs. Их документация по локализации покрывает практически всё, что вам нужно знать — создание ещё одного проекта, установку его языка и связывание его с основным проектом в качестве перевода.

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

Таблица 1: Конфигурация компонента

Название компонента	`Documentation`
Маска файла	`docs/locales/*/LC_MESSAGES/index.po`
Шаблон для новых переводов	`docs/locales/index.pot`
Формат файла	*gettext PO file*
Флаги перевода	`rst-text`

Таблица 2: Component discovery configuration

Регулярное выражение для сопоставления с файлами перевода	`docs/locales/(?P<language>[^/.]*)/LC_MESSAGES/(?P<component>[^/]*)\.po`
Настроить название компонента	`Documentation: {{ component|title }}`
Задать базовый файл для нового перевода	`docs/locales/{{ component }}.pot`

Подсказка: Would you prefer Sphinx to generate just single PO file? There is a hacky way to achieve this (used by

Weblate documentation) by overriding Sphinx way to get a Gettext domain of a document. Place following snippet to your Sphinx configuration in `conf.py`:

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

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

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

См.также:

Документация по python'ьему модулю Odorik собрана при помощи Sphinx и Read the Docs и переведена с помощью Weblate.

1.7.4 Translating HTML and JavaScript using Weblate CDN

Starting with Weblate 4.2 it is posible to export localization to a CDN using *CDN локализации JavaScript'а* addon.

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

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

Creating component

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

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

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

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

Configuring Weblate CDN addon

The *CDN локализации JavaScript'а* addon provides few configuration options.

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

Селектор CSS Configures which strings from the HTML documents are translatable, see *String extraction for Weblate CDN* and *HTML localization using Weblate CDN*.

Имя языковой куки Name of cookie which contains user selected language. Used in the JavaScript snippet for *HTML localization using Weblate CDN*.

Извлечение строк из HTML-файлов List of files in the repository or URLs where Weblate will look for translatable strings and offer them for a translation, see *String extraction for Weblate CDN*.

String extraction for Weblate CDN

Строки перевода должны присутствовать в Weblate. Вы можете управлять ими вручную, использовать API для их создания или перечислять файлы или URL-адреса, используя :guilabel: *Извлечь строки из файлов HTML*, и Weblate извлечет их автоматически. Файлы должны присутствовать в репозитории или содержать удаленные URL-адреса, которые будут регулярно загружаться и анализироваться Weblate.

В конфигурации по умолчанию для *CSS selector* извлекаются элементы с CSS класса `l10n`, например, извлекаются две строки из следующих фрагментов:

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

Если вы не хотите изменять существующий код, вы также можете использовать `` * `` в качестве селектора для обработки всех элементов.

Примечание: Сейчас извлекается только текст элементов. Этот аддон не поддерживает локализацию атрибутов элементов или элементов с дочерними элементами.

HTML localization using Weblate CDN

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

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

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

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

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

Локализация JavaScript

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

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

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

1.7.5 Предупреждения компонента перевода

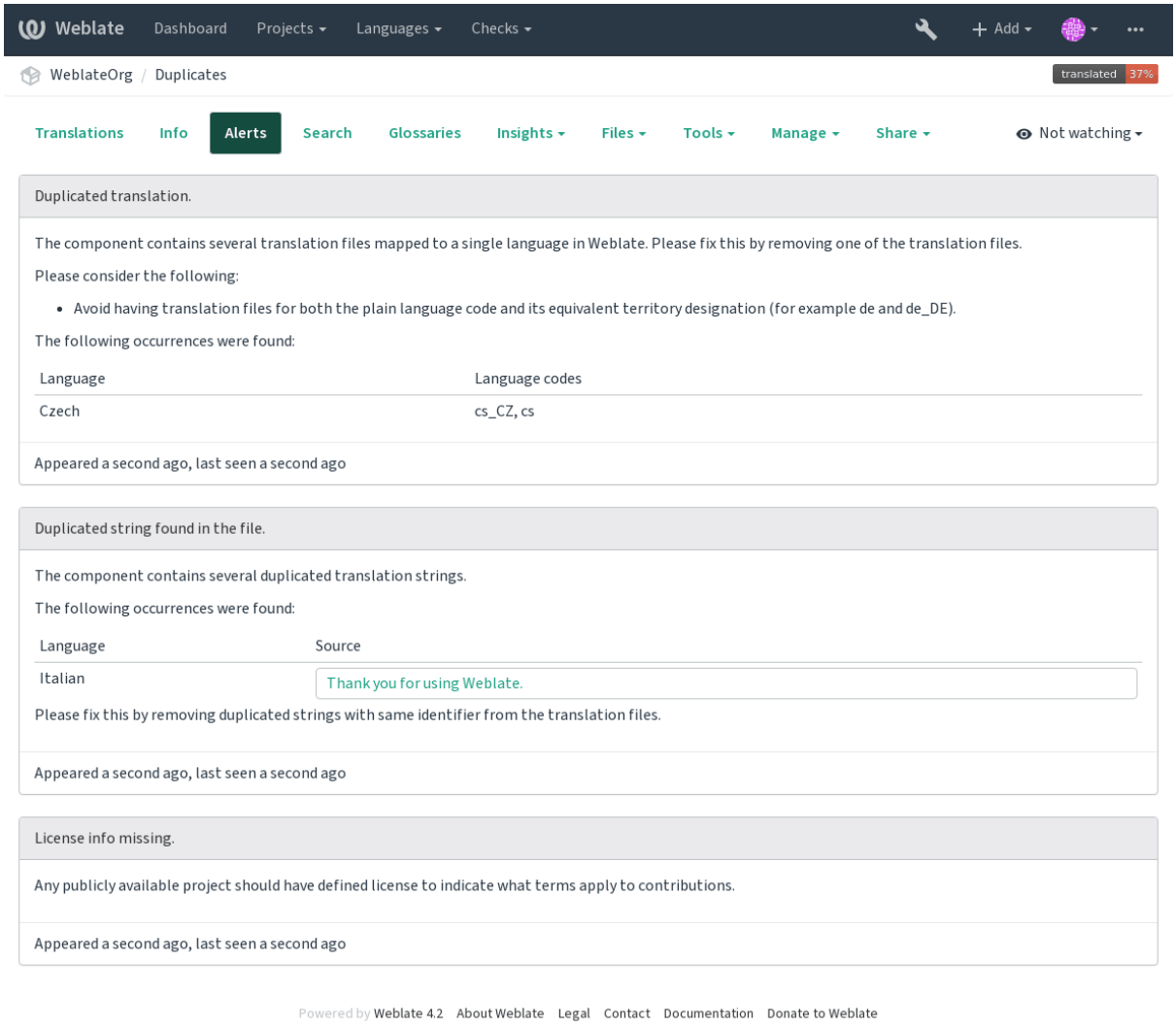
Показывает ошибки в конфигурации Weblate или в проекте перевода для любого компонента перевода. Также предлагает руководство по решению найденных проблем.

В настоящее время проверяются следующие вещи:

- Дублирование исходных строк в файлах перевода
- Дублирование языков в переводах
- Сбои при объединении или при обновлении исходного репозитория
- Неиспользованный новый базовый файл в настройках компонента
- Синтаксические ошибки в файлах перевода
- Дублирование маски файлов, используемой для связанных компонентов
- Сломанные URL-адреса

Предупреждения выводятся на соответствующей странице компонента в разделе *Предупреждения*. Если они отсутствуют, компонент очищает все текущие проверки. Предупреждения не могут быть проигнорированы, но исчезнут, как только нижележащая проблема будет устранена.

Компонент с дубликатами исходных строк и языков выглядит вот так:



См.также:

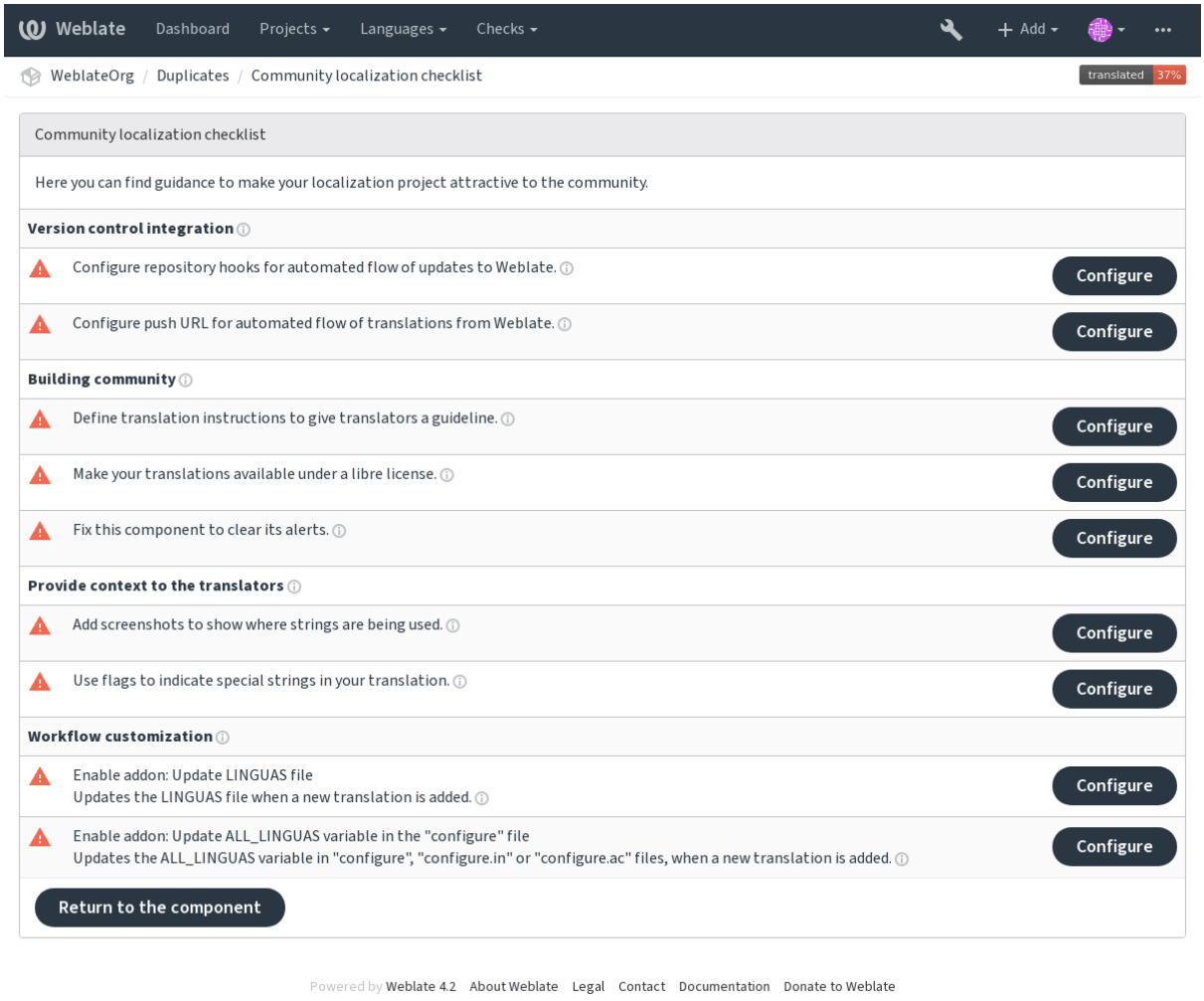
Использование пользовательского центра сертификации

1.7.6 Создание сообщества переводчиков

Контрольный список локализации сообщества

Добавлено в версии 3.9.

Контрольный список локализации сообщества, который можно найти в меню *Меню* каждого компонента, может дать вам рекомендации, выполнив которые, вы упростите сообществу переводчиков процесс локализации вашей программы.



1.7.7 Управление переводами

Добавление новых переводов

Новые строки становятся доступными для перевода когда они появляются в базовом файле, называемом *Шаблон для новых переводов* (смотрите раздел *Конфигурация компонента*). Если ваш формат файла не требует такого файла, как и происходит в большинстве рабочих процессах одноязычных переводов, вы можете начать с пустых файлов).

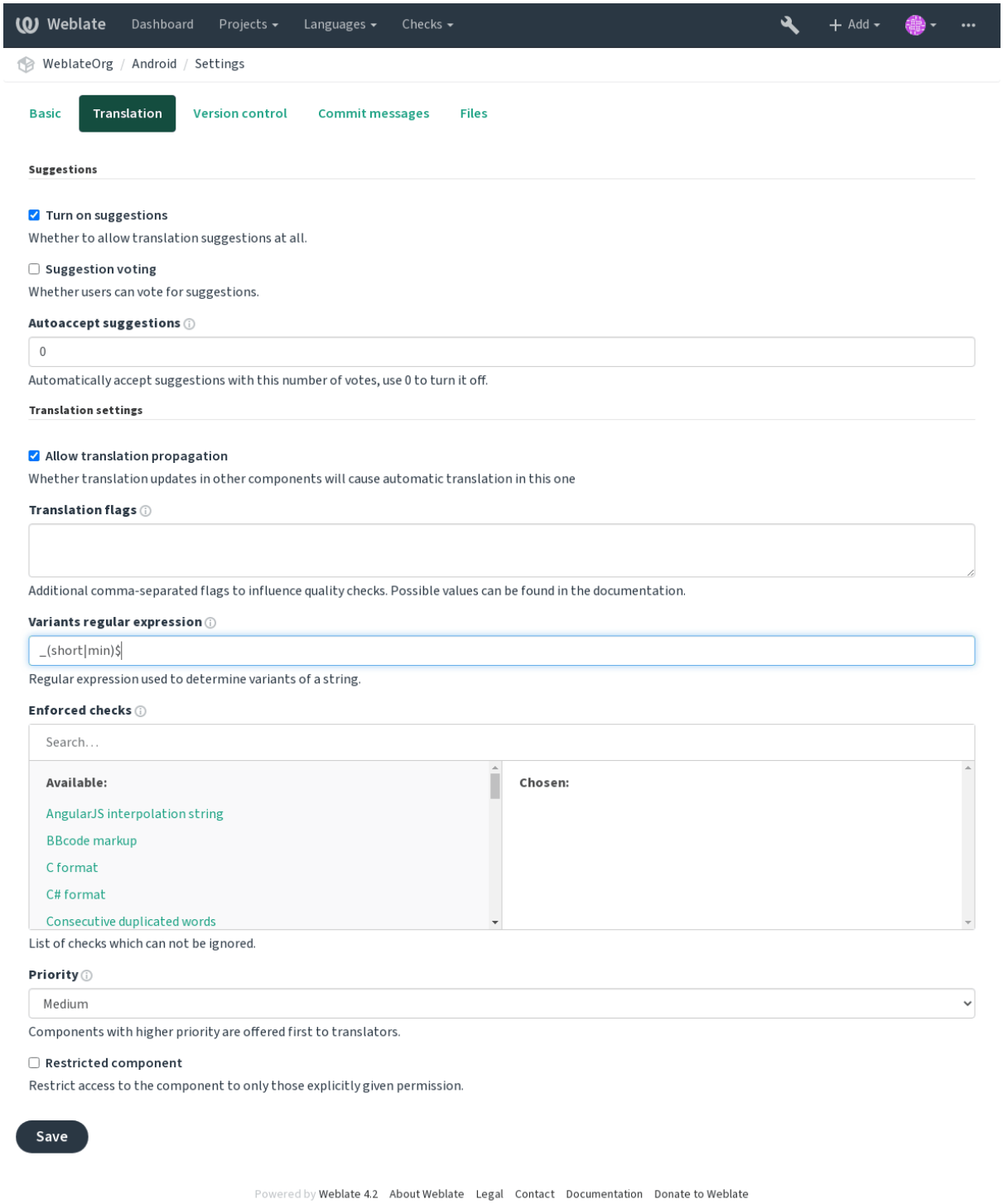
Новые языки могут быть добавлены сразу же, как пользователь их запросит в Weblate, либо же администраторам проекта будет отправлено уведомление, чтобы они одобрили запрос и добавили новый язык вручную. Это можно сделать с помощью кнопки *Начать новый перевод компонента*.

Примечание: Администраторы проектов всегда могут начать перевод напрямую в Weblate.

Языковые файлы, вручную добавленные в систему контроля версий, добавляются в компонент, когда Weblate обновляет репозиторий. Для получения информации о настройках обновления репозитория смотрите раздел *Updating repositories*.

Варианты строк

Варианты полезны для группировки нескольких строк, чтобы переводчики могли видеть все варианты строки в одном месте. Вы можете задать регулярное выражение для группировки строк в *настройках компонента*:

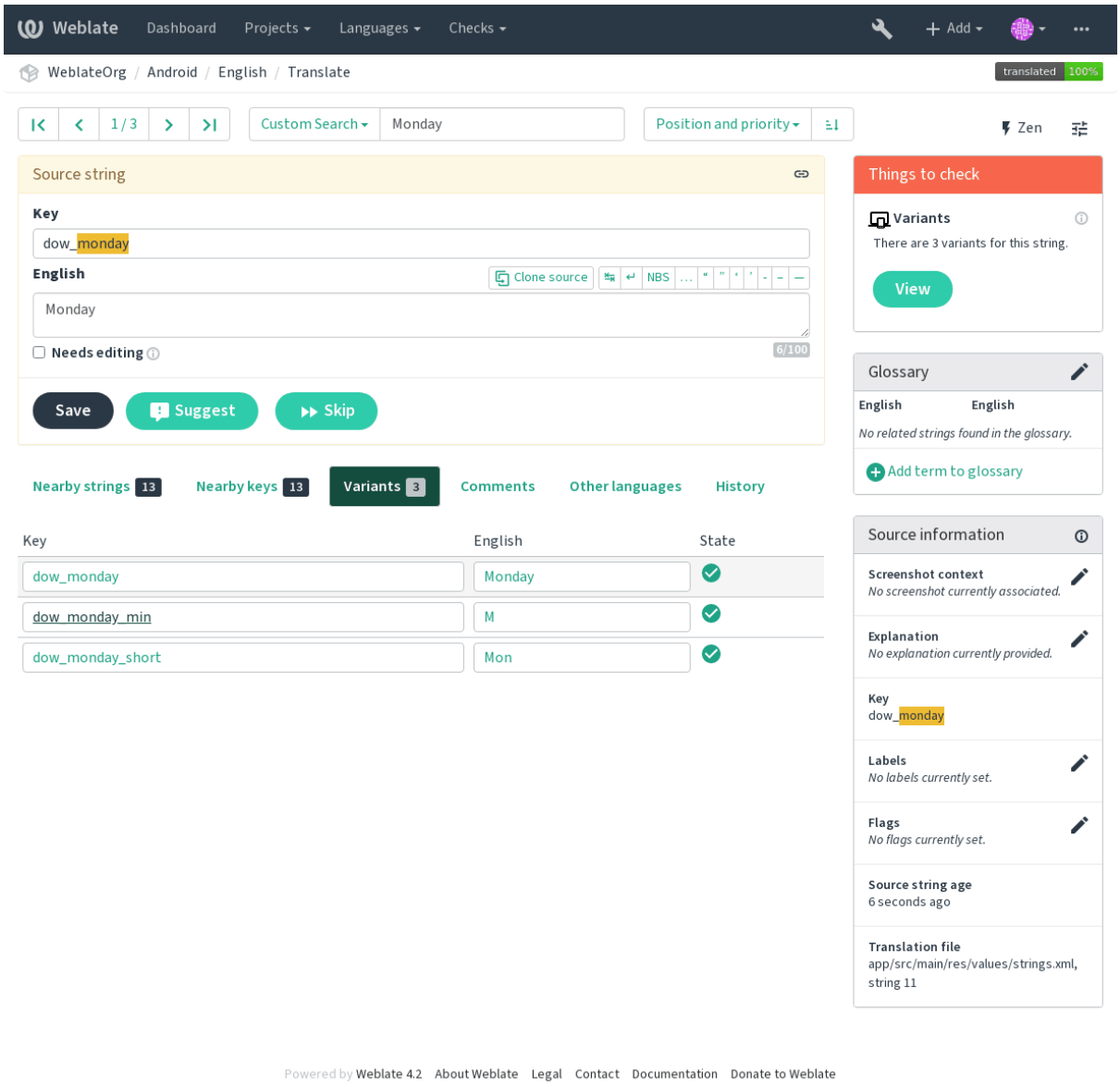


Это выражение сопоставляется с *Ключ*ом для генерации корневого ключа варианта. После этого все найденные строки становятся частью одной группы вариантов, включая перевод, точно совпадающий с корневым ключом, даже если он не подпадает под регулярное выражение.

В следующей таблице приведены некоторые примеры использования:

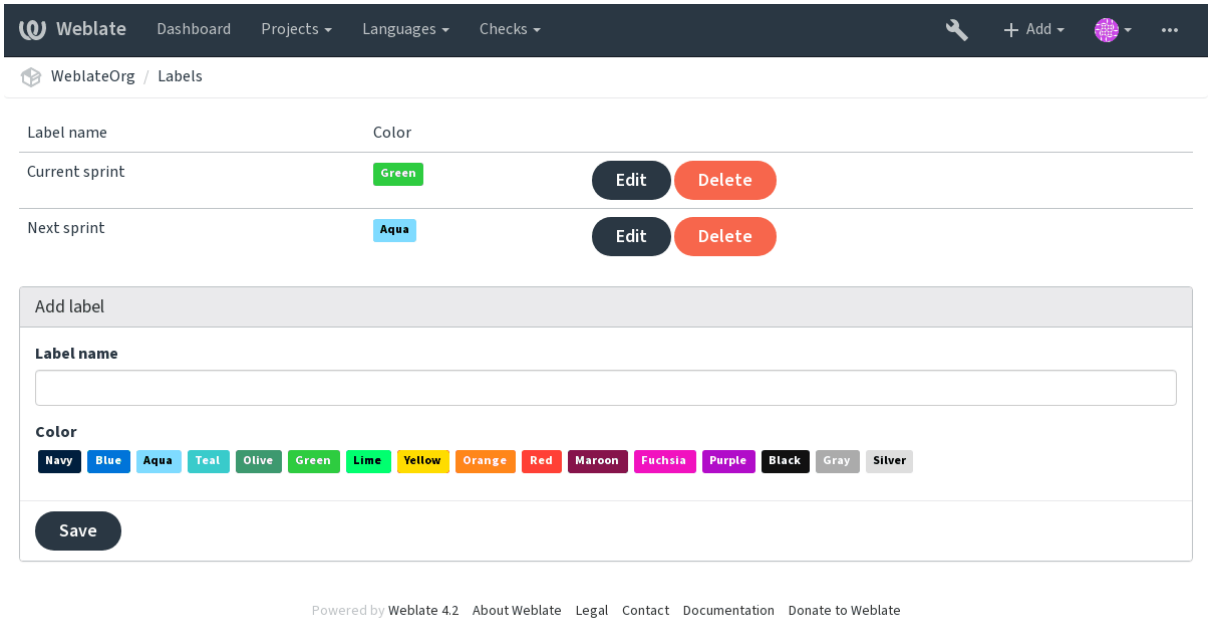
Пример использования	Регулярное выражение варианта	Соответствующие ключи перевода
Идентификация по суффиксу	`(Short|Min)$`	`monthShort`, `monthMin`, `month`
Идентификация по части строки	`#[SML]`	`dial#S.key`, `dial#M.key`, `dial.key`

Позже при переводе строки с одним вариантом группируются вместе:



Метки строк

Разбивают переводимые строки компонента на категории по тексту и цвету, настроенным в конфигурации проекта.



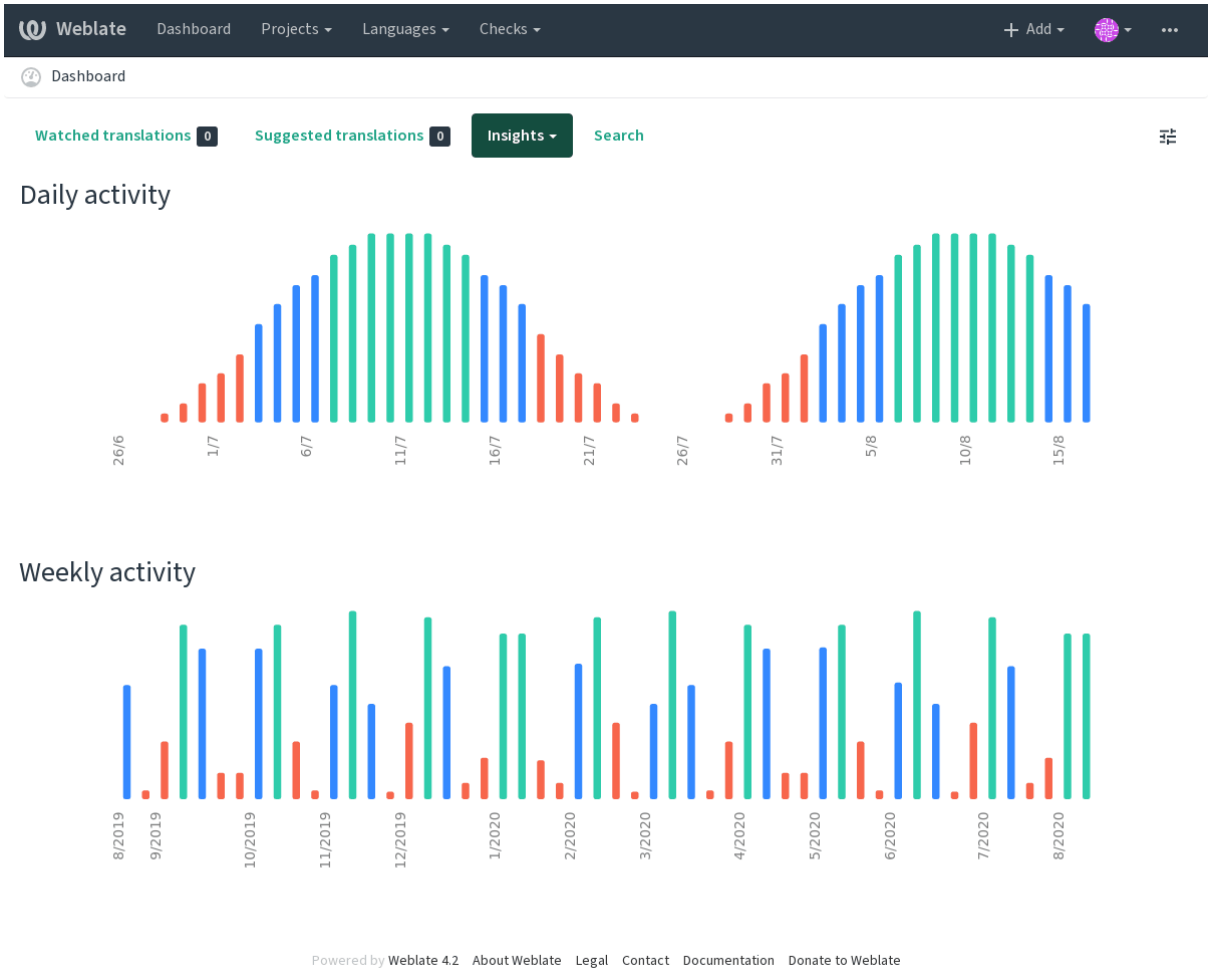
Подсказка: Метки могут быть назначены единицам перевода в окне *Additional info on source strings* путём массовой правки строк или с помощью надстройки *Массовая правка*.

1.7.8 Рецензирование строк

Отчёты о деятельности

Отчёты о деятельности показывают изменения переводов для проектов, компонентов или отдельных пользователей.

Отчёты о деятельности для проекта или компонента доступны из его панели управления, на вкладке *Аналитика*, в подпункте *Деятельность*.



Другие отчеты доступны на вкладке *Аналитика* в подпункте *Отчёты по переводам*.

Активность зарегистрированного в данный момент пользователя можно увидеть, нажав на пункт *Профиль* в меню пользователя из правого верхнего угла.

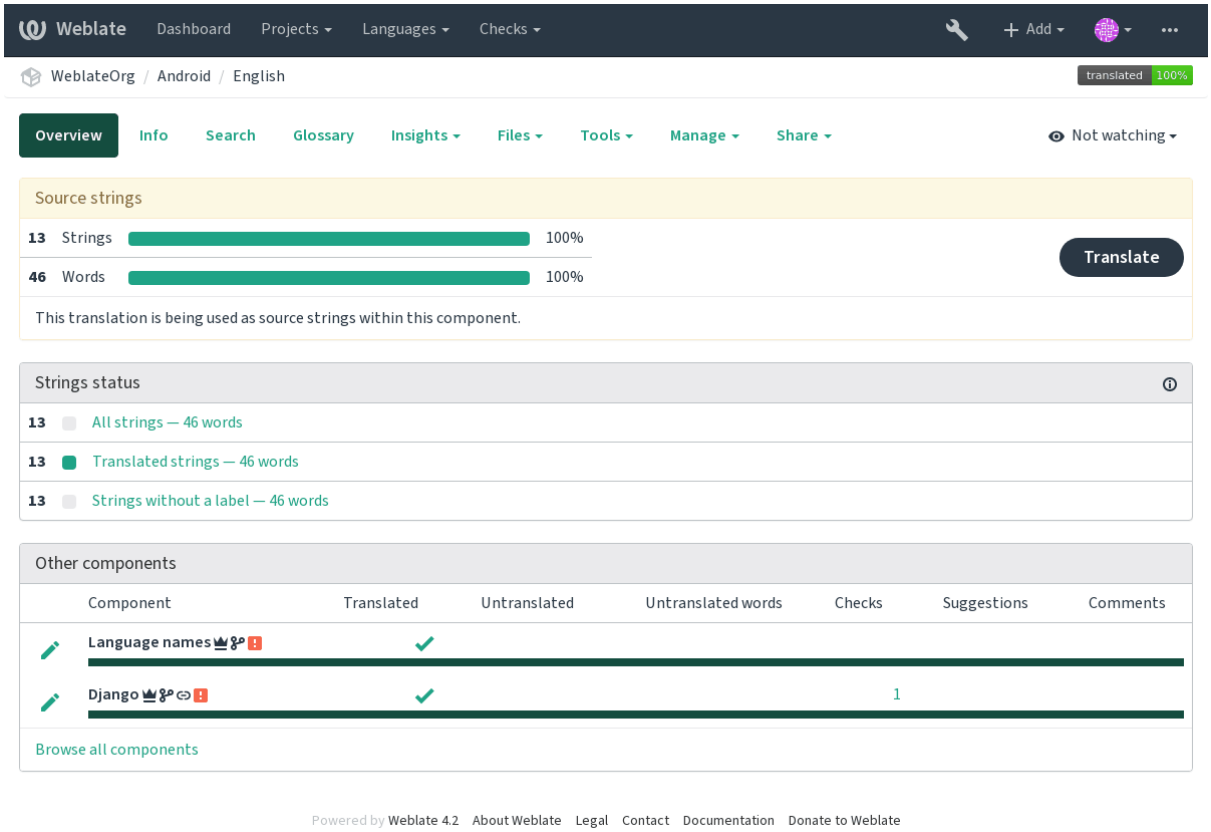
Проверки исходных строк

В Weblate существует множество *проверок* и некоторые из них направлены на улучшение качества исходных строк. Множество неудачных проверок косвенно намекают на то, что исходные строки нужно сделать более дружественными к их переводу. Все типы неудачных проверок исходных строк отображаются на вкладке *Источник* каждого компонента.

Проверки переведённых строк

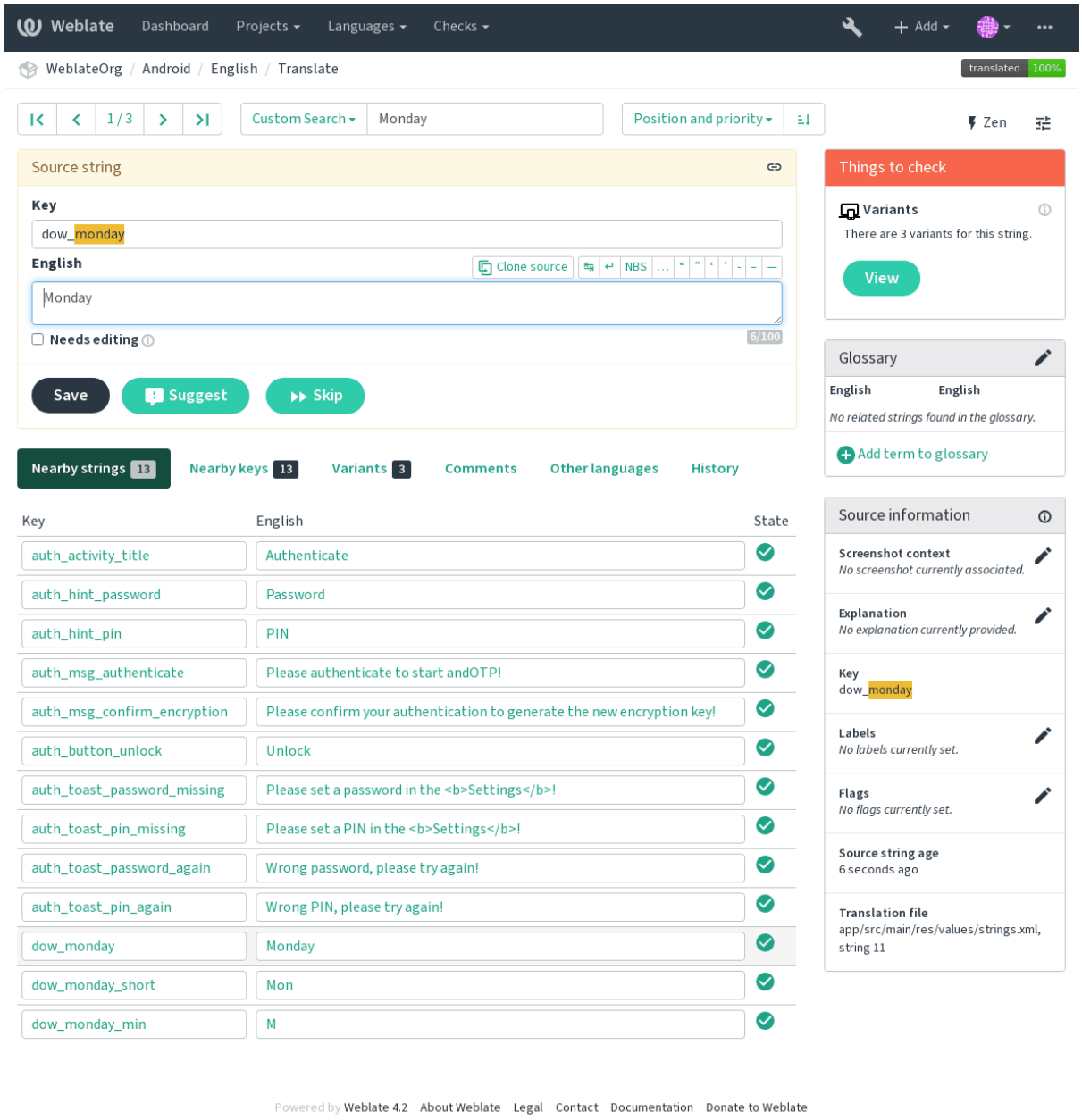
Ошибочные неудачные проверки переведённых строк говорят о том, что проблема находится в исходной строке. Иногда переводчики вместо того, чтобы сообщать о ошибках в исходных строках, просто исправляют их в переводе; типичный такой пример — пропущенная точка в конце предложения.

Просмотр всех неудачных проверок может дать ценную обратную связь для улучшения исходных строк проекта. Чтобы упростить рецензирование исходных строк, Weblate автоматически создаёт фиктивный перевод на исходный язык и показывает в нём проверки для исходных строк:



Одной из наиболее интересных проверок здесь является проверка *Множество неудачных проверок* — она возникает всякий раз, когда возникает неудачная проверка в нескольких переводах указанной строки. Обычно эта строка и оказывается тем, что вы искали — поскольку на ней у переводчиков возникли проблемы с её правильным переводом.

По каждому языку даётся подробный список:

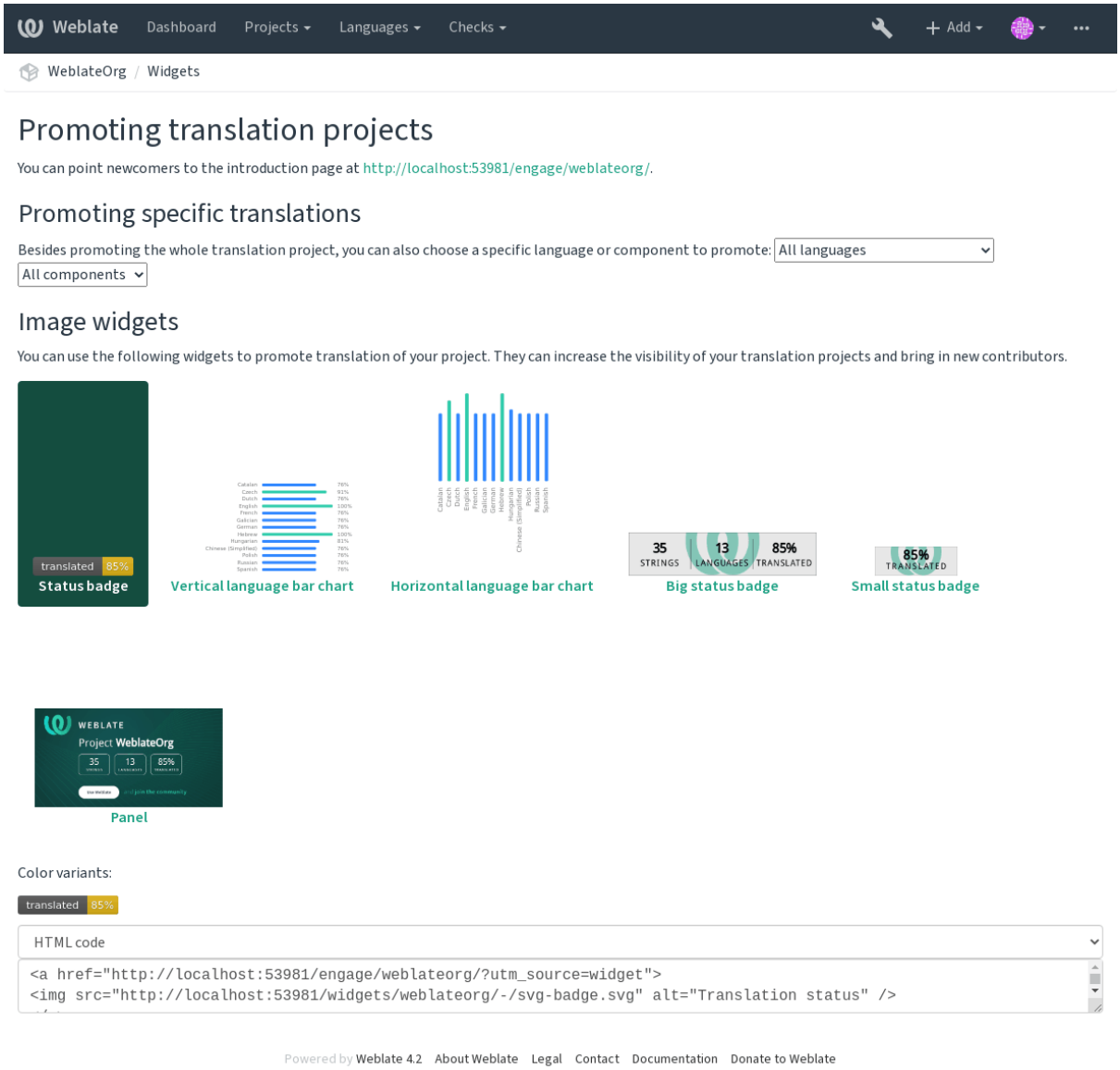


Комментарии к строкам

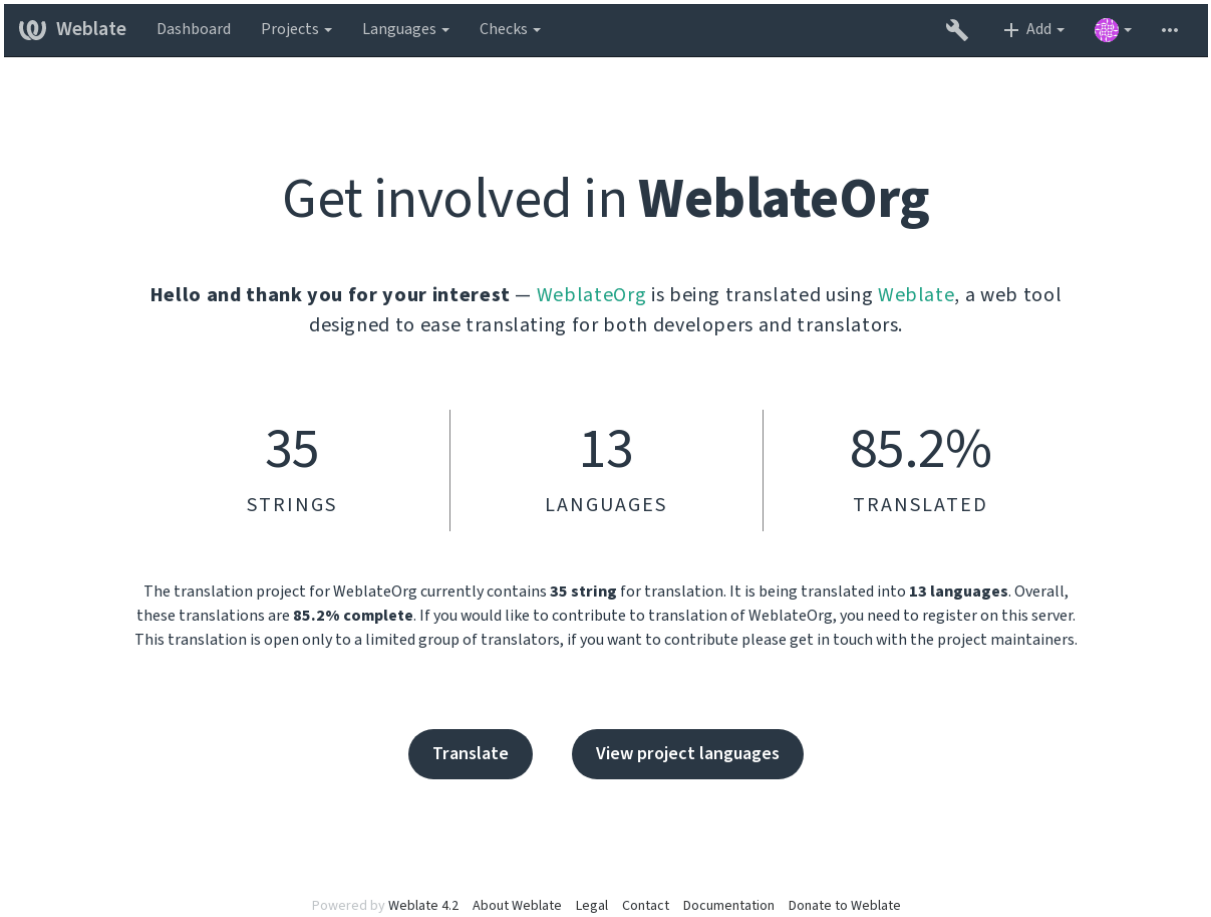
Переводчики могут комментировать как перевод, так и исходные строки. В *конфигурации* каждого компонента можно настроить отправку таких комментариев на электронный адрес; как правило, наилучшим подходом является указание там адреса списка рассылки для разработчиков. Таким образом, вы сможете следить за возникновением проблем в переводе, вовремя о них узнавать и быстро их устранять.

1.7.9 Продвижение перевода

Weblate предоставляет вам виджеты, которые вы можете разместить на своем сайте или в других источниках для продвижения проекта перевода. Ещё он предоставляет красивую страницу приветствия для новых участников, на которой они могут получить основную информацию о переводе. Кроме того, вы можете поделиться информацией о переводе через Facebook или Twitter. Все эти возможности можно найти на вкладке *Поделиться*:

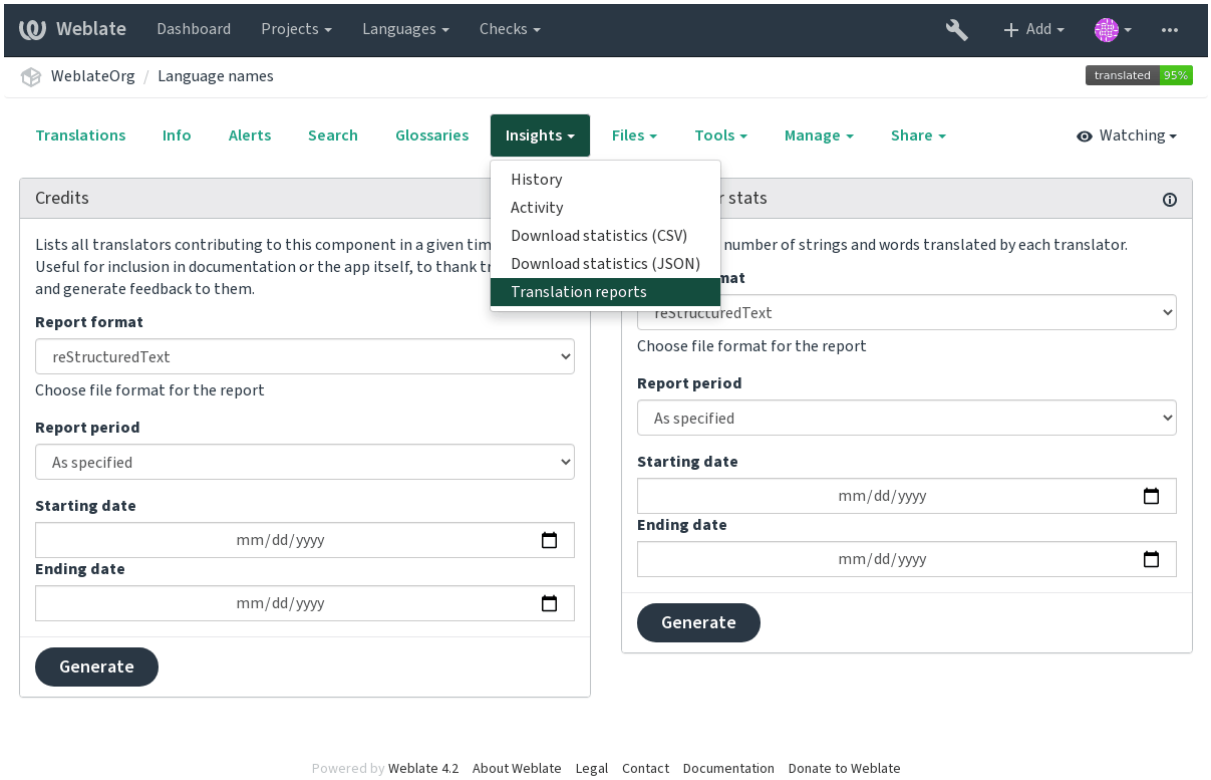


Все эти значки снабжаются ссылкой на простую страницу, которая объясняет пользователям, как переводить с помощью Weblate:



1.7.10 Отчёт о прогрессе перевода

Функции отчётности дают вам представление о том, как продвигался перевод за определённый период. В отчёте содержится сводка по переводам того или иного компонента за определённый период времени. Инструмент для создания отчётов находится в меню *Аналитика* любого компонента перевода, проекта или на панели управления:



На этой странице доступно несколько инструментов создания отчётов и все они могут производить вывод в HTML, reStructuredText или JSON. Первые два формата подходят для встраивания статистики в существующую документацию, в то время как JSON полезен для дальнейшей обработки данных.

Списки переводчиков

Генерирует документ, пригодный для выражения благодарности переводчикам — отсортированный по языку и содержащий список всех участников, внёсших вклад в перевод на данный язык:

```
* Czech

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

* Dutch

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

Будет отрисован как:

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

Подсказка: Число в скобках указывает количество вкладов за указанный период.

Статистика участника

Генерирует количество переведённых слов и строк по имени переводчика:

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

И будет отрисован в виде:

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

Он может быть полезен, если вы платите переводчикам в зависимости от объёма выполненной ими работы, он даёт вам различную статистику по работе переводчиков.

Вся статистика доступна в трёх вариантах:

Итоги Общее количество отредактированных строк.

Новые Недавно переведённые строки, у которых раньше перевода не было.

Одобренные Считает количество одобренных строк в процессе рецензирования (смотрите раздел *Выделенные рецензенты*).

Отредактированные Отредактированные строки, которые ранее уже были переведены.

Для каждого из этих вариантов доступны следующие метрики:

Количество Количество строк.

Правки Количество правок в строке, измеряемое в расстоянии Дамерау — Левенштейна.

Исходных слов Количество слов в исходной строке.

Исходных символов Количество символов в исходной строке.

Целевых слов Количество слов в переведённой строке.

Целевых символов Количество символов в переведённой строке.

1.8 Рабочие процессы перевода

Weblate поддерживает несколько рабочих процессов перевода.

Ниже приведён не полный список способов настройки Weblate. Вы можете основывать другие рабочие процессы на перечисленных здесь самых обычных примерах.

1.8.1 Доступ к переводу

Управление доступом рассматривается в рабочих процессах не слишком подробно, поскольку каждый параметр контроля доступа может быть применён к любому рабочему процессу. Пожалуйста, ознакомьтесь с этой документацией для получения информации о том, как управлять доступом к переводам.

В следующих главах *любой пользователь* означает пользователя, имеющего доступ к переводу. Это может быть любой аутентифицированный пользователь, если проект является публичным, или пользователь, имеющий в проекте разрешение *Перевод*.

1.8.2 Состояния перевода

Каждая переведённая строка может находиться в одном из следующих состояний:

Не переведёно Перевод пуст, он может как храниться в файле, так и не храниться в нём, в зависимости от формата файла.

На правку Перевод нуждается в редактировании, обычно это результат изменения исходной строки. Перевод хранится в файле, в зависимости от формата файла он может быть помечен как нуждающийся в редактировании (например, получить флаг неточного перевода).

Ожидает рецензирования Перевод выполнен, но ещё не проверен. Он хранится в файле как готовый перевод.

Одобрено Перевод был одобрен в процессе рецензирования. Он больше не может быть изменён переводчиками — только рецензентами. Переводчики могут только добавлять к нему предложения.

Предложения Предложения хранятся только в Weblate, а не в файле перевода.

1.8.3 Прямой перевод

Это самая обычная настройка для небольших команд, любой может переводить напрямую. Также она действует по умолчанию в Weblate.

- *Любой пользователь* может редактировать переводы.
- Предложения являются необязательным способом предложения изменений, когда переводчики не уверены в этом изменении.

Параметр	Значение	Примечание
Включить рецензии	выкл	Настраивается на уровне проекта.
Включить предложения	вкл	Полезны для пользователей в качестве способа предложения изменений, когда они в них не уверены.
Голосование по предложению	выкл	
Автоматическое принятие предложений	0	
Группа переводчиков	*Пользователи*	Или *Перевод*, если включено *Управление доступом*.
Группа рецензентов	Н/Д	Не используется.

1.8.4 Коллегиальное рецензирование

В этом рабочем процессе любой может добавлять предложения и чтобы они стали переводом, они должны получить одобрение от другого участника (или участников).

- *Любой пользователь* может добавлять предложения.
- *Любой пользователь* может голосовать за предложения.
- Предложения становятся переводами, когда они получают заранее определённое количество голосов.

Параметр	Значение	Примечание
Включить рецензии	выкл	Настраивается на уровне проекта.
Включить предложения	вкл	
Голосование по предложению	выкл	
Автоматическое принятие предложений	1	Вы можете установить более высокое значение, чтобы потребовать больше рецензий от коллег.
Группа переводчиков	*Пользователи*	Или *Перевод*, если включено *Управление доступом*.
Группа рецензентов	Н/Д	Не используется, рецензентами являются все переводчики.

1.8.5 Выделенные рецензенты

Добавлено в версии 2.18: Соответствующий рабочий процесс рецензирования поддерживается с версии Weblate 2.18.

В рабочем процессе с выделенными рецензентами у вас есть две группы пользователей, одна из которых может отправлять переводы, а другая — их проверять, чтобы убедиться в том, что переводы согласованы и хорошего качества.

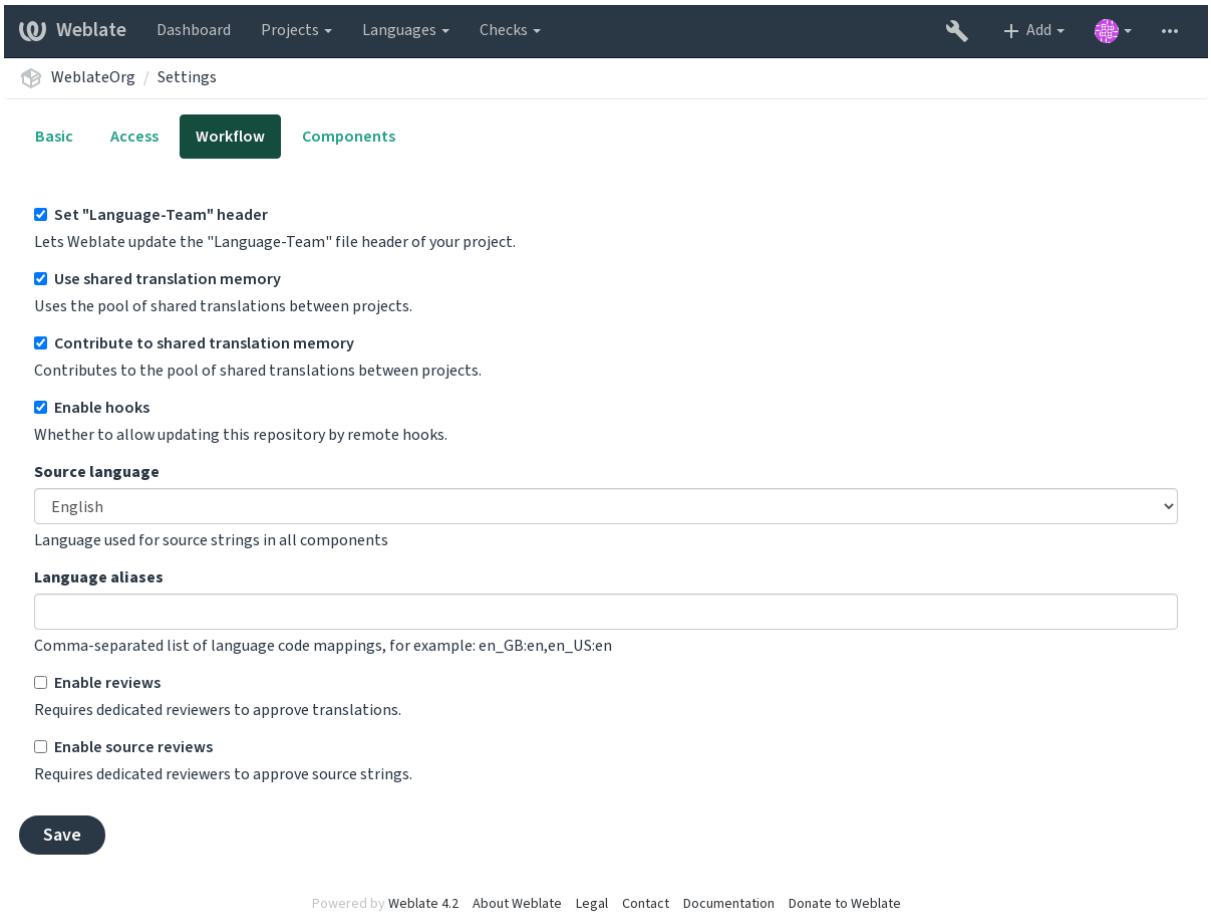
- *Любой пользователь* может редактировать не одобренные переводы.
- *Рецензент* может одобрять строки или снимать с них одобрение.

- *Рецензент* может редактировать все переводы (включая одобренные).
- Также для внесения предложений на изменение одобренных строк могут использовать предложения.

Параметр	Значение	Примечание
Включить рецензии	вкл	Настраивается на уровне проекта.
Включить предложения	выкл	Полезны для пользователей в качестве способа предложения изменений, когда они в них не уверены.
Голосование по предложению	выкл	
Автоматическое принятие предложений	0	
Группа переводчиков	*Пользователи*	Или *Перевод*, если включено *Управление доступом*.
Группа рецензентов	*Рецензенты*	Или *Рецензировать*, если включено *Управление доступом*.

1.8.6 Включение рецензирования

Рецензирование можно включить в конфигурации проекта, на странице *Рабочий процесс* настроек проекта (находится в меню *Управлять* → *Параметры*):

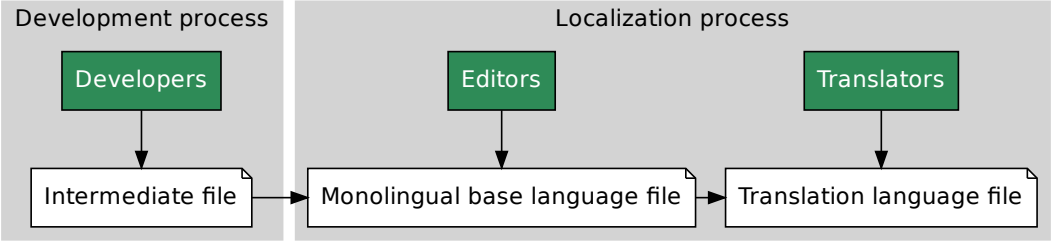


Примечание: В зависимости от конфигурации Weblate этот параметр может быть недоступен. Например, на Hosted Weblate он недоступен для проектов, размещённых бесплатно.

1.8.7 Шлюз качества для исходных строк

Во многих случаях строки на исходном языке поступают от разработчиков, потому что они пишут код и предоставляют исходные строки. Однако зачастую разработчики не являются носителями исходного языка и не обеспечивают желаемого качества исходных строк. Помочь решить эту проблему может промежуточный перевод — это дополнительный шлюз качества для строк между разработчиками, переводчиками и пользователями.

Если установлен *Файл промежуточного языка*, он будет использоваться в качестве источника исходных строк, но его можно будет отредактировать, чтобы отполировать предложения на исходном языке. Когда строка на исходном языке будет готова, она также станет доступна переводчикам для перевода на остальные языки.



См.также:

Файл промежуточного языка, *Одноязычный базовый файл*, *Двуязычные и одноязычные форматы*

1.8.8 Рецензирование исходных строк

При включённом параметре *Включить рецензии исходных строк* процесс рецензирования может быть применён также к исходным строкам. После включения этого параметра пользователи смогут сообщать о проблемах в исходных строках. Реальный процесс зависит от того, используете ли вы двуязычные или одноязычные форматы.

Для одноязычных форматов рецензирование исходной строки ничем не отличается от обычного *рецензирования* — когда сообщается о проблеме в исходной строке, она помечается флагом *На правку*.

Двуязычные форматы не позволяют напрямую редактировать исходные строки (они, как правило, извлекаются непосредственно из исходного кода). В этом случае к строкам, передаваемым переводчиками, прикрепляется метка *Исходная строка требует рецензирования*. Вы должны просмотреть такие строки и либо отредактировать их в исходном коде, либо удалить эту метку.

См.также:

Двуязычные и одноязычные форматы, *Выделенные рецензенты*, *Метки строк*

1.9 Часто задаваемые вопросы

1.9.1 Конфигурация

Как создать автоматизированный рабочий процесс?

Weblate может выполнить все действия, связанные с переводом, в полуавтоматическом режиме. Если вы дадите ему доступ на отправку изменений в ваш репозиторий, переводы смогут в нём появляться без запроса подтверждения, если только не возникнут конфликты слияния.

1. Настройте свой Git-репозиторий так, чтобы он сообщал Weblate, когда происходят какие-либо изменения; для получения информации о том, как это сделать, смотрите раздел *Обработчики уведомлений*.
2. В *конфигурации вашего компонента* установите URL для отправки, это позволит Weblate'у отправлять изменения в ваш репозиторий.
3. В *конфигурации вашего проекта* включите отправку-при-коммите, это заставит Weblate отправлять изменения в ваш репозиторий всякий раз, когда они происходят в Weblate.

См.также:

Непрерывная локализация, *Avoiding merge conflicts*

Как получить доступ к репозиториям по SSH?

Информацию о настройке SSH-ключей смотрите в разделе *Доступ к репозиториям*.

Как исправить в переводах конфликты слияния?

Время от времени, когда файл перевода изменяется одновременно и в Weblate, и в вышестоящем репозитории, происходят конфликты слияния. Обычно этого можно избежать, влив переводы из Weblate до внесения изменений в файлы перевода (например, перед запуском msgmerge). Просто скажите Weblate закоммитить все отложенные переводы (вы можете сделать это в подпункте *Обслуживание репозитория* меню *Управлять*) и влейте их в репозиторий (если не включена автоматическая отправка).

Если же вы уже нарвались на конфликт слияния, то самым простым способом решения всех конфликтов локально на вашей рабочей машине будет добавление Weblate в качестве удалённого репозитория, объединения изменений из него с изменениями из вышестоящего репозитория с исправлением возникших конфликтов. После того, как вы отправите изменения обратно, Weblate сможет использовать объединённую версию без каких-либо других специальных действий.

Примечание: В зависимости от ваших настроек, доступ к репозиторию Weblate может потребовать аутентификации. При использовании встроенного в Weblate *экспортера в Git* вы аутентифицируетесь с помощью своего имени пользователя и ключа API.

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

# Update weblate remote:
git remote update weblate

# Merge Weblate changes:
git merge weblate/master

# Resolve conflicts:
edit …
git add …
…
git commit

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

(continues on next page)

(продолжение с предыдущей страницы)

```
# Open Weblate for translation:
wlc unlock
```

Если вы используете в Weblate несколько ветвей, то же самое вы можете сделать со всеми ними:

```
# Add and update Weblate remotes
git remote add weblate-one https://hosted.weblate.org/git/project/one/
git remote add weblate-second https://hosted.weblate.org/git/project/second/
git remote update weblate-one weblate-second

# Merge QA_4_7 branch:
git checkout QA_4_7
git merge weblate-one/QA_4_7
... # Resolve conflicts
git commit

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

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

В случае использования файлов PO gettext'а существует способ полуавтоматического исправления конфликтов:

Извлеките и сохраните локальный клон Git-репозитория Weblate. Также возьмите второй свежий локальный клон вышестоящего Git-репозитория (то есть, вам нужно две копии вышестоящего Git-репозитория: нетронутая и рабочая):

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

# Update Weblate remote:
git remote update weblate

# Merge Weblate changes:
git merge weblate/master

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

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

См.также:

Как экспортировать используемый Weblate репозиторий Git?, *Непрерывная локализация*, *Avoiding merge conflicts*

Как мне сразу перевести несколько веток?

Weblate поддерживает одновременную отправку изменений перевода внутри одного *проекта*. В каждом *компоненте*, в котором она включена (это поведение по умолчанию), внесённое изменение автоматически вносится и в прочие компоненты. Таким образом, переводы остаются синхронизированными, даже если сами ветки уже достаточно сильно разошлись, и простое объединение изменений перевода между ними невозможно.

После вливания изменений из Weblate, вам, возможно, придётся объединить эти ветви (в зависимости от вашего рабочего процесса разработки) для устранения различий:

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

См.также:

Keeping translations same across components

Как переводить многоплатформенные проекты?

Weblate поддерживает широкий диапазон форматов файлов (смотрите раздел *Поддерживаемые форматы файлов*) и самым простым подходом является использование родного формата для каждой платформы.

После того, как вы добавите все файлы перевода для платформ в качестве компонентов в один проект (смотрите раздел *Adding translation projects and components*), вы сможете использовать функцию распространения перевода (включена по умолчанию и может быть отключена в *настройках компонента*) для перевода строк одновременно для всех платформ.

См.также:

Keeping translations same across components

Как экспортировать используемый Weblate репозиторий Git?

Репозиторий ничем таким не выделяется, он живёт в каталоге `DATA_DIR` и называется `vcs/<проект>/<компонент>/`. Если у вас есть SSH-доступ к этой машине, вы можете работать с репозиторием напрямую.

Для анонимного доступа вы можете запустить Git-сервер и позволить ему открыть репозиторий во внешний мир.

В качестве альтернативы вы можете использовать *Git exporter* изнутри Weblate для автоматизации этого процесса.

Какие есть способы для отправки изменений в вышестоящий репозиторий?

В значительной степени это зависит от вашей установки, в этом плане Weblate достаточно гибок. Вот несколько примеров используемых с Weblate'ом рабочих процессов:

- Weblate автоматически отправляет и вливает изменения (смотрите раздел *Как создать автоматизированный рабочий процесс?*).
- Вы вручную приказываете Weblate отправить изменения (ему нужны права доступа на отправку изменений в вышестоящий репозиторий).
- Кто-то вручную вливает изменения из git-репозитория Weblate в вышестоящий репозиторий.
- Кто-то переписывает созданную Weblate историю (например, устраняя коммиты слияния), объединяет изменения и приказывает Weblate'у сбросить содержимое в вышестоящий репозиторий.

Конечно, вы можете смешивать все эти рабочие процессы по своему усмотрению.

Как мне ограничить Weblate доступом только к переводам, не раскрывая ему при этом исходный код?

Для отделения переводов от исходного кода вы можете использовать подмодули git, при этом вы по-прежнему держите их под контролем версий.

1. Создайте репозиторий с вашими файлами перевода.
2. Добавьте его в качестве подмодуля к своему коду:

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

3. Свяжите Weblate с этим репозиторием, ему больше не нужен доступ к репозиторию, содержащему ваш исходный код.
4. Вы можете обновлять основной репозиторий переводами из Weblate, выполняя команду:

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

Для получения более подробной информации обратитесь к документации по подмодулям git.

Как мне проверить, правильно ли настроен мой Weblate?

Weblate включает в себя набор проверок конфигурации, которые можно увидеть в интерфейсе администратора, просто перейдите по ссылке *Отчёт о производительности* в интерфейсе администратора, или напрямую откройте адрес `/manage/performance/`.

Почему все коммиты от имени Weblate имеют электронную почту <noreply@weblate.org>?

Это умолчательное имя коммитящего, настроенное при создании компонента перевода. Вы в любое время можете изменить его в административной зоне сайта.

В любом случае, в качестве пользователя, сделавший перевод всё равно корректно записывается автор коммита (если это поддерживает нижележащая система контроля версий).

См.также:

Конфигурация компонента

1.9.2 Использование

Как мне отрецензировать переводы других переводчиков?

- Вы можете подписаться на любые изменения на вкладке *Уведомления*, а затем проверять вклады других переводчиков по мере их поступления по электронной почте.
- В нижней части представления перевода имеется инструмент рецензирования, с помощью которого можно просматривать переводы, выполненные другими пользователями с заданной даты.

Как мне предоставить обратную связь по исходной строке?

На контекстных вкладках, расположенных ниже перевода, вы можете использовать вкладку *Источник* для предоставления обратной связи по исходной строке или для её обсуждения с другими переводчиками.

Как я могу использовать при переводе уже существующие переводы?

- Используйте функциональность импорта для загрузки компендиума в качестве переводов, предложений или переводов, нуждающихся в рецензировании. Это лучший подход для одноразового перевода с использованием компендиума или аналогичной базы данных переводов.
- Вы можете настроить *tmserver* со всеми имеющимися у вас базами данных и позволить Weblate'у его использовать. Это хороший вариант, если вы хотите использовать его при переводе несколько раз.
- Другим вариантом является перенос всех связанных проектов на один экземпляр Weblate, что позволит автоматически подбирать переводы из других проектов.

См.также:

Машинный перевод, *Машинный перевод*

Обновляет ли Weblate помимо самих переводов также и файлы перевода?

Weblate пытается свести к минимуму изменения в файлах перевода. Для некоторых форматов файлов это, к сожалению, может привести к переформатированию файла. Если вы хотите, чтобы файл был отформатирован по-вашему, пожалуйста, используйте для этого перехватчик pre-commit.

Для одноязычных файлов (смотрите раздел *Поддерживаемые форматы файлов*) Weblate может добавлять новые строки перевода, отсутствующие как в *шаблоне*, так и в реальных переводах. Однако он не выполняет автоматическую очистку устаревших строк, так как это может привести к неожиданным результатам. Если вы хотите её выполнять, пожалуйста, установите перехватчик pre-commit, который будет производить очистку в соответствии с вашими требованиями.

Weblate также не будет пытаться каким-либо образом обновлять двуязычные файлы, так что, если вам нужны файлы `po`, обновляемые из файлов `pot`, вам нужно делать это самостоятельно.

См.также:

Processing repository with scripts

Откуда берутся определения языка и как я могу добавить свои собственные определения?

Базовый набор определений языка входит в состав Weblate и Translate-toolkit. Он охватывает более 150 языков и включает информацию о множественных формах или направлении текста.

Вы вольны определять в административном интерфейсе свои собственные языки, для этого вам просто необходимо предоставить о них необходимую информацию.

Может ли Weblate подсветить изменения в строке «на правку»?

Weblate поддерживает такую подсветку, однако, чтобы показать различия, ему нужны данные.

К примеру, для PO-файлов Gettext при обновлении PO-файлов необходимо в вызов команды **`msgmerge`** передать параметр `--previous`:

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

Для одноязычных переводов Weblate может найти предыдущую строку по её идентификатору, поэтому различия он показывает автоматически.

Почему Weblate до сих пор показывает старые строки перевода, хотя я уже обновил шаблон?

Weblate не пытается манипулировать файлами перевода никаким другим способом, кроме как разрешения переводчикам их переводить. Поэтому он также не обновляет переводимые файлы при изменении шаблона или исходного кода. Вам просто нужно сделать это вручную и внести изменения в репозиторий, после чего Weblate автоматически подхватит изменения.

Примечание: Обычно перед обновлением файлов перевода полезно влить изменения, сделанные в Weblate, в основной репозиторий, так как в противном случае вы, вероятно, столкнётесь с некоторыми конфликтами при слиянии.

Например, если вы используете PO-файлы gettext, вы можете обновить файлы перевода с помощью инструмента **msgmerge**:

```
msgmerge -U locale/cs/LC_MESSAGES/django.mo locale/django.pot
```

В случае, если вы хотите выполнять такое обновление автоматически, вы можете установить надстройку *Обновлять PO-файлы до соответствия POT (msgmerge)*.

1.9.3 Решение проблем

Запросы иногда завершаются с ошибкой «слишком много открытых файлов» (“too many open files”)

Такое иногда случается, когда ваш Git-репозиторий слишком разрастается и у вас их очень много. Сжатие Git-репозиториев улучшит эту ситуацию.

Простейший способ это сделать — запустить следующий скрипт:

```
# 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

При доступе к сайту я получаю ошибку «Плохой запрос (400)» (“Bad Request (400)”)

Скорее всего это вызвано неправильной настройкой параметра *ALLOWED_HOSTS*. Он должен содержать все имена хостов, с которых вы хотите получать доступ к вашему Weblate’у. Например:

```
ALLOWED_HOSTS = ['weblate.example.com', 'weblate', 'localhost']
```

См.также:

Настройка разрешённых хостов

Что означает сообщение «Есть больше файлов для одного языка (en)»?

Обычно оно возникает, когда у вас есть файл перевода для исходного языка. Weblate отслеживает исходные строки и резервирует для этого язык оригинала. Дополнительный файл для того же самого языка не обрабатывается.

- В случае необходимости перевода на исходный язык в настройках проекта необходимо изменить параметр *Исходный язык*.
- Если файл перевода для исходного языка не нужен, пожалуйста, удалите его из репозитория.
- Если же файл перевода для исходного языка необходим, но должен игнорироваться Weblate, пожалуйста, настройте для его исключения *языковой фильтр*.

1.9.4 Возможности

Поддерживает ли Weblate другие системы контроля версий, кроме Git и Mercurial?

В настоящее время Weblate не имеет родной поддержки ни для чего, кроме *Git* (с расширенной поддержкой *GitHub*, *Gerrit* и *Subversion*) и ref:*vcs-mercurial*, но вполне возможно дописать поддержку и других систем контроля версий.

Также для доступа к другим системам контроля версий вы можете использовать *Удалённые помощники Git* в Git'е .

Кроме того, Weblate поддерживает операции, не требующие системы контроля версий, смотрите раздел *Локальные файлы*.

Примечание: Для родной поддержки других систем контроля версий Weblate требует использования распределённых СКВ, и, вероятно, может быть настроен на работу с любой системой, помимо Git или Mercurial, но кто-то должен эту поддержку реализовать.

См.также:

Интеграция с системой контроля версий

Как Weblate сообщает о том, кто какую строку перевёл?

Каждое изменение, сделанное через Weblate, фиксируется в системе контроля версий под именем переводчика. Таким образом, каждое изменение имеет надлежащее авторство, и вы можете отследить его с помощью стандартных инструментов СКВ, которые вы используете для кода.

Кроме того, если формат файла перевода это поддерживает, он обновляет заголовки файлов и включает в них имя переводчика.

См.также:

`list_translators`, *Отчёт о прогрессе перевода*

Почему Weblate принудительно показывает все PO-файлы в одном дереве?

Weblate был спроектирован таким образом, что каждый PO-файл представляется одним компонентом. Это выгодно переводчиками, поскольку они знают, что они переводят на самом деле. Если вы считаете, что ваш проект должен переводиться как один компонент, подумайте об объединении этих po-файлов. Это облегчит жизнь даже переводчикам, не использующим Weblate.

Примечание: В случае большого спроса на эту функцию она может быть реализована в будущих версиях.

Почему Weblate использует такие коды языков, как sr_Latn или zh_Hant?

Это коды языков, определяемые **RFC 4646**, чтобы лучше показать, что это и в самом деле разные языки, вместо ранее ошибочно используемых модификаторов (для вариантов `@latin`) или кодов стран (для китайского языка).

Weblate по-прежнему понимает старые коды языков и сопоставляет их с текущими — например, `sr@latin` будет обрабатываться как `sr_Latn`, а `zh@CN` — как `sr_Hans`.

1.10 Поддерживаемые форматы файлов

Weblate поддерживает большинство форматов перевода, понимаемых translate-toolkit, однако понимание любого из них чем-нибудь да отличаются, поэтому могут возникнуть некоторые проблемы с форматами, которые были протестированы недостаточно тщательно.

См.также:

Форматы файлов, относящиеся к переводу

Примечание: При выборе формата файла для вашего приложения лучше придерживаться какого-нибудь устоявшегося формата для используемого вами инструментария/платформы. Таким образом, ваши переводчики в дополнение смогут использовать любые инструменты, к которым они привыкли, и вероятность внесения ими своего вклада в ваш проект будет выше.

1.10.1 Двуязычные и одноязычные форматы

Поддерживаются как одноязычные, так и двуязычные форматы. Двуязычные форматы хранят в одном файле два языка — исходный язык и перевод (типичными примерами являются *GNU gettext*, *XLIFF* или *Строки Apple iOS*). С другой стороны, одноязычные форматы идентифицируют строку по идентификатору, и каждый языковой файл содержит только сопоставление этих форматов с любым заданным языком (обычно это *Строковые ресурсы Android*). Некоторые форматы файлов могут использоваться и так, и эдак, подробнее смотрите ниже.

Для правильного использования одноязычных файлов Weblate требует доступа к файлу, содержащему полный список строк для перевода и их «перевод» на исходный язык — этот файл внутри Weblate называется *одноязычным базовым файлом*, хотя в вашей парадигме наименование этой сущности может отличаться.

Кроме того, этот рабочий процесс может быть расширен с помощью *файла промежуточного языка*, позволяющего включить в перевод строки, предоставленные разработчиками, но не использованными «как есть» в окончательных строках.

1.10.2 Автоматическое обнаружение

Weblate умеет автоматически обнаруживать некоторые широко распространённые форматы файлов, но это обнаружение может навредить вашей производительности и ограничено возможностями, специфичными для данного формата файла (к примеру, автоматическим добавлением новых переводов).

1.10.3 Возможности типов переводов

Возможности всех поддерживаемых форматов:

Формат	Языковость[1]	Множественные формы[2]	Комментарии[3]	Контекст[4]	Расположение[5]	Флаги[8]	Дополнительные состояния[6]
GNU gettext	двуязычный	да	да	да	да	да[9]	требуется правка
Одноязычный gettext	одноязычный	да	да	да	да	да[9]	требуется правка
XLIFF	оба	да	да	да	да	да[10]	требуется правка, одобрено
Файлы свойств Java	оба	нет	да	нет	нет	нет	
Файлы свойств GWT	одноязычный	да	да	нет	нет	нет	
Переводы Joomla	одноязычный	нет	да	нет	да	нет	
Файлы .ts Qt Linguist	оба	да	да	нет	да	да[10]	требуется правка
Строковые ресурсы Android	одноязычный	да	да[7]	нет	нет	да[10]	
Строки Apple iOS	двуязычный	нет	да	нет	нет	нет	
Строки PHP	одноязычный	нет[11]	да	нет	нет	нет	
JSON-файлы	одноязычный	нет	нет	нет	нет	нет	
JSON -файлы i18next	одноязычный	да	нет	нет	нет	нет	
JSON -файлы go-i18n	одноязычный	да	нет	нет	нет	нет	
Файл ARB	одноязычный	да	да	нет	нет	нет	
JSON-файлы WebExtension	одноязычный	да	да	нет	нет	нет	
.XML-файлы ресурсов	одноязычный	нет	да	нет	нет	да[10]	
Файлы CSV	одноязычный	нет	да	да	да	нет	требуется правка

continues on next page

Таблица 3 – продолжение с предыдущей страницы

Формат	Языковость[1]	Множественные формы[2]	Комментарии[3]	Контекст[4]	Расположение[5]	Флаги[8]	Дополнительные состояния[6]
Файлы YAML	одноязычный	нет	да	нет	нет	нет	
Файлы Ruby YAML	одноязычный	да	да	нет	нет	нет	
Файлы DTD	одноязычный	нет	нет	нет	нет	нет	
Flat XML	одноязычный	нет	нет	нет	нет	да[10]	
Файлы ресурсов Windows	одноязычный	нет	да	нет	нет	нет	
Excel Open XML	одноязычный	нет	да	да	да	нет	требуется правка
Файлы метаданных магазина приложений	одноязычный	нет	нет	нет	нет	нет	
Файлы субтитров	одноязычный	нет	нет	нет	да	нет	
Файлы HTML	одноязычный	нет	нет	нет	нет	нет	
Формат OpenDocument	одноязычный	нет	нет	нет	нет	нет	
Формат IDML	одноязычный	нет	нет	нет	нет	нет	
Переводы в INI-файлах	одноязычный	нет	нет	нет	нет	нет	
Переводы в INI-файлах Inno Setup	одноязычный	нет	нет	нет	нет	нет	

[1] Смотрите раздел *Двуязычные и одноязычные форматы*

[2] Множественные формы необходимы для правильной локализации строк, содержащих переменную количества.

[3] Комментарии могут быть использованы для указания дополнительной информации о строке для перевода.

[4] Контекст используется для различения одинаковых строк, используемых в разных областях (например, *Sun* может означать как сокращённое название дня «воскресенье» — «Sunday» — так и название нашей ближайшей звезды).

[5] Расположение строки в исходном коде может помочь опытным переводчикам понять предназначение этой строки.

[8] Смотрите раздел *Настройка поведения*

[6] Дополнительные состояния, поддерживаемые форматом файла в дополнение к состояниям «Не переведено» и «Переведено».

[9] В качестве флагов используются комментарии типа gettext.

[10] Для всех форматов на основе XML флаги извлекаются из нестандартного атрибута `weblate-flags`. Кроме того, флаг `max-length:N` поддерживается через атрибут `maxwidth`, как определено в стандарте XLIFF, смотрите раздел *Указание флагов перевода*.

[7] Комментарий XML, размещённый перед элементом `<string>`, трактуется как комментарий разработчика.

[11] Множественные формы поддерживаются только для Laravel, который использует для их определения внутристрочный синтаксис, смотрите документ Локализация в Laravel.

1.10.4 GNU gettext

Наиболее широко используемый формат для перевода свободного программного обеспечения. Этот формат Weblate поддержал самым первым, и он до сих пор имеет самую лучшую поддержку.

Контекстная информация, хранящаяся в файле, поддерживается настройкой его заголовков или его привязкой к соответствующим исходным файлам.

Двуязычный PO-файл gettext обычно выглядит следующим образом:

```
#: 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"
```

Типичная *настройка компонента* в Weblate	
Маска файла	`po/*.po`
Одноязычный базовый файл	*Пусто*
Шаблон для новых переводов	`po/messages.pot`
Формат файла	*PO-файл gettext*

См.также:

Перевод программ с помощью GNU Gettext, *Перевод документации с помощью Sphinx*, Gettext on Wikipedia, PO Files, *Обновлять переменную ALL_LINGUAS в файле «configure»*, *Настроить вывод gettext*, *Обновлять LINGUAS-файл*, *Генерировать MO-файлы*, *Обновлять PO-файлы до соответствия POT (msgmerge)*

Одноязычный gettext

Некоторые проекты решают использовать gettext в качестве одноязычного формата — в их исходном коде присутствуют только идентификаторы и строка должна быть переведена на все языки, в том числе и на английский. Такая конфигурация поддерживается, хотя для этого и надо явно выбирать этот формат файлов при импорте компонентов в Weblate.

Одноязычный PO-файл gettext обычно выглядит следующим образом:

```
#: 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"
```

А базовый языковой файл — вот таким образом:

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

(continues on next page)

(продолжение с предыдущей страницы)

```
msgstr "Monday"

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

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

Типичная *настройка компонента* в Weblate	
Маска файла	`po/*.po`
Одноязычный базовый файл	`po/en.po`
Шаблон для новых переводов	`po/messages.pot`
Формат файла	*PO-файл gettext (одноязычный)*

1.10.5 XLIFF

Формат на основе XML, созданный для стандартизации файлов перевода, но в конечном итоге ставший просто ещё одним из многочисленных стандартов в этой области.

Формат файла XML для обмена локализациями (XLIFF) обычно используется как двуязычный, но Weblate поддерживает его использование и в качестве одноязычного.

См.также:

Спецификация *формата файла XML для обмена локализациями (XLIFF)*

Состояния перевода

Изменено в версии 3.3: До выпуска 3.3 Weblate игнорировал атрибут состояния.

Атрибут состояния `state` в файле частично обрабатывается и отображается в Weblate в состояние «На правку» (следующие состояния используются для пометки строки на правку, если присутствует цель: `new`, `needs-translation`, `needs-adaptation`, `needs-l10n`). Если атрибут `state` отсутствует, то строка считается переведённой, если существует элемент `<target>`.

Если строка перевода имеет атрибут `approved="yes"`, она также будет импортирована в Weblate в состоянии «Одобрено», все остальные строки будут импортированы в состоянии «Ожидает рецензирования» (что соответствует спецификации XLIFF).

При сохранении Weblate добавляет эти атрибуты только если они действительно необходимы:

- Атрибут `state` добавляется только в том случае, если строка помечена на правку.
- Атрибут `approved` добавляется только в том случае, если строка была отрецензирована.
- В остальных случаях эти атрибуты не добавляются, но если они присутствуют, они обновляются.

Это означает, что при использовании формата XLIFF настоятельно рекомендуется включить в Weblate процесс рецензирования, чтобы видеть и изменять состояние одобрения строк.

Смотрите раздел *Выделенные рецензенты*.

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

Пробелы и переводы строк в файлах XLIFF

Как правило, тип или количество пробельных символов в XML-форматах не различаются. Если вы хотите их сохранить, вы должны добавить к строке флаг `xml:space="preserve"`.

Например:

```
    <trans-unit id="10" approved="yes">
        <source xml:space="preserve">hello</source>
        <target xml:space="preserve">Hello, world!
</target>
    </trans-unit>
```

Указание флагов перевода

С помощью атрибута `weblate-flags` вы можете указать дополнительные флаги перевода (смотрите раздел *Настройка поведения*). Также Weblate понимает атрибуты `maxwidth` и `font` из спецификации XLIFF:

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

Атрибут `font` разбирается в семейство шрифтов, размер и вес шрифта. Приведённый выше пример показывает использование всех трёх параметров, хотя к указанию обязательно только семейство шрифтов. Любые пробельные символы в наименовании семейства шрифтов преобразуются в подчёркивания, так что семейство `Source Sans Pro` становится семейством `Source_Sans_Pro`, поэтому при именовании группы шрифтов (смотрите раздел *Управление шрифтами*) имейте это в виду.

Типичная *настройка компонента* в Weblate для двуязычного XLIFF	
Маска файла	`localizations/*.xliff`
Одноязычный базовый файл	*Пусто*
Шаблон для новых переводов	`localizations/en-US.xliff`
Формат файла	*Файл перевода XLIFF*

Типичная *настройка компонента* в Weblate для одноязычного XLIFF	
Маска файла	`localizations/*.xliff`
Одноязычный базовый файл	`localizations/en-US.xliff`
Шаблон для новых переводов	`localizations/en-US.xliff`
Формат файла	*Файл перевода XLIFF*

См.также:

XLIFF в Википедии, XLIFF, Атрибут font в XLIFF 1.2, Атрибут maxwidth в XLIFF 1.2

1.10.6 Файлы свойств Java

Родной формат Java для переводов.

Файлы свойств Java обычно используются в качестве одноязычных переводов.

Weblate поддерживает варианты этого формата в кодировках ISO-8859-1, UTF-8 и UTF-16. Все они поддерживают сохранение всех символов юникода, просто в разных кодировках. В формате ISO-8859-1 используются юникодные экранирующие последовательности (например, `zkou\u0161ka`), остальные кодируют символы непосредственно в UTF-8 или UTF-16.

Примечание: Загрузка экранирующих последовательностей также работает в режиме UTF-8, поэтому, пожалуйста, будьте внимательны при выборе правильного набора кодировок в соответствии с потребностями вашего приложения.

Типичная *настройка компонента* в Weblate	
Маска файла	`src/app/Bundle_*.properties`
Одноязычный базовый файл	`src/app/Bundle.properties`
Шаблон для новых переводов	*Пусто*
Формат файла	*Свойства Java (ISO 8859-1)*

См.также:

Java properties on Wikipedia, Mozilla and Java properties files, *Форматирует файл свойств Java*, *Очистка файлов перевода*

1.10.7 Файлы свойств GWT

Родной формат GWT для переводов.

Файлы свойств GWT обычно используются в качестве одноязычных переводов.

Типичная *настройка компонента* в Weblate	
Маска файла	`src/app/Bundle_*.properties`
Одноязычный базовый файл	`src/app/Bundle.properties`
Шаблон для новых переводов	*Пусто*
Формат файла	*Свойства GWT*

См.также:

GWT localization guide Mozilla and Java properties files, *Форматирует файл свойств Java*, *Очистка файлов перевода*

1.10.8 Переводы в INI-файлах

Добавлено в версии 4.1.

Формат INI-файла для переводов.

Переводы в INI-файлах обычно используются в качестве одноязычных переводов.

Типичная *настройка компонента* в Weblate	
Маска файла	`language/*.ini`
Одноязычный базовый файл	`language/en.ini`
Шаблон для новых переводов	*Пусто*
Формат файла	*INI-файл*

См.также:

INI-файлы, *Переводы Joomla*, *Переводы в INI-файлах Inno Setup*

1.10.9 Переводы в INI-файлах Inno Setup

Добавлено в версии 4.1.

Формат INI-файлов Inno Setup для переводов.

Переводы в INI-файлах Inno Setup обычно используются в качестве одноязычных переводов.

Примечание: Единственным заметным отличием от *обычных переводов в INI-файлах* является поддержка заполнителей `%n` и `%t` для символов переноса строки и табуляции.

Типичная *настройка компонента* в Weblate	
Маска файла	`language/*.islu`
Одноязычный базовый файл	`language/en.islu`
Шаблон для новых переводов	*Пусто*
Формат файла	*INI-файл Inno Setup*

Примечание: На текущий момент поддерживаются только юникодные файлы (`.islu`), файлы в кодировке ANSI (`.isl`) в настоящее время не поддерживаются.

См.также:

INI-файлы, *Переводы Joomla*, *Переводы в INI-файлах*

1.10.10 Переводы Joomla

Добавлено в версии 2.12.

Родной формат Joomla для переводов.

Переводы Joomla обычно используются в качестве одноязычных переводов.

Типичная *настройка компонента* в Weblate	
Маска файла	`language/*/com_foobar.ini`
Одноязычный базовый файл	`language/en-GB/com_foobar.ini`
Шаблон для новых переводов	*Пусто*
Формат файла	*Языковой файл Joomla*

См.также:

Спецификация языковых файлов Joomla, Файлы свойств Mozilla и Java, *Переводы в INI-файлах*, *Переводы в INI-файлах Inno Setup*

1.10.11 Файлы .ts Qt Linguist

Формат перевода, используемый в приложениях на основе Qt.

Файлы Qt Linguist используются и как двуязычные, так и одноязычные переводы.

Типичная *настройка компонента* в Weblate при использовании формата как двуязычного	
Маска файла	`i18n/app.*.ts`
Одноязычный базовый файл	*Пусто*
Шаблон для новых переводов	`i18n/app.ru.ts`
Формат файла	*Файл перевода Qt Linguist*

Типичная *настройка компонента* в Weblate при использовании формата как одноязычного	
Маска файла	`i18n/app.*.ts`
Одноязычный базовый файл	`i18n/app.en.ts`
Шаблон для новых переводов	`i18n/app.en.ts`
Формат файла	*Файл перевода Qt Linguist*

См.также:

Руководство по Qt Linguist, Qt .ts, *Двуязычные и одноязычные форматы*

1.10.12 Строковые ресурсы Android

Специфичный для Android формат файла для перевода приложений.

Строковые ресурсы Android являются одноязычными, *Одноязычный базовый файл* хранится в другом месте, нежели остальные строки — в файле `res/values/strings.xml`.

Типичная *настройка компонента* в Weblate	
Маска файла	`res/values-*/strings.xml`
Одноязычный базовый файл	`res/values/strings.xml`
Шаблон для новых переводов	*Пусто*
Формат файла	*Строковый ресурс Android*

См.также:

Документация по строковым ресурсам Android, Строковые ресурсы Android

---

Примечание: Структуры строкового массива Android, *string-array*, в настоящее время не поддерживаются. Чтобы это обойти, вы можете разбить свои строковые массивы на части:

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

станет:

```
<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>
```

Элемент *string-array*, указывающий на элементы *string*, должен храниться в другом файле и не должен быть доступен для перевода.

Помочь в подготовке существующих файлов strings.xml и переводов вам может вот этот скрипт: https://gist.github.com/paour/11291062

1.10.13 Строки Apple iOS

Специфичный для Apple формат файла для перевода приложений, используемый для переводов приложений как для iOS, так и для iPhone'a/iPad'a.

Строки Apple iOS обычно используются в качестве двуязычных переводов.

Типичная *настройка компонента* в Weblate	
Маска файла	`Resources/*.lproj/Localizable.strings`
Одноязычный базовый файл	`Resources/en.lproj/Localizable.strings`
Шаблон для новых переводов	*Пусто*
Формат файла	*Строки iOS (UTF-8)*

См.также:

Документация Apple по «строковым файлам», Строки Mac OSX

1.10.14 Строки PHP

Переводы PHP обычно одноязычные, поэтому рекомендуется указывать базовый файл со строками на (как это чаще всего бывает) английском языке.

Пример файла:

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

Типичная *настройка компонента* в Weblate	
Маска файла	`lang/*/texts.php`
Одноязычный базовый файл	`lang/en/texts.php`
Шаблон для новых переводов	`lang/en/texts.php`
Формат файла	*Строки PHP*

Строки Laravel PHP

Изменено в версии 4.1.

Файлы локализации Laravel PHP также поддерживают множественные формы:

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

См.также:

PHP, Локализация в Laravel

1.10.15 JSON-файлы

Добавлено в версии 2.0.

Изменено в версии 2.16: Начиная с Weblate версии 2.16 и translate-toolkit версии не ниже 2.2.4 также поддерживаются JSON-файлы с вложенной структурой.

Формат JSON используется в основном для перевода приложений, реализованных на JavaScript.

В настоящее время Weblate поддерживает несколько вариантов переводов в JSON:

- Простые файлы с парами ключ / значение.
- Файлы с вложенными ключами.
- *JSON -файлы i18next*
- *JSON -файлы go-i18n*
- *JSON-файлы WebExtension*
- *Файл ARB*

Переводы JSON обычно одноязычные, поэтому рекомендуется указывать базовый файл со строками на (как это чаще всего бывает) английском языке.

Пример файла:

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

Также поддерживается вложенная структура файлов (условия поддержки смотрите выше), такой файл может выглядеть следующим образом:

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

Предупреждение: В настоящее время Weblate обрабатывает вложенные JSON-файлы, уплощая их ключи. Это приводит к проблемам сериализации, когда в реальных ключах используются такие символы как `.` или `[]`, потому что Weblate думает, что это признак вложенного ключа.

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

Типичная *настройка компонента* в Weblate	
Маска файла	`langs/translation-*.json`
Одноязычный базовый файл	`langs/translation-en.json`
Шаблон для новых переводов	*Пусто*
Формат файла	*Файл вложенной структуры JSON*

См.также:

JSON, *Настроить вывод JSON*, *Очистка файлов перевода*,

1.10.16 JSON -файлы i18next

Изменено в версии 2.17: Начиная с Weblate версии 2.17 и translate-toolkit версии не ниже 2.2.5 также поддерживаются JSON-файлы i18next с множественными формами.

i18next — это фреймворк интернационализации, написанный на и для JavaScript. Weblate поддерживает его файлы локализации с такими возможностями, как формы множественного числа.

Переводы i18next обычно одноязычные, поэтому рекомендуется указывать базовый файл со строками на (как это чаще всего бывает) английском языке.

Примечание: Weblate поддерживает формат JSON-файла i18next версии v3. Версии v2 и v1 в основном совместимы, за исключением того, как они работают с формами множественного числа.

Пример файла:

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

Типичная *настройка компонента* в Weblate	
Маска файла	`langs/*.json`
Одноязычный базовый файл	`langs/en.json`
Шаблон для новых переводов	*Пусто*
Формат файла	*JSON-файл i18next*

См.также:

JSON, i18next JSON Format, *Настроить вывод JSON*, *Очистка файлов перевода*

1.10.17 JSON -файлы go-i18n

Добавлено в версии 4.1.

Переводы go-i18n обычно одноязычные, поэтому рекомендуется указывать базовый файл со строками на (как это чаще всего бывает) английском языке.

Примечание: Weblate поддерживает формат JSON-файла go-i18n версии v2, который не поддерживает плоские JSON-файлы, поддерживаемые версией v1.

Типичная *настройка компонента* в Weblate	
Маска файла	`langs/*.json`
Одноязычный базовый файл	`langs/en.json`
Шаблон для новых переводов	*Пусто*
Формат файла	*JSON -файл go-i18n*

См.также:

JSON, go-i18n, *Настроить вывод JSON*, *Очистка файлов перевода*,

1.10.18 Файл ARB

Добавлено в версии 4.1.

Переводы ARB обычно одноязычные, поэтому рекомендуется указывать базовый файл со строками на (как это чаще всего бывает) английском языке.

Типичная *настройка компонента* в Weblate	
Маска файла	`lib/l10n/intl_*.arb`
Одноязычный базовый файл	`lib/l10n/intl_en.arb`
Шаблон для новых переводов	*Пусто*
Формат файла	*Файл ARB*

См.также:

JSON, Application Resource Bundle Specification, Internationalizing Flutter apps, *Настроить вывод JSON*, *Очистка файлов перевода*

1.10.19 JSON-файлы WebExtension

Добавлено в версии 2.16: Поддерживаются с версии Weblate 2.16 и translate-toolkit версии не ниже 2.2.4.

Формат файла, используемый при переводе расширений для Mozilla Firefox или Google Chromium.

Примечание: Хотя этот формат называется JSON, его спецификация позволяет включать комментарии, которые не являются частью спецификации JSON. В настоящее время Weblate не поддерживает файл с комментариями.

Пример файла:

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

Типичная *настройка компонента* в Weblate	
Маска файла	`_locales/*/messages.json`
Одноязычный базовый файл	`_locales/en/messages.json`
Шаблон для новых переводов	*Пусто*
Формат файла	*JSON-файл WebExtension*

См.также:

JSON, Google chrome.i18n, Интернационализация расширений Mozilla

1.10.20 .XML-файлы ресурсов

Добавлено в версии 2.3.

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

Типичная *настройка компонента* в Weblate	
Маска файла	`Resources/Language.*.resx`
Одноязычный базовый файл	`Resources/Language.resx`
Шаблон для новых переводов	*Пусто*
Формат файла	*.NET resource file*

См.также:

Файлы ресурсов .NET (.resx), *Очистка файлов перевода*

1.10.21 Файлы CSV

Добавлено в версии 2.4.

Файлы CSV могут содержать простой список исходных строк и переводов. Weblate поддерживает следующие файлы:

- Файлы с заголовком, определяющим поля (источник, перевод, местоположение, …). Это рекомендуемый подход, так как он наименее подвержен ошибкам.
- Файлы с двумя полями — источником и переводом (в этом порядке), для этого в качестве формата файла выберите *Простой CSV файл*
- Файлы с полями, определёнными translate-toolkit: location, source, target, ID, fuzzy, context, translator_comments, developer_comments

Предупреждение: Формат CSV в настоящее время автоматически определяет диалект файла CSV. В некоторых случаях автоматическое определение может не сработать, и вы получите смешанные результаты. Особенно это касается файлов CSV с символами новых строк внутри значений. В качестве обходного пути рекомендуется не использовать кавычки.

Пример файла:

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

Типичная *настройка компонента* в Weblate	
Маска файла	`locale/*.csv`
Одноязычный базовый файл	*Пусто*
Шаблон для новых переводов	`locale/en.csv`
Формат файла	*Файл CSV*

См.также:

CSV

1.10.22 Файлы YAML

Добавлено в версии 2.9.

Простые файлы YAML со строковыми ключами и значениями. Также Weblate извлекает строки из списков и словарей.

Пример файла YAML:

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

Типичная *настройка компонента* в Weblate	
Маска файла	`translations/messages.*.yml`
Одноязычный базовый файл	`translations/messages.en.yml`
Шаблон для новых переводов	*Пусто*
Формат файла	*Файл YAML*

См.также:

YAML, *Файлы Ruby YAML*

1.10.23 Файлы Ruby YAML

Добавлено в версии 2.9.

Файлы YAML локализации Ruby с языком в качестве корневого узла.

Пример файла YAML локализации Ruby:

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

Типичная *настройка компонента* в Weblate	
Маска файла	`translations/messages.*.yml`
Одноязычный базовый файл	`translations/messages.en.yml`
Шаблон для новых переводов	*Пусто*
Формат файла	*Файл Ruby YAML*

См.также:

YAML, *Файлы YAML*

1.10.24 Файлы DTD

Добавлено в версии 2.18.

Пример DTD-файла:

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

Типичная *настройка компонента* в Weblate	
Маска файла	`locale/*.dtd`
Одноязычный базовый файл	`locale/en.dtd`
Шаблон для новых переводов	*Пусто*
Формат файла	*Файл DTD*

См.также:

Формат DTD Mozilla

1.10.25 Плоские XML-файлы

Добавлено в версии 3.9.

Пример плоского XML-файла:

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

Типичная *настройка компонента* в Weblate	
Маска файла	`locale/*.xml`
Одноязычный базовый файл	`locale/en.xml`
Шаблон для новых переводов	*Пусто*
Формат файла	*Плоский XML-файл*

См.также:

Плоские XML-файлы

1.10.26 Файлы ресурсов Windows

Изменено в версии 4.1: Поддержка файлов ресурсов Windows была переписана.

Примечание: Поддержка этого формата в настоящее время находится в бета-режиме, приветствуются обратная связь от тестирования.

Пример файла ресурсов Windows:

```
LANGUAGE LANG_CZECH, SUBLANG_DEFAULT

STRINGTABLE
BEGIN
```

(continues on next page)

(продолжение с предыдущей страницы)

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

Типичная *настройка компонента* в Weblate	
Маска файла	`lang/*.rc`
Одноязычный базовый файл	`lang/en-US.rc`
Шаблон для новых переводов	`lang/en-US.rc`
Формат файла	*Файл RC*

См.также:

Файлы ресурсов Windows

1.10.27 Файлы метаданных магазина приложений

Добавлено в версии 3.5.

Метаданные, используемые для публикации приложений в различных магазинах приложений, могут быть переведены. В настоящее время совместимы следующие инструменты:

- Triple-T gradle-play-publisher
- Fastlane
- F-Droid

Метаданные состоят из нескольких текстовых файлов, которые Weblate представляет для перевода в виде отдельных строк.

Типичная *настройка компонента* в Weblate	
Маска файла	`fastlane/android/metadata/*`
Одноязычный базовый файл	`fastlane/android/metadata/en-US`
Шаблон для новых переводов	`fastlane/android/metadata/en-US`
Формат файла	*Файлы метаданных магазина приложений*

1.10.28 Файлы субтитров

Добавлено в версии 3.7.

Weblate может переводить различные файлы субтитров:

- Файл субтитров SubRip (`*.srt`)
- Файл субтитров MicroDVD (`*.sub`)
- Файл субтитров Advanced SubStation Alpha (`*.ass`)
- Файл субтитров SubStation Alpha (`*.ssa`)

Типичная *настройка компонента* в Weblate	
Маска файла	`path/*.srt`
Одноязычный базовый файл	`path/en.srt`
Шаблон для новых переводов	`path/en.srt`
Формат файла	*Файл субтитров SubRip*

См.также:

Субтитры

1.10.29 Excel Open XML

Добавлено в версии 3.2.

Можно импортировать и экспортировать файлы Excel Open XML (.xlsx).

Загружая для перевода файлы XLSX, имейте в виду, что учитывается только активный рабочий лист, и должны быть хотя бы колонки с названием `source` (содержащая исходную строку) и `target` (содержащая перевод). Дополнительно может быть колонка с названием `context` (содержащая путь к контексту строки перевода). Если вы используете скачанный XLSX для экспорта переводов в рабочую книгу Excel, вы уже получите файл в правильном формате.

1.10.30 Файлы HTML

Добавлено в версии 4.1.

Примечание: Поддержка этого формата в настоящее время находится в бета-режиме, приветствуются обратная связь от тестирования.

Переводимое содержимое извлекается из HTML-файлов и предлагается для перевода.

См.также:

HTML

1.10.31 Формат OpenDocument

Добавлено в версии 4.1.

Примечание: Поддержка этого формата в настоящее время находится в бета-режиме, приветствуются обратная связь от тестирования.

Переводимое содержимое извлекается из файлов OpenDocument и предлагается для перевода.

См.также:

Формат OpenDocument

1.10.32 Формат IDML

Добавлено в версии 4.1.

Примечание: Поддержка этого формата в настоящее время находится в бета-режиме, приветствуются обратная связь от тестирования.

Переводимое содержимое извлекается из файлов языка разметки Adobe InDesign и предлагается для перевода.

1.10.33 Другие

Большинство форматов, которые поддерживают сериализацию и которые поддерживает translate-toolkit, могут быть легко поддержаны, но они (пока) не прошли никакого тестирования. В большинстве случаев, чтобы скрыть различия в поведении различных хранилищ translate-toolkit, для Weblate требуется тонкая прослойка.

См.также:

Форматы файлов, относящиеся к переводу

1.10.34 Добавление новых переводов

Изменено в версии 2.18: В версиях до 2.18 поведение добавления новых переводов зависело от формата файла.

Weblate может автоматически начинать новый перевод для всех форматов файлов.

Некоторые форматы ожидают, что они будут начинаться с пустого файла и в него будут включены только переведённые строки (например, *Строковые ресурсы Android*), в то время как другие ожидают, что в файле будут присутствовать все ключи (например, *GNU gettext*). В некоторых ситуациях это на самом деле зависит не от формата, а от фреймворка, который вы используете для работы с файлами перевода (например, с *файлами JSON*).

При указании в *конфигурации компонента* файла *шаблона нового перевода* Weblate будет использовать его для начинания новых переводов. Все существующие переводы при этом будут из файла удалены.

Если поле *Шаблон для новых переводов* на заполнено, а формат файла его требует, создаётся пустой файл, в который добавляются новые строки после их перевода.

Параметр *Стиль кода языка* позволяет вам настраивать код языка, используемый в именах сгенерированных файлов:

Умолчание, основанное на формате файла Зависит от формата файла, для большинства форматов используется стиль POSIX.

Стиль POSIX, использующий подчёркивание как разделитель Обычно используется gettext'ом и связанными с ним инструментами, выдаёт коды языков вида *ru_RU*.

Стиль POSIX, использующий подчёркивание как разделитель и включающий код страны Код языка в стиле POSIX, включающий код страны, даже если в этом нет необходимости (например, *cs_CZ*).

Стиль BCP, использующий в качестве разделителя дефис Обычно используется на веб-платформах, выдаёт коды языков вида *ru-RU*.

Стиль BCP, использующий в качестве разделителя дефис и включающий код страны Код языка в стиле BCP, включающий код страны, даже если в этом нет необходимости (например, *cs-CZ*).

Стиль Android Используется только в приложениях для Android, выдаёт коды языков вида *ru-rRU*.

Стиль Java Используется Java — в основном BCP с устаревшими кодами для китайского языка.

Примечание: Weblate распознаёт любой из этих стилей при разборе файлов перевода, вышеуказанные настройки влияют только на то, в каком стиле создавать новые файлы.

1.10.35 Строки только для чтения

Добавлено в версии 3.10.

Строки из файлов перевода, доступные только для чтения, будут включены, но не могут быть отредактированы в Weblate. Поддержка этой возможности уже встроена в некоторые форматы (*XLIFF* и *Строковые ресурсы Android*), а в других может быть сэмулирована добавлением к строке флага `read-only` — смотрите раздел *Настройка поведения*.

1.11 Интеграция с системой контроля версий

В настоящее время Weblate в качестве систем контроля версий поддерживает *Git* (с расширенной поддержкой *GitHub*, *Gerrit* и *Subversion*) и *Mercurial*.

1.11.1 Доступ к репозиториям

Репозиторий системы контроля версий, который вы хотите использовать, должен быть доступен для Weblate. При использовании общедоступного репозитория вам просто надо указать правильный URL-адрес (например, `https://github.com/WeblateOrg/weblate.git`), но для частных репозиториев или для URL-адресов отправки изменений настройка будет более сложной и требует аутентификации.

Доступ к репозиториям из Hosted Weblate

Для Hosted Weblate существует выделенный пользователь для отправки изменений, зарегистрированный на GitHub, Bitbucket, Codeberg и GitLab (с именем пользователя *weblate*, известный также под именем *Weblate push user*). Вам необходимо добавить этого пользователя в качестве сотрудника и дать ему в вашем репозитории соответствующее разрешение (для клонирования подойдет только чтение, отправки изменений требуется запись). В зависимости от сервиса и настроек вашей организации, это произойдёт или немедленно, или потребует подтверждения со стороны Weblate.

Приглашения на GitHub принимаются автоматически в течение пяти минут, на другие сервисы может потребоваться ручная обработка, поэтому, пожалуйста, наберитесь терпения.

После того, как пользователь *weblate* будет добавлен, вы сможете настроить *Репозиторий исходного кода* и *URL для отправки в репозиторий* по протоколу SSH (например, `git@github.com:WeblateOrg/weblate.git`).

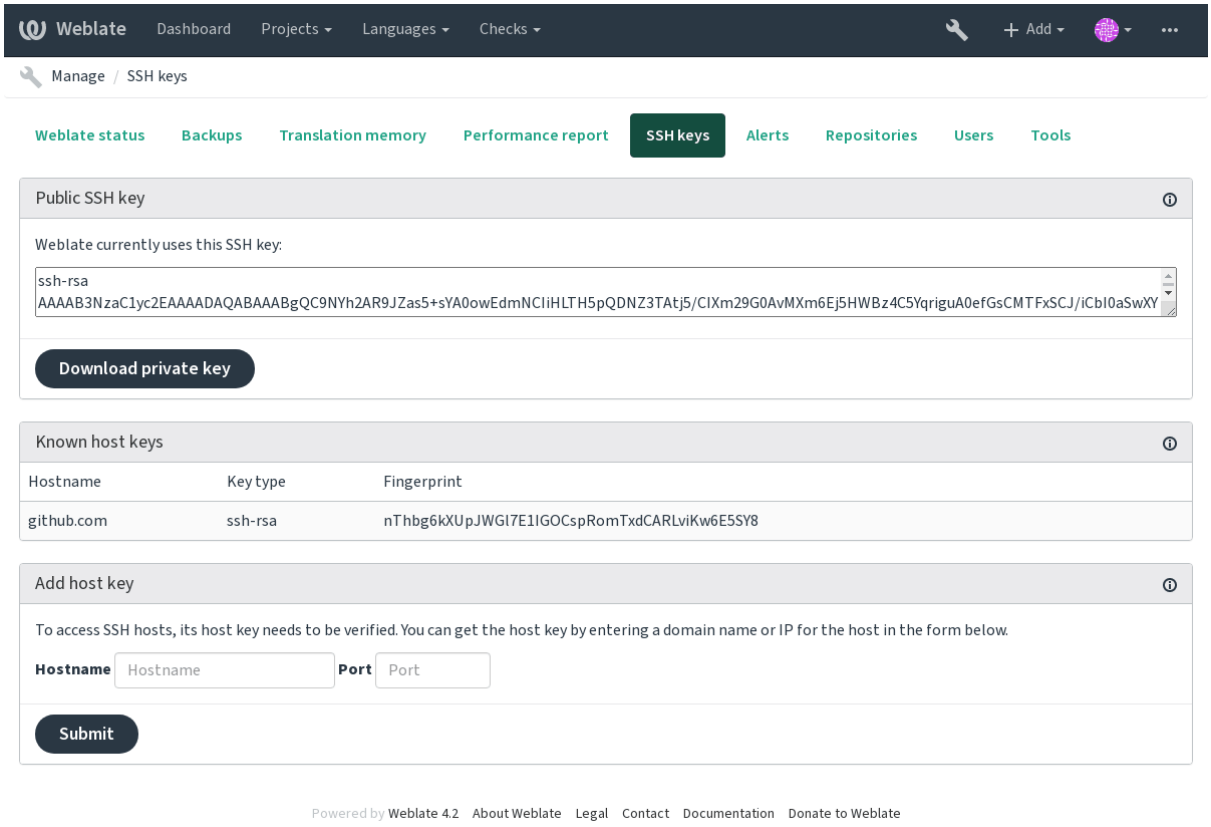
Репозитории по SSH

Наиболее часто используемый метод доступа к частным репозиториям основан на SSH. Чтобы получить таким образом доступ к вышестоящему репозиторию, авторизуйте публичный SSH-ключ Weblate (смотрите раздел *SSH-ключ Weblate*).

> **Предупреждение:** На GitHub каждый ключ может быть добавлен только в один репозиторий, смотрите разделы *Репозитории GitHub* и *Доступ к репозиториям из Hosted Weblate*.

Weblate также сохраняет при первом подключении отпечаток ключа хоста и не сможет подключиться к хосту, если позже ключ будет изменён (смотрите раздел *Проверка SSH-ключей хоста*).

В случае необходимости его изменения это можно сделать из интерфейса администратора Weblate:



SSH-ключ Weblate

Открытый ключ Weblate виден всем пользователям, просматривающим страницу *О Weblate*.

Администраторы могут сгенерировать или посмотреть публичный ключ, который в настоящее время использует Weblate (из блока *SSH-ключи*) на странице входа в интерфейс администратора.

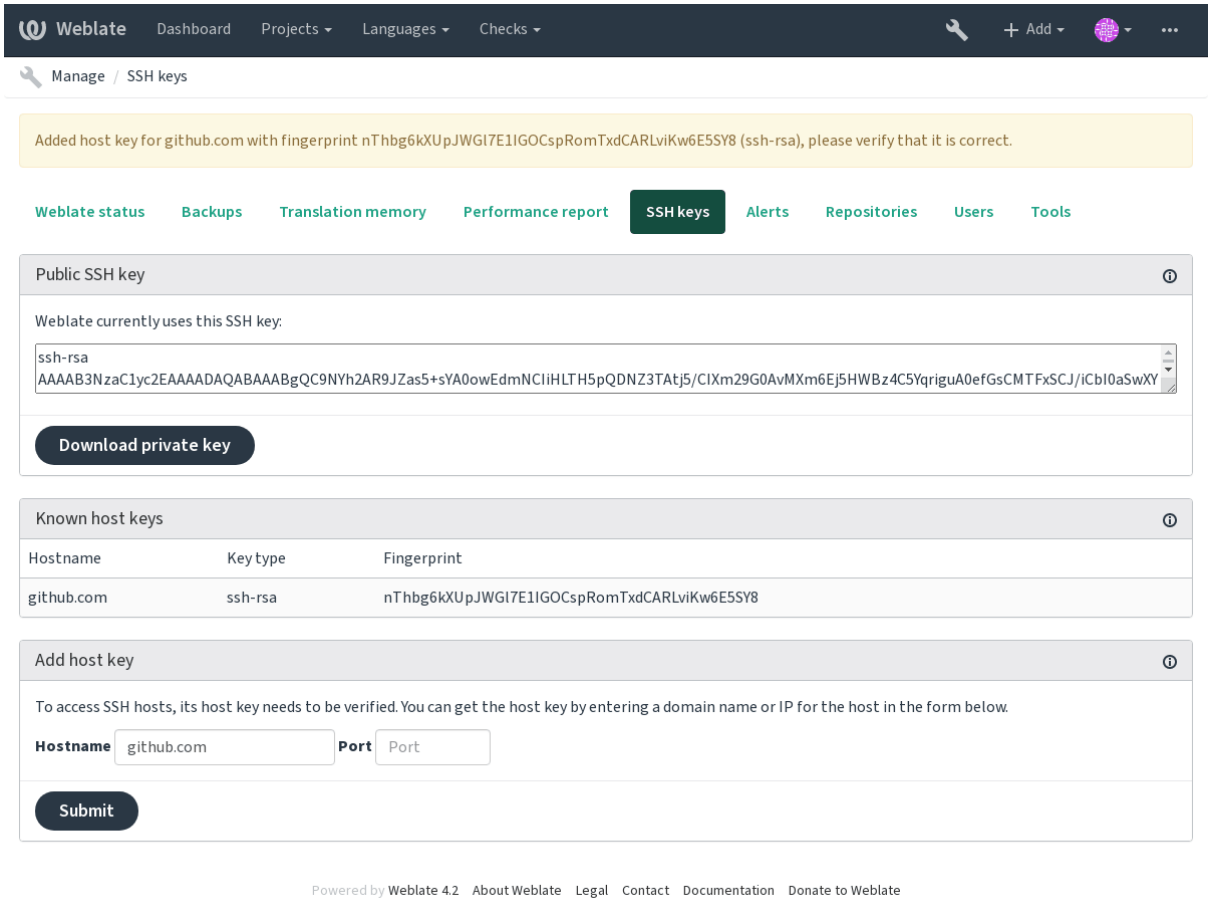
Примечание: Соответствующий закрытый SSH-ключ в настоящее время не может быть запаролен, поэтому убедитесь, что он хорошо защищён.

Подсказка: Сделайте резервную копию сгенерированного закрытого SSH-ключа Weblate.

Проверка SSH-ключей хоста

Weblate автоматически запоминает SSH-ключи хостов при первом к ним обращении и запоминает их для дальнейшего использования.

В случае, если вы хотите проверить их перед подключением к репозиторию, проверьте SSH-ключи хостов серверов, к которым вы собираетесь получить доступ через пункт *Добавить ключ хоста*, в том же разделе административного интерфейса. Введите имя хоста, к которому вы собираетесь получить доступ (например, `gitlab.com`), и нажмите *Подтвердить*. Убедитесь, что его отпечаток совпадает с отпечатком добавленного вами сервера. Они отображаются в сообщении о подтверждении:



Репозитории GitHub

Доступ по SSH возможен (смотрите раздел. *Репозитории по SSH*), но в случае, если вам необходимо получить доступ к более чем одному репозиторию, вы столкнётесь с ограничением GitHub'а по разрешённому использованию ключей SSH (поскольку один ключ может быть использован только для одного репозитория).

В случае, если *ветка для отправки* не задана, проект форкается и изменения отправляются через форк. В случае если она задана, изменения отправляются в вышестоящий репозиторий и выбранную ветку.

Для небольших развёртываний используйте HTTPS-аутентификацию с персональным токеном доступа и вашей GitHub'овской учётной записью, смотрите документ Создание токена доступа для использования из командной строки.

Для более крупных установок обычно лучше создать для Weblate специального пользователя, назначить ему сгенерированный в Weblate открытый SSH ключ (смотрите раздел *SSH-ключ Weblate*) и предоставить ему доступ ко всем репозиториям, которые вы хотите переводить. Этот подход также используется в Hosted Weblate, для этого есть выделенный пользователь *weblate*.

См.также:

Доступ к репозиториям из Hosted Weblate

Внутренние URL-адреса Weblate

Для совместного использования одного репозитория между различными компонентами можно использовать специальный URL-адрес вида `weblate://проект/компонент`. Таким образом, компонент будет совместно использовать конфигурацию репозитория системы контроля версий с соответствующим компонентом (в примере это `проект/компонент`).

Weblate при создании компонента автоматически корректирует URL-адрес репозитория, когда находит компонент с подходящей настройкой репозитория. Вы можете переопределить эту настройку на последнем этапе конфигурации компонента.

Причины для использования такого поведения:

- Экономия дискового пространства на сервере, репозиторий хранится всего один.
- Ускорение обновлений, обновляется только один репозиторий.
- Существует только один экспортируемый репозиторий с переводами Weblate (смотрите раздел *Git exporter*).
- Некоторые надстройки могут работать с несколькими компонентами, совместно использующими один репозиторий, например, надстройка *Склеивать Git-коммиты*.

Репозитории по HTTPS

Для доступа к защищённым HTTPS репозиториям включите в URL-адрес имя пользователя и пароль. Не волнуйтесь, Weblate обрежет эту информацию, когда адрес будет показываться пользователям (даже если им разрешено видеть адрес репозитория).

Например, URL-адрес GitHub с добавленной аутентификацией может выглядеть следующим образом: `https://пользователь:ваш_токен_доступа@github.com/WeblateOrg/weblate.git`.

Примечание: If you username or password contains special characters, those have to be URL encoded, for example `https://user%40example.com:%24password%23@bitbucket.org/`….

Использование прокси

Если вам необходимо получить доступ к репозиториям системы контроля версий по HTTP/HTTPS через прокси-сервер, настройте СКВ на его использование.

Это можно сделать с помощью установки переменных окружения `http_proxy`, `https_proxy` и `all_proxy` (как описано в документации cURL) или принудительно включив его использование в конфигурации системы контроля версий, например:

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

Примечание: Настройка прокси должна выполняться под пользователем, запустившим Weblate (смотрите также раздел *Разрешения файловой системы*) и с `HOME=$DATA_DIR/home` (смотрите *`DATA_DIR`*), иначе Git, запускаемый Weblate, не будет его использовать.

См.также:

Страница руководства по cURL, Документация по команде git config

1.11.2 Git

См.также:

Для получения информации о том, как получить доступ к различным типам репозиториев, смотрите раздел *Доступ к репозиториям*.

Git с принудительной отправкой (force push)

Ведёт себя точно так же, как и сам Git, с той лишь разницей, что он всегда выполняет принудительную отправку. Он предназначен только для случая использования для переводов отдельного репозитория.

> **Предупреждение:** Используйте с осторожностью, поскольку он легко может потерять коммиты в вашем репозитории.

Настройка конфигурации Git'а

Weblate выполняет все команды системы контроля версий с установкой `HOME=$DATA_DIR/home` (смотрите *DATA_DIR*), поэтому изменение конфигурации пользователя должно выполняться в `DATA_DIR/home/.git`.

Удалённые помощники Git

Вы также можете использовать удалённых помощников Git для дополнительной поддержки других систем управления версиями, но будьте готовы к отладке проблем, которые могут за этим последовать.

В настоящее время помощники для Bazaar и Mercurial доступны в отдельных GitHub'овских репозиториях: git-remote-hg и git-remote-bzr. Скачайте их вручную и поместите куда-нибудь в пути поиска (например, в `~/bin`). Убедитесь, что у вас установлены соответствующие системы контроля версий.

После их установки такие удалённые репозитории можно будет использовать при указании репозитория в Weblate.

Для клонирования проекта `gnuhello` из Launchpad с помощью Bazaar используйте:

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

Для клонирования репозитория `hello` с selenic.com через Mercurial используйте:

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

> **Предупреждение:** Неудобство использования удалённых помощников Git'а заключается в, на примере Mercurial, том, что удалённый помощник иногда создаёт новую оконечную фиксацию при обратном проталкивании изменений.

1.11.3 GitHub

Добавлено в версии 2.3.

Он добавляет с помощью инструмента hub тонкий слой логики поверх *Git*, который позволяет отправлять изменения перевода в виде запросов на извлечение, вместо того, чтобы отправлять их непосредственно в репозиторий.

Git отправляет изменения непосредственно в репозиторий, в то время как *GitHub* создаёт запросы на извлечение. Для простого доступа к Git-репозиториям последние не нужны.

См.также:

Pushing changes from Weblate

Отправка изменений в GitHub в виде запросов на извлечение

Если вы не хотите отправлять переводы непосредственно в репозиторий GitHub, их можно отправить в виде одного или нескольких запросов на извлечение.

См.также:

`GITHUB_USERNAME`, *Установка hub* для инструкций по настройке

Установка hub

Отправка изменений в GitHub в виде запросов на извлечение требует на вашем сервере настроенной установки hub. Следуйте инструкциям по установке, расположенным по адресу https://hub.github.com/, используя для завершения конфигурации hub, например:

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

hub спросит у вас ваши учётные данные от GitHub'а, извлечёт токен и сохранит его в файле `~/.config/hub`. Этот файл должен быть доступен для чтения пользователю, запустившему Weblate.

Примечание: Установите имя пользователя, которое вы использовали при настройке *hub* переменной `GITHUB_USERNAME` (переменной окружения `WEBLATE_GITHUB_USERNAME` для образа Docker).

1.11.4 GitLab

Добавлено в версии 3.9.

Просто добавляет с помощью инструмента lab тонкий слой логики поверх *Git*, который позволяет отправлять изменения перевода в виде запросов на слияние вместо того, чтобы отправлять их непосредственно в репозиторий.

Нет необходимости использовать его для доступа к Git-репозиториям, обычный *Git* работает так же, единственная разница заключается в том, как выполняется отправка изменений в репозиторий. С помощью *Git* изменения отправляются непосредственно в репозиторий, в то время как *GitLab* создаёт запрос на слияние.

См.также:

Pushing changes from Weblate

Отправка изменений в GitLab в виде запросов на слияние

Если вы не хотите отправлять переводы непосредственно в репозиторий GitLab, их можно отправить в виде одного или нескольких запросов на слияние.

Настройте инструмент командной строки lab и установите переменную `GITLAB_USERNAME`, чтобы он заработал.

См.также:

`GITLAB_USERNAME`, *Установка Lab* для инструкций по настройке

Установка Lab

Отправка изменений в GitLab в виде запросов на слияние требует на вашем сервере настроенной установки lab. Следуйте инструкциям по установке, расположенным по адресу lab и запустите его без аргументов для завершения конфигурации, например:

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

lab спросит у вас ваш токен доступа к GitLab'у, извлечёт его и сохранит в файле `~/.config/lab.hcl`. Этот файл должен быть доступен для чтения пользователю, запустившему Weblate.

Примечание: Установите имя пользователя, которое вы использовали при настройке *lab* переменной `GITLAB_USERNAME` (переменной окружения `WEBLATE_GITLAB_USERNAME` для образа Docker).

1.11.5 Gerrit

Добавлено в версии 2.2.

Добавляет с помощью инструмента git-review тонкий слой логики поверх *Git*, который позволяет отправлять изменения перевода в виде запросов на рецензирование Gerrit, вместо того, чтобы отправлять их непосредственно в репозиторий.

В документации Gerrit подробно описана конфигурация, необходимая для настройки таких репозиториев.

1.11.6 Mercurial

Добавлено в версии 2.1.

Mercurial — это ещё одна система контроля версий, которую вы можете использовать в Weblate напрямую.

Примечание: Weblate должен работать с любой версией Mercurial'a, но иногда в его интерфейсе командной строки происходят несовместимые изменения, которые нарушают его интеграцию с Weblate.

См.также:

Для получения информации о том, как получить доступ к различным типам репозиториев, смотрите раздел *Доступ к репозиториям*.

1.11.7 Subversion

Добавлено в версии 2.8.

Для взаимодействия с хранилищем subversion Weblate использует команду git-svn. Это Perl-скрипт, который позволяет использовать subversion из клиента Git'a, позволяя пользователям поддерживать полный клон внутреннего хранилища и фиксировать в него локально.

Примечание: Weblate автоматически пытается определить компоновку хранилища Subversion — он поддерживает как прямые URL-адреса для веток, так и хранилища со стандартной компоновкой (каталоги branches/, tags/ и trunk/). Подробнее об этом можно прочитать в документации к git-svn. Если ваше хранилище не придерживается стандартной компоновки и вы столкнулись с ошибками, попробуйте включить в URL-адрес хранилища имя ветки, а саму ветку оставить пустой.

Изменено в версии 2.19: До этого существовала поддержка только стандартных компоновок хранилищ.

Учётные данные Subversion

Weblate ожидает, что вы заранее приняли сертификат и, если необходимо, свои учётные данные. Он будет искать возможность поместить их в каталог DATA_DIR. Примите сертификат, используя команду *svn* один раз с переменной окружения *$HOME*, установленной в каталог DATA_DIR:

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

См.также:

DATA_DIR

1.11.8 Локальные файлы

Добавлено в версии 3.8.

Weblate также может работать без удалённой системы контроля версий. Первоначальные переводы импортируются путём их загрузки. Позже вы можете заменить отдельные файлы через загрузку файлов или добавить строки перевода непосредственно из Weblate (в настоящее время этот функционал доступен только для одноязычных переводов).

В фоне Weblate создаст Git-репозиторий, и будет отслеживать все изменения. В случае, если вы позже решите использовать для хранения переводов систему контроля версий, у вас уже будет репозиторий внутри Weblate, на основе которого вы сможете настроить свою интеграцию.

1.12 Weblate's REST API

Добавлено в версии 2.6: The REST API is available since Weblate 2.6.

API доступен по адресу `/api/` и основан на фреймворке Django REST. Вы можете использовать его напрямую или через *Клиент Weblate*.

1.12.1 Аутентификация и общие параметры

API публичного проекта доступен без аутентификации, хотя количество неаутентифицированных запросов сильно ограничено (по умолчанию до 100 запросов в день), так что аутентификацию всё равно рекомендуется использовать. Аутентификация использует токен, который вы можете получить в своём профиле. Используйте его в заголовке `Authorization`:

ANY /

Описанные здесь общее поведение запросов для API, заголовки, коды состояния и параметры применяются ко всем конечным точкам.

Параметры запроса

- **format** – Формат ответа (переопределяет заголовок Accept). Возможные значения зависят от настройки REST-фреймворка, по умолчанию поддерживаются `json` и `api`. Последний предоставляет для API интерфейс веб-браузера.

Заголовки запроса

- Accept – тип содержимого ответа зависит от заголовка Accept
- Authorization – необязательный токен для аутентификации

Заголовки ответа

- Content-Type – зависит от заголовка Accept запроса
- Allow – список разрешённых для объекта методов HTTP

JSON -объект ответа

- **detail** (*string*) – подробное описание сбоя (для кодов статусов HTTP, отличных от 200 OK)
- **count** (*int*) – общее количество элементов для списков объектов
- **next** (*string*) – URL-адрес следующей страницы для списков объектов
- **previous** (*string*) – URL-адрес предыдущей страницы для списков объектов
- **results** (*array*) – результаты для списков объектов
- **url** (*string*) – URL-адрес для доступа к этому ресурсу с помощью API
- **web_url** (*string*) – URL-адрес для доступа к этому ресурсу через веб-браузер

Коды статусов

- 200 OK – при правильной обработке запроса
- 400 Bad Request – при отсутствии параметров формы
- 403 Forbidden – в случае отказа в доступе
- 429 Too Many Requests – при превышении количества запросов из-за ограничения на количество запросов

Примеры аутентификации

Пример запроса:

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

Пример ответа:

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

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

Пример CURL:

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

Примеры передачи параметров

Для метода POST параметры могут быть указаны либо в виде представления формы (*application/x-www-form-urlencoded*), либо в JSON (*application/json*).

Пример запроса формы:

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

operation=pull
```

Пример JSON запроса:

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

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

Пример CURL:

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

Пример CURL с JSON:

```
curl \
    --data-binary '{"operation":"pull"}' \
    -H "Content-Type: application/json" \
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    -H "Authorization: Token TOKEN" \
    http://example.com/api/components/hello/weblate/repository/
```

Ограничение частоты запросов

Частота запросов к API ограничена; конфигурация по умолчанию ограничивает её до 100 запросов в день для анонимных пользователей и до 5000 запросов в час для аутентифицированных пользователей.

Это ограничение можно настроить в `settings.py`; подробнее о его настройке смотрите в разделе Throttling документации фреймворка Django REST.

О состоянии ограничения частоты запросов сообщается в следующих заголовках:

`X-RateLimit-Limit`	Ограничение на максимальное число выполняемых запросов
`X-RateLimit-Remaining`	Оставшийся лимит запросов
`X-RateLimit-Reset`	Количество секунд до сброса окна ограничения количества запросов

Изменено в версии 4.1: Добавлены заголовки состояния ограничения количества запросов.

1.12.2 Точка входа в API

GET /api/

Корневая точка входа в API.

Пример запроса:

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

Пример ответа:

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

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

1.12.3 Пользователи

Добавлено в версии 4.0.

GET /api/users/

Возвращает список пользователей, если у вас есть разрешения на просмотр пользоватей для управления. Если их нет, то вы увидите только свои данные.

См.также:

Атрибуты объекта пользователя задокументированы в разделе *GET /api/users/(str:username)/*.

POST /api/users/

Создаёт нового пользователя.

Параметры

- **username** (*string*) – Имя пользователя
- **full_name** (*string*) – Полное имя пользователя
- **email** (*string*) – Электронная почта пользователя
- **is_superuser** (*boolean*) – Является ли пользователь суперпользователем? (необязательный)
- **is_active** (*boolean*) – Активен ли пользователь? (необязательный)

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

Возвращает информацию о пользователях.

Параметры

- **username** (*string*) – Имя пользователя

JSON -объект ответа

- **username** (*string*) – имя пользователя
- **full_name** (*string*) – полное имя пользователя
- **email** (*string*) – электронная почта пользователя
- **is_superuser** (*boolean*) – является ли пользователь суперпользователем
- **is_active** (*boolean*) – активен ли пользователь
- **date_joined** (*string*) – дата создания пользователя
- **groups** (*array*) – ссылка на связанные группы; смотрите описание *GET /api/groups/(int:id)/*

Пример данных в JSON:

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

PUT /api/users/(**str:** *username*)**/**
Изменяет параметры пользователя.

Параметры

- **username** (*string*) – Имя пользователя

JSON -объект ответа

- **username** (*string*) – имя пользователя
- **full_name** (*string*) – полное имя пользователя
- **email** (*string*) – электронная почта пользователя
- **is_superuser** (*boolean*) – является ли пользователь суперпользователем
- **is_active** (*boolean*) – активен ли пользователь
- **date_joined** (*string*) – дата создания пользователя

PATCH /api/users/(**str:** *username*)**/**
Изменяет параметры пользователя.

Параметры

- **username** (*string*) – Имя пользователя

JSON -объект ответа

- **username** (*string*) – имя пользователя
- **full_name** (*string*) – полное имя пользователя
- **email** (*string*) – электронная почта пользователя
- **is_superuser** (*boolean*) – является ли пользователь суперпользователем
- **is_active** (*boolean*) – активен ли пользователь
- **date_joined** (*string*) – дата создания пользователя

DELETE /api/users/(**str:** *username*)**/**
Удаляет всю информацию о пользователе и помечает пользователя как неактивного.

Параметры

- **username** (*string*) – Имя пользователя

POST /api/users/(**str:** *username*)**/groups/**
Связывает группы с пользователем.

Параметры

- **username** (*string*) – Имя пользователя

Параметры формы

- **string group_id** – Уникальный идентификатор группы

GET /api/users/(**str:** *username*)**/notifications/**
Список подписок пользователя.

Параметры

- **username** (*string*) – Имя пользователя

POST /api/users/(**str:** *username*)**/notifications/**
Связывает подписки с пользователем.

Параметры

- **username** (*string*) – Имя пользователя

JSON -объект запроса

- **notification** (*string*) – Название зарегистрированного уведомления
- **scope** (*int*) – Область действия уведомления из доступных вариантов
- **frequency** (*int*) – Выбор частоты уведомлений

GET /api/users/(**str:** *username*)**/notifications/**
int: *subscription_id***/** Возвращает связанную с пользователем подписку.

Параметры

- **username** (*string*) – Имя пользователя
- **subscription_id** – Название зарегистрированного уведомления
- **subscription_id** – целое

PUT /api/users/(**str:** *username*)**/notifications/**
int: *subscription_id***/** Редактирует связанную с пользователем подписку.

Параметры

- **username** (*string*) – Имя пользователя
- **subscription_id** – Название зарегистрированного уведомления
- **subscription_id** – целое

JSON -объект запроса

- **notification** (*string*) – Название зарегистрированного уведомления
- **scope** (*int*) – Область действия уведомления из доступных вариантов
- **frequency** (*int*) – Выбор частоты уведомлений

PATCH /api/users/(**str:** *username*)**/notifications/**
int: *subscription_id***/** Редактирует связанную с пользователем подписку.

Параметры

- **username** (*string*) – Имя пользователя
- **subscription_id** – Название зарегистрированного уведомления
- **subscription_id** – целое

JSON -объект запроса

- **notification** (*string*) – Название зарегистрированного уведомления
- **scope** (*int*) – Область действия уведомления из доступных вариантов
- **frequency** (*int*) – Выбор частоты уведомлений

DELETE /api/users/(**str:** *username*)**/notifications/**
int: *subscription_id***/** Удаляет связанную с пользователем подписку.

Параметры

- **username** (*string*) – Имя пользователя
- **subscription_id** – Название зарегистрированного уведомления
- **subscription_id** – целое

1.12.4 Группы

Добавлено в версии 4.0.

GET /api/groups/

Возвращает список групп, если у вас есть разрешения на просмотр групп для управления. Если их нет, то вы можете видеть только те группы, в которые входит пользователь.

См.также:

Атрибуты объекта группы задокументированы в разделе *GET /api/groups/(int:id)/*.

POST /api/groups/

Создаёт новую группу.

Параметры

- **name** (*string*) – Название группы
- **project_selection** (*int*) – Группа проекта, выбранная из предложенных вариантов
- **language_selection** (*int*) – Группа языков, выбранная из предложенных вариантов

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

Возвращает информацию о группе.

Параметры

- **id** (*int*) – Идентификатор группы

JSON -объект ответа

- **name** (*string*) – название группы
- **project_selection** (*int*) – целое число, соответствующее группе проектов
- **language_selection** (*int*) – целое число, соответствующее группе языков
- **roles** (*array*) – ссылка на связанные роли; смотрите описание *GET /api/roles/(int:id)/*
- **projects** (*array*) – ссылка на связанные проекты; смотрите описание *GET /api/projects/(string:project)/*
- **components** (*array*) – ссылка на связанные компоненты; смотрите описание *GET /api/components/(string:project)/(string:component)/*
- **componentlist** (*array*) – ссылка на связанный список компонентов; смотрите описание *GET /api/component-lists/(str:slug)/*

Пример данных в JSON:

```
{
    "name": "Guests",
    "project_selection": 3,
    "language_selection": 1,
    "url": "http://example.com/api/groups/1/",
    "roles": [
        "http://example.com/api/roles/1/",
        "http://example.com/api/roles/2/"
    ],
    "languages": [
        "http://example.com/api/languages/en/",
        "http://example.com/api/languages/cs/",
    ],
    "projects": [
        "http://example.com/api/projects/demo1/",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        "http://example.com/api/projects/demo/"
    ],
    "componentlist": "http://example.com/api/component-lists/new/",
    "components": [
        "http://example.com/api/components/demo/weblate/"
    ]
}
```

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

Изменяет параметры группы.

Параметры

- **id** (*int*) – Идентификатор группы

JSON -объект ответа

- **name** (*string*) – название группы
- **project_selection** (*int*) – целое число, соответствующее группе проектов
- **language_selection** (*int*) – целое число, соответствующее группе языков

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

Изменяет параметры группы.

Параметры

- **id** (*int*) – Идентификатор группы

JSON -объект ответа

- **name** (*string*) – название группы
- **project_selection** (*int*) – целое число, соответствующее группе проектов
- **language_selection** (*int*) – целое число, соответствующее группе языков

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

Удаляет группу.

Параметры

- **id** (*int*) – Идентификатор группы

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

Связывает роли с группой.

Параметры

- **id** (*int*) – Идентификатор группы

Параметры формы

- **string role_id** – Уникальный идентификатор роли

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

Связывает компоненты с группой.

Параметры

- **id** (*int*) – Идентификатор группы

Параметры формы

- **string component_id** – Уникальный идентификатор компонента

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

int: *component_id* Удаляет компонент из группы.

Параметры

- **id** (*int*) – Идентификатор группы
- **component_id** (*int*) – Уникальный идентификатор компонента

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

Связывает проекты с группой.

Параметры

- **id** (*int*) – Идентификатор группы

Параметры формы

- **string project_id** – Уникальный идентификатор проекта

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

int: *project_id* Удаляет проект из группы.

Параметры

- **id** (*int*) – Идентификатор группы
- **project_id** (*int*) – Уникальный идентификатор проекта

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

Связывает языки с группой.

Параметры

- **id** (*int*) – Идентификатор группы

Параметры формы

- **string language_code** – Уникальный код языка

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

string: *language_code* Удаляет язык из группы.

Параметры

- **id** (*int*) – Идентификатор группы
- **language_code** (*string*) – Уникальный код языка

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

Связывает списки компонентов с группой.

Параметры

- **id** (*int*) – Идентификатор группы

Параметры формы

- **string component_list_id** – Уникальный идентификатор списка компонентов

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

int: *component_list_id* Удаляет список компонентов из группы.

Параметры

- **id** (*int*) – Идентификатор группы
- **component_list_id** (*int*) – Уникальный идентификатор списка компонентов

1.12.5 Роли

GET /api/roles/

Возвращает список всех ролей, связанных с пользователем. Если пользователь является суперпользователем, то возвращается список всех существующих ролей.

См.также:

Атрибуты объекта роли задокументированы в разделе *GET /api/roles/(int:id)/*.

POST /api/roles/

Создаёт новую роль.

Параметры

- **name** (*string*) – Название роли
- **permissions** (*array*) – Список кодовых названий разрешений

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

Возвращает информацию о роли.

Параметры

- **id** (*int*) – Идентификатор роли

JSON -объект ответа

- **name** (*string*) – Название роли
- **permissions** (*array*) – список кодовых названий разрешений

Пример данных в JSON:

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

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

Изменяет параметры роли.

Параметры

- **id** (*int*) – Идентификатор роли

JSON -объект ответа

- **name** (*string*) – Название роли
- **permissions** (*array*) – список кодовых названий разрешений

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

Изменяет параметры роли.

Параметры

- **id** (*int*) – Идентификатор роли

JSON -объект ответа

- **name** (*string*) – Название роли
- **permissions** (*array*) – список кодовых названий разрешений

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

Удаляет роль.

Параметры

- **id** (*int*) – Идентификатор роли

1.12.6 Языки

GET /api/languages/

Возвращает список всех языков.

См.также:

Атрибуты объекта языка задокументированы в разделе *GET /api/languages/(string:language)/*.

POST /api/languages/

Создаёт новый язык.

Параметры

- **code** (*string*) – Название языка
- **name** (*string*) – Название языка
- **direction** (*string*) – Направление языка
- **plural** (*object*) – Формула множественного числа языка и количество форм множественных чисел

GET /api/languages/(string: *language*)/

Возвращает информацию о языке.

Параметры

- **language** (*string*) – Код языка

JSON -объект ответа

- **code** (*string*) – Код языка
- **direction** (*string*) – Направление текста
- **plural** (*object*) – Объект информации о формах множественного числа языка
- **aliases** (*array*) – Массив псевдонимов языка

Пример данных в JSON:

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

(continues on next page)

(продолжение с предыдущей страницы)

```
    "statistics_url": "http://example.com/api/languages/en/statistics/"
}
```

PUT /api/languages/(string: *language*)/

Изменяет параметры языка.

Параметры

- **language** (*string*) – Код языка

JSON -объект запроса

- **name** (*string*) – Название языка
- **direction** (*string*) – Направление языка
- **plural** (*object*) – Данные по формам множественного числа языка

PATCH /api/languages/(string: *language*)/

Изменяет параметры языка.

Параметры

- **language** (*string*) – Код языка

JSON -объект запроса

- **name** (*string*) – Название языка
- **direction** (*string*) – Направление языка
- **plural** (*object*) – Данные по формам множественного числа языка

DELETE /api/languages/(string: *language*)/

Удаляет язык.

Параметры

- **language** (*string*) – Код языка

GET /api/languages/(string: *language*)/statistics/

Возвращает статистику по языку.

Параметры

- **language** (*string*) – Код языка

JSON -объект ответа

- **total** (*int*) – общее количество строк
- **total_words** (*int*) – общее количество слов
- **last_change** (*timestamp*) – последние изменения в языке
- **recent_changes** (*int*) – общее количество изменений
- **translated** (*int*) – количество переведённых строк
- **translated_percent** (*float*) – процент переведённых строк
- **translated_words** (*int*) – количество переведённых слов
- **translated_words_percent** (*int*) – процент переведённых слов
- **translated_chars** (*int*) – количество переведённых символов
- **translated_chars_percent** (*int*) – процент переведённых символов
- **total_chars** (*int*) – общее количество символов
- **fuzzy** (*int*) – количество неточных переводов

- **fuzzy_percent** (*int*) – процент неточных переводов
- **failing** (*int*) – количество строк с неудачными проверками
- **failing** – процент строк с неудачными проверками

1.12.7 Проекты

GET /api/projects/

Возвращает список всех проектов.

См.также:

Атрибуты объекта проекта задокументированы в разделе *GET /api/projects/(string:project)/*.

POST /api/projects/

Добавлено в версии 3.9.

Создаёт новый проект.

Параметры

- **name** (*string*) – Название проекта
- **slug** (*string*) – Плашка проекта
- **web** (*string*) – Веб-сайт проекта
- **source_language** (*string*) – Код исходного языка проекта (необязательный)

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

Возвращает информацию о проекте.

Параметры

- **project** (*string*) – URL-плашка проекта

JSON -объект ответа

- **name** (*string*) – название проекта
- **slug** (*string*) – плашка проекта
- **source_language** (*object*) – объект исходного языка; смотрите описание *GET /api/languages/(string:language)/*
- **web** (*string*) – веб-сайт проекта
- **components_list_url** (*string*) – URL-адрес списка компонентов; смотрите описание *GET /api/projects/(string:project)/components/*
- **repository_url** (*string*) – URL-адрес статуса репозитория; смотрите описание *GET /api/projects/(string:project)/repository/*
- **changes_list_url** (*string*) – URL-адрес списка изменений; смотрите описание *GET /api/projects/(string:project)/changes/*

Пример данных в JSON:

```
{
    "name": "Hello",
    "slug": "hello",
    "source_language": {
        "code": "en",
        "direction": "ltr",
        "name": "English",
        "url": "http://example.com/api/languages/en/",
        "web_url": "http://example.com/languages/en/"
```

(continues on next page)

(продолжение с предыдущей страницы)

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

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

Добавлено в версии 3.9.

Удаляет проект.

Параметры

- **project** (*string*) – URL-плашка проекта

GET /api/projects/(string: *project*)/changes/

Возвращает список изменений проекта. По сути, эта точка входа работает так же и принимает те же самые параметры, что и *GET /api/changes/*, только для проекта.

Параметры

- **project** (*string*) – URL-плашка проекта

JSON -объект ответа

- **results** (*array*) – массив объектов компонентов; смотрите описание *GET /api/changes/(int:id)/*

GET /api/projects/(string: *project*)/repository/

Возвращает информацию о состоянии репозитория системы контроля версий. Эта конечная точка содержит только общую сводку по всем репозиториям проекта. Для получения более подробной информации о состоянии используйте конечную точку *GET /api/components/(string:project)/(string:component)/repository/*.

Параметры

- **project** (*string*) – URL-плашка проекта

JSON -объект ответа

- **needs_commit** (*boolean*) – есть ли какие-нибудь отложенные изменения для коммита
- **needs_merge** (*boolean*) – есть ли какие-нибудь изменения в вышестоящем репозитории для объединения
- **needs_push** (*boolean*) – есть ли какие-нибудь локальные изменения для отправки

Пример данных в JSON:

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

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

Выполняет заданную операцию на репозитории системы контроля версий.

Параметры

- **project** (*string*) – URL-плашка проекта

JSON -объект запроса

- **operation** (*string*) – Выполняемые операции: одна из `push`, `pull`, `commit`, `reset` или `cleanup`

JSON -объект ответа

- **result** (*boolean*) – результат операции

Пример CURL:

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

Пример JSON запроса:

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

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

Пример ответа в JSON:

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

{"result":true}
```

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

Возвращает список компонентов перевода в указанном проекте.

Параметры

- **project** (*string*) – URL-плашка проекта

JSON -объект ответа

- **results** (*array*) – массив объектов компонентов; смотрите описание *GET /api/components/(string:project)/(string:component)/*

POST /api/projects/(string: *project*)/components/

Добавлено в версии 3.9.

Создаёт компоненты перевода в указанном проекте.

Параметры

- **project** (*string*) – URL-плашка проекта

JSON -объект ответа

- **result** (*object*) – Созданный объект компонента; смотрите описание *GET /api/components/(string:project)/(string:component)/*

Пример CURL:

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

(continues on next page)

(продолжение с предыдущей страницы)

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

Пример JSON запроса:

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

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

Пример ответа в JSON:

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

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

(continues on next page)

(продолжение с предыдущей страницы)

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

GET /api/projects/(string: *project*)/languages/

Возвращает разбитую на страницы статистику по всем языкам проекта.

Добавлено в версии 3.8.

Параметры

- **project** (*string*) – URL-плашка проекта

JSON -объект ответа

- **results** (*array*) – массив объектов статистики перевода
- **language** (*string*) – название языка
- **code** (*string*) – код языка
- **total** (*int*) – общее количество строк
- **translated** (*int*) – количество переведённых строк
- **translated_percent** (*float*) – процент переведённых строк
- **total_words** (*int*) – общее количество слов
- **translated_words** (*int*) – количество переведённых слов
- **words_percent** (*float*) – процент переведённых слов

GET /api/projects/(string: *project*)/statistics/

Возвращает статистику по проекту.

Добавлено в версии 3.8.

Параметры

- **project** (*string*) – URL-плашка проекта

JSON -объект ответа

- **total** (*int*) – общее количество строк
- **translated** (*int*) – количество переведённых строк
- **translated_percent** (*float*) – процент переведённых строк
- **total_words** (*int*) – общее количество слов
- **translated_words** (*int*) – количество переведённых слов

- **words_percent** (*float*) – процент переведённых слов

1.12.8 Компоненты

GET /api/components/

Возвращает список компонентов перевода.

См.также:

Атрибуты объекта компонента задокументированы в разделе *GET /api/components/(string:project)/(string:component)/*.

GET /api/components/(string: *project*)**/**
string: *component***/** Возвращает информацию о компоненте перевода.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

JSON -объект ответа

- **project** (*object*) – проект перевода; смотрите описание *GET /api/projects/(string:project)/*
- **name** (*string*) – *Название компонента*
- **slug** (*string*) – *Плашка компонента*
- **vcs** (*string*) – *Система контроля версий*
- **repo** (*string*) – *Репозиторий исходного кода*
- **git_export** (*string*) – *URL экспортированного репозитория*
- **branch** (*string*) – *Ветка репозитория*
- **push_branch** (*string*) – *Ветка для отправки*
- **filemask** (*string*) – *Маска файла*
- **template** (*string*) – *Одноязычный базовый файл*
- **edit_template** (*string*) – *Правка базового файла*
- **intermediate** (*string*) – *Файл промежуточного языка*
- **new_base** (*string*) – *Шаблон для новых переводов*
- **file_format** (*string*) – *Формат файла*
- **license** (*string*) – *Лицензия перевода*
- **agreement** (*string*) – *Соглашение участника*
- **new_lang** (*string*) – *Добавление нового перевода*
- **language_code_style** (*string*) – *Стиль кода языка*
- **push** (*string*) – *URL для отправки в репозиторий*
- **check_flags** (*string*) – *Флаги перевода*
- **priority** (*string*) – *Приоритет*
- **enforced_checks** (*string*) – *Принудительные проверки*
- **restricted** (*string*) – *Restricted access*
- **repoweb** (*string*) – *Просмотрщик репозитория*
- **report_source_bugs** (*string*) – *Source string bug report address*

- **merge_style** (*string*) – *Стиль слияния*
- **commit_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **add_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **delete_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **merge_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **addon_message** (*string*) – *Commit, add, delete, merge and addon messages*
- **allow_translation_propagation** (*string*) – *Разрешить распространение перевода*
- **enable_suggestions** (*string*) – *Включить предложения*
- **suggestion_voting** (*string*) – *Голосование по предложению*
- **suggestion_autoaccept** (*string*) – *Автоматическое принятие предложений*
- **push_on_commit** (*string*) – *Отправлять при коммите*
- **commit_pending_age** (*string*) – *Возраст изменений для коммита*
- **auto_lock_error** (*string*) – *Блокировать при ошибке*
- **language_regex** (*string*) – *Языковой фильтр*
- **variant_regex** (*string*) – *Регулярное выражение вариантов*
- **repository_url** (*string*) – URL-адрес статуса репозитория; смотрите описание `GET /api/components/(string:project)/(string:component)/repository/`
- **translations_url** (*string*) – URL-адрес списка переводов; смотрите описание `GET /api/components/(string:project)/(string:component)/translations/`
- **lock_url** (*string*) – URL-адрес статуса блокировки; смотрите описание `GET /api/components/(string:project)/(string:component)/lock/`
- **changes_list_url** (*string*) – URL-адрес списка изменений; смотрите описание `GET /api/components/(string:project)/(string:component)/changes/`

Пример данных в JSON:

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

(continues on next page)

(продолжение с предыдущей страницы)

```
        "web_url": "http://example.com/projects/hello/"
    },
    "repo": "file:///home/nijel/work/weblate-hello",
    "template": "",
    "new_base": "",
    "url": "http://example.com/api/components/hello/weblate/",
    "vcs": "git",
    "web_url": "http://example.com/projects/hello/weblate/"
}
```

PATCH /api/components/(string: *project*)/
string: *component*/ Редактирует компонент запросом на исправление.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

JSON -объект запроса

- **name** (*string*) – название компонента
- **slug** (*string*) – плашка компонента
- **repo** (*string*) – URL-адрес репозитория системы контроля версий

Пример CURL:

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

Пример JSON запроса:

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

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

Пример ответа в JSON:

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

{
    "branch": "master",
    "file_format": "po",
    "filemask": "po/*.po",
    "git_export": "",
```

(continues on next page)

(продолжение с предыдущей страницы)

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

PUT /api/components/(string: *project*)/(string: *component*)/ Редактирует компонент запросом на замену.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

JSON -объект запроса

- **branch** (*string*) – Ветка репозитория системы контроля версий
- **file_format** (*string*) – формат файла переводов
- **filemask** (*string*) – маска файлов перевода в репозитории
- **name** (*string*) – название компонента
- **slug** (*string*) – плашка компонента
- **repo** (*string*) – URL-адрес репозитория системы контроля версий
- **template** (*string*) – базовый файл для одноязычных переводов
- **new_base** (*string*) – базовый файл для добавления новых переводов
- **vcs** (*string*) – система контроля версий

DELETE /api/components/(string: *project*)/(string: *component*)/ Добавлено в версии 3.9.

Удаляет компонент.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

GET /api/components/(string: *project*)/
string: *component*/changes/ Возвращает список изменений компонента. По сути, эта точка входа работает так же и принимает те же самые параметры, что и *GET /api/changes/*, только для компонента.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

JSON -объект ответа

- **results** (*array*) – массив объектов компонентов; смотрите описание *GET /api/changes/(int:id)/*

GET /api/components/(string: *project*)/
string: *component*/screenshots/ Возвращает список снимков экрана компонента.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

JSON -объект ответа

- **results** (*array*) – массив снимков экрана компонента; смотрите описание *GET /api/screenshots/(int:id)/*

GET /api/components/(string: *project*)/
string: *component*/lock/ Возвращает состояние блокировки компонента.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

JSON -объект ответа

- **locked** (*boolean*) – заблокирован ли компонент для обновлений

Пример данных в JSON:

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

POST /api/components/(string: *project*)/
string: *component*/lock/ Устанавливает статус блокировки компонента.

Ответ такой же, как и у *GET /api/components/(string:project)/(string:component)/lock/*.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

JSON -объект запроса

- **lock** – Логическое значение, стоит ли блокировка или нет.

Пример CURL:

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

Пример JSON запроса:

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

{"lock": true}
```

Пример ответа в JSON:

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

{"locked":true}
```

GET /api/components/(string: *project*)/(string: *component*)/repository/ Возвращает информацию о состоянии репозитория системы контроля версий.

Ответ такой же, как и у *GET /api/projects/(string:project)/repository/*.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

JSON -объект ответа

- **needs_commit** (*boolean*) – есть ли какие-нибудь отложенные изменения для коммита
- **needs_merge** (*boolean*) – есть ли какие-нибудь изменения в вышестоящем репозитории для объединения
- **needs_push** (*boolean*) – есть ли какие-нибудь локальные изменения для отправки
- **remote_commit** (*string*) – Информация о коммите из удалённого репозитория
- **status** (*string*) – Статус репозитория системы контроля версий по её же данным
- **merge_failure** – Текст, описывающий причину неудачи объединения или `null`, если её нет

POST /api/components/(string: *project*)/(string: *component*)/repository/ Выполняет заданную операцию на репозитории системы контроля версий.

Документацию смотрите в описании *POST /api/projects/(string:project)/repository/*.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

JSON -объект запроса

- **operation** (*string*) – Выполняемые операции: одна из `push`, `pull`, `commit`, `reset` или `cleanup`

JSON -объект ответа

- **result** (*boolean*) – результат операции

Пример CURL:

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

Пример JSON запроса:

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

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

Пример ответа в JSON:

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

{"result":true}
```

GET /api/components/(string: *project*)/(string: *component*)/monolingual_base/ Скачивает базовый файл для одноязычных переводов.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

GET /api/components/(string: *project*)/(string: *component*)/new_template/ Скачивает файл шаблона для новых переводов.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

GET /api/components/(string: *project*)/(string: *component*)/translations/ Возвращает список объектов перевода в указанном компоненте.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

JSON -объект ответа

- **results** (*array*) – массив объектов перевода; смотрите описание *GET /api/translations/(string:project)/(string:component)/(string:language)/*

POST /api/components/(string: *project*)/string: *component*/translations/ Создаёт новый перевод в указанном компоненте.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

JSON -объект запроса

- **language_code** (*string*) – код языка перевода; смотрите описание *GET /api/languages/(string:language)/*

JSON -объект ответа

- **result** (*object*) – новый созданный объект перевода

Пример CURL:

```
curl \
    -d language_code=cs \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/projects/hello/components/
```

Пример JSON запроса:

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

{"language_code": "cs"}
```

Пример ответа в JSON:

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

{
    "failing_checks": 0,
    "failing_checks_percent": 0,
    "failing_checks_words": 0,
    "filename": "po/cs.po",
    "fuzzy": 0,
    "fuzzy_percent": 0.0,
    "fuzzy_words": 0,
    "have_comment": 0,
    "have_suggestion": 0,
    "is_template": false,
    "is_source": false,
    "language": {
        "code": "cs",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        "direction": "ltr",
        "name": "Czech",
        "url": "http://example.com/api/languages/cs/",
        "web_url": "http://example.com/languages/cs/"
    },
    "language_code": "cs",
    "id": 125,
    "last_author": null,
    "last_change": null,
    "share_url": "http://example.com/engage/hello/cs/",
    "total": 4,
    "total_words": 15,
    "translate_url": "http://example.com/translate/hello/weblate/cs/",
    "translated": 0,
    "translated_percent": 0.0,
    "translated_words": 0,
    "url": "http://example.com/api/translations/hello/weblate/cs/",
    "web_url": "http://example.com/projects/hello/weblate/cs/"
}
```

GET /api/components/(string: *project*)/
string: *component*/statistics/ Возвращает разбитую на страницы статистику по всем переводам компонента.

Добавлено в версии 2.7.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента

JSON -объект ответа

- **results** (*array*) – массив объектов статистики перевода; смотрите описание *GET /api/translations/(string:project)/(string:component)/(string:language)/statistics/*

1.12.9 Переводы

GET /api/translations/
Возвращает список переводов.

См.также:

Атрибуты объекта перевода задокументированы в разделе *GET /api/translations/(string:project)/(string:component)/(string:language)/*.

GET /api/translations/(string: *project*)/
string: *component*/string: *language*/ Возвращает информацию о переводе.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента
- **language** (*string*) – Код языка перевода

JSON -объект ответа

- **component** (*object*) – объект компонента; смотрите описание *GET /api/components/(string:project)/(string:component)/*
- **failing_checks** (*int*) – количество строк с неудачными проверками

- **failing_checks_percent** (*float*) – процент строк с неудачными проверками
- **failing_checks_words** (*int*) – количество слов с неудачными проверками
- **filename** (*string*) – имя файла перевода
- **fuzzy** (*int*) – количество строк, помеченных на правку
- **fuzzy_percent** (*float*) – процент строк, помеченных на правку
- **fuzzy_words** (*int*) – количество слов, помеченных на правку
- **have_comment** (*int*) – количество строк с комментариями
- **have_suggestion** (*int*) – количество строк с комментариями
- **is_template** (*boolean*) – использует ли перевод одноязычные файлы
- **language** (*object*) – объект исходного языка; смотрите описание *GET /api/languages/(string:language)/*
- **language_code** (*string*) – код языка, используемый в репозитории; он может отличаться от кода языка в объекте языка
- **last_author** (*string*) – имя последнего автора
- **last_change** (*timestamp*) – метка времени последнего изменения
- **revision** (*string*) – хэш ревизии файла
- **share_url** (*string*) – URL-адрес для принятия участия в переводе, ведущий на вводную страницу
- **total** (*int*) – общее количество строк
- **total_words** (*int*) – общее количество слов
- **translate_url** (*string*) – URL-адрес для перевода
- **translated** (*int*) – количество переведённых строк
- **translated_percent** (*float*) – процент переведённых строк
- **translated_words** (*int*) – количество переведённых слов
- **repository_url** (*string*) – URL-адрес статуса репозитория; смотрите описание *GET /api/translations/(string:project)/(string:component)/(string:language)/repository/*
- **file_url** (*string*) – URL-адрес объекта файла; смотрите описание *GET /api/translations/(string:project)/(string:component)/(string:language)/file/*
- **changes_list_url** (*string*) – URL-адрес списка изменений; смотрите описание *GET /api/translations/(string:project)/(string:component)/(string:language)/changes/*
- **units_list_url** (*string*) – URL-адрес списка строк; смотрите описание *GET /api/translations/(string:project)/(string:component)/(string:language)/units/*

Пример данных в JSON:

```
{
    "component": {
        "branch": "master",
        "file_format": "po",
        "filemask": "po/*.po",
        "git_export": "",
```

(continues on next page)

(продолжение с предыдущей страницы)


```
        "license": "",
        "license_url": "",
        "name": "Weblate",
        "new_base": "",
        "project": {
            "name": "Hello",
            "slug": "hello",
            "source_language": {
                "code": "en",
                "direction": "ltr",
                "name": "English",
                "url": "http://example.com/api/languages/en/",
                "web_url": "http://example.com/languages/en/"
            },
            "url": "http://example.com/api/projects/hello/",
            "web": "https://weblate.org/",
            "web_url": "http://example.com/projects/hello/"
        },
        "repo": "file:///home/nijel/work/weblate-hello",
        "slug": "weblate",
        "template": "",
        "url": "http://example.com/api/components/hello/weblate/",
        "vcs": "git",
        "web_url": "http://example.com/projects/hello/weblate/"
    },
    "failing_checks": 3,
    "failing_checks_percent": 75.0,
    "failing_checks_words": 11,
    "filename": "po/cs.po",
    "fuzzy": 0,
    "fuzzy_percent": 0.0,
    "fuzzy_words": 0,
    "have_comment": 0,
    "have_suggestion": 0,
    "is_template": false,
    "language": {
        "code": "cs",
        "direction": "ltr",
        "name": "Czech",
        "url": "http://example.com/api/languages/cs/",
        "web_url": "http://example.com/languages/cs/"
    },
    "language_code": "cs",
    "last_author": "Weblate Admin",
    "last_change": "2016-03-07T10:20:05.499",
    "revision": "7ddfafe6daaf57fc8654cc852ea6be212b015792",
    "share_url": "http://example.com/engage/hello/cs/",
    "total": 4,
    "total_words": 15,
    "translate_url": "http://example.com/translate/hello/weblate/cs/",
    "translated": 4,
    "translated_percent": 100.0,
    "translated_words": 15,
    "url": "http://example.com/api/translations/hello/weblate/cs/",
    "web_url": "http://example.com/projects/hello/weblate/cs/"
}
```


DELETE /api/translations/(string: *project*)/
string: *component*/string: *language*/ Добавлено в версии 3.9.

Удаляет перевод.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента
- **language** (*string*) – Код языка перевода

GET /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/changes/ Returns a list of translation changes. This is essentially a translations-scoped *GET /api/changes/* accepting the same parameters.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента
- **language** (*string*) – Код языка перевода

JSON -объект ответа

- **results** (*array*) – массив объектов компонентов; смотрите описание *GET /api/changes/(int:id)/*

GET /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/units/ Возвращает список единиц перевода.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента
- **language** (*string*) – Код языка перевода

JSON -объект ответа

- **results** (*array*) – массив объектов компонента; смотрите описание *GET /api/units/(int:id)/*

POST /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/units/ Добавляет новую одноязычную единицу перевода.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента
- **language** (*string*) – Код языка перевода

JSON -объект запроса

- **key** (*string*) – Название единицы перевода
- **value** (*string*) – Значение единицы перевода

POST /api/translations/(string: *project*)/(string: *component*)/(string: *language*)/autotranslate/ Запускает автоматический перевод.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента
- **language** (*string*) – Код языка перевода

JSON -объект запроса

- **mode** (*string*) – Режим автоматического перевода
- **filter_type** (*string*) – Тип фильтра автоматического перевода

- **auto_source** (*string*) – Источник автоматического перевода
- **component** (*string*) – Добавлять переводы этого компонента в общую память переводов проекта, дабы получить к ним доступ из других компонентов.
- **engines** (*string*) – Механизмы машинного перевода
- **threshold** (*string*) – Порог оценки

GET /api/translations/(string: *project*)/
string: *component*/string: *language*/file/ Скачивает текущий файл перевода в том виде, в котором он сохранён в системе контроля версий (без параметра `format`) или преобразованный в стандартный формат (в настоящее время поддерживаются: Gettext PO, MO, XLIFF и TBX).

Примечание: Эта конечная точка API использует отличающуюся от остального API логику вывода, так как она работает не с данными, а с целым файлом. Отличается логика при установке допустимого параметра `format` — без него вы получите файл перевода, хранящийся в системе контроля версий.

Параметры запроса

- **format** – Используемый формат файла; если не указан, преобразование формата не происходит; поддерживаемые форматы файлов: `po`, `mo`, `xliff`, `xliff11`, `tbx`, `csv`, `xlsx`, `json`, `aresource` и `strings`

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента
- **language** (*string*) – Код языка перевода

POST /api/translations/(string: *project*)/
string: *component*/string: *language*/file/ Загружает новый файл с переводами.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента
- **language** (*string*) – Код языка перевода

Параметры формы

- **string conflicts** – Как поступать с конфликтами (`ignore`, `replace-translated` или `replace-approved`)
- **file file** – Загруженный файл
- **string email** – E-mail автора
- **string author** – Имя автора
- **string method** – Метод загрузки (`translate`, `approve`, `suggest`, `fuzzy`, `replace` или `source`), смотрите раздел *Способы импортирования*
- **string fuzzy** – Обработка строк «на правку» (*пусто*, `process` или `approve`)

Пример CURL:

```
curl -X POST \
    -F file=@strings.xml \
    -H "Authorization: Token TOKEN" \
    http://example.com/api/translations/hello/android/cs/file/
```

GET /api/translations/(string: *project*)/ string: *component*/string: *language*/repository/ Возвращает информацию о состоянии репозитория системы контроля версий.

Ответ такой же, как и у *GET /api/components/(string:project)/(string:component)/repository/*.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента
- **language** (*string*) – Код языка перевода

POST /api/translations/(string: *project*)/ string: *component*/string: *language*/repository/ Выполняет заданную операцию на репозитории системы контроля версий.

Документацию смотрите в описании *POST /api/projects/(string:project)/repository/*.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента
- **language** (*string*) – Код языка перевода

JSON -объект запроса

- **operation** (*string*) – Выполняемые операции: одна из `push`, `pull`, `commit`, `reset` или `cleanup`

JSON -объект ответа

- **result** (*boolean*) – результат операции

GET /api/translations/(string: *project*)/ string: *component*/string: *language*/statistics/ Возвращает подробную статистику по переводам.

Добавлено в версии 2.7.

Параметры

- **project** (*string*) – URL-плашка проекта
- **component** (*string*) – URL-плашка компонента
- **language** (*string*) – Код языка перевода

JSON -объект ответа

- **code** (*string*) – код языка
- **failing** (*int*) – количество неудачных проверок
- **failing_percent** (*float*) – процент неудачных проверок
- **fuzzy** (*int*) – количество строк, нуждающихся в пересмотре
- **fuzzy_percent** (*float*) – процент строк, нуждающихся в пересмотре
- **total_words** (*int*) – общее количество слов
- **translated_words** (*int*) – количество переведённых слов
- **last_author** (*string*) – имя последнего автора
- **last_change** (*timestamp*) – дата последнего изменения
- **name** (*string*) – название языка

- **total** (*int*) – общее количество строк
- **translated** (*int*) – количество переведённых строк
- **translated_percent** (*float*) – процент переведённых строк
- **url** (*string*) – URL-адрес для доступа к переводу (URL-адрес вводной страницы)
- **url_translate** (*string*) – URL-адрес для доступа к переводу (реальный URL-адрес перевода)

1.12.10 Единицы перевода

Добавлено в версии 2.10.

GET /api/units/

Возвращает список единиц перевода.

См.также:

Атрибуты объекта единицы перевода задокументированы в разделе *GET /api/units/(int:id)/*.

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

Возвращает информацию о единице перевода.

Параметры

- **id** (*int*) – Идентификатор единицы перевода

JSON -объект ответа

- **translation** (*string*) – URL-адрес связанного объекта перевода
- **source** (*string*) – исходная строка
- **previous_source** (*string*) – предыдущая исходная строка, использованная для поиска нечёткого соответствия
- **target** (*string*) – целевая строка
- **id_hash** (*string*) – уникальный идентификатор единицы перевода
- **content_hash** (*string*) – уникальный идентификатор исходной строки
- **location** (*string*) – местоположение единицы перевода в исходном коде
- **context** (*string*) – контекст единицы перевода
- **note** (*string*) – примечания к единице перевода
- **flags** (*string*) – флаги единицы перевода
- **fuzzy** (*boolean*) – является ли перевод единицы перевода неточным или отмеченным на правку
- **translated** (*boolean*) – переведена ли единица перевода
- **approved** (*boolean*) – одобрена ли единица перевода
- **position** (*int*) – позиция единицы перевода в файле перевода
- **has_suggestion** (*boolean*) – содержит ли единица перевода предложения
- **has_comment** (*boolean*) – содержит ли единица перевода комментарии
- **has_failing_check** (*boolean*) – имеются ли у единицы перевода неудачные проверки
- **num_words** (*int*) – количество слов в исходной строке
- **priority** (*int*) – приоритет перевода; по умолчанию равен 100
- **id** (*int*) – идентификатор единицы перевода

- **web_url** (*string*) – URL-адрес, по которому можно отредактировать единицу перевода
- **souce_info** (*string*) – Ссылка на информацию по исходной строке; смотрите описание *GET /api/units/(int:id)/*

1.12.11 Изменения

Добавлено в версии 2.10.

GET /api/changes/

Изменено в версии 4.1: Фильтрация изменений была введена в выпуске 4.1.

Возвращает список изменений в переводе.

См.также:

Атрибуты объекта изменения задокументированы в разделе *GET /api/changes/(int:id)/*.

Параметры запроса

- **user** (*string*) – Имя пользователя для фильтров
- **action** (*int*) – Действие для фильтра, может быть указано несколько раз
- **timestamp_after** (*timestamp*) – Временная метка в формате ISO 8601, изменения после которой нужно вернуть
- **timestamp_before** (*timestamp*) – Временная метка в формате ISO 8601, изменения до которой нужно вернуть

GET /api/changes/(int: *id*)/

Возвращает информацию об изменении в переводе.

Параметры

- **id** (*int*) – Идентификатор изменения

JSON -объект ответа

- **unit** (*string*) – URL-адрес связанного объекта единицы перевода
- **translation** (*string*) – URL-адрес связанного объекта перевода
- **component** (*string*) – URL-адрес связанного объекта компонента
- **glossary_term** (*string*) – URL-адрес связанного объекта термина глоссария
- **user** (*string*) – URL-адрес связанного объекта пользователя
- **author** (*string*) – URL-адрес связанного объекта автора
- **timestamp** (*timestamp*) – временная метка события
- **action** (*int*) – числовой идентификатор действия
- **action_name** (*string*) – текстовое описание действия
- **target** (*string*) – изменённый текст или информация о событии
- **id** (*int*) – идентификатор изменения

1.12.12 Снимки экрана

Добавлено в версии 2.14.

GET /api/screenshots/

Возвращает список снимков экрана для исходных строк.

См.также:

Атрибуты объекта снимка экрана перевода задокументированы в разделе *GET /api/screenshots/(int:id)/*.

GET /api/screenshots/(int: *id*)/

Возвращает информацию о снимке экрана.

Параметры

- **id** (*int*) – Идентификатор снимка экрана

JSON -объект ответа

- **name** (*string*) – название снимка экрана
- **component** (*string*) – URL-адрес связанного объекта компонента
- **file_url** (*string*) – URL-адрес для скачивания файла; смотрите описание *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – ссылка на связанную информацию об исходной строке; смотрите описание *GET /api/units/(int:id)/*

GET /api/screenshots/(int: *id*)/file/

Скачивает изображение снимка экрана.

Параметры

- **id** (*int*) – Идентификатор снимка экрана

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

Заменяет изображение снимка экрана.

Параметры

- **id** (*int*) – Идентификатор снимка экрана

Параметры формы

- **file image** – Загруженный файл

Пример CURL:

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

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

Связывает исходную строку со снимком экрана.

Параметры

- **id** (*int*) – Идентификатор снимка экрана

Параметры формы

- **string unit_id** – Идентификатор единицы перевода

JSON -объект ответа

- **name** (*string*) – название снимка экрана
- **component** (*string*) – URL-адрес связанного объекта компонента

- **file_url** (*string*) – URL-адрес для скачивания файла; смотрите описание *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – ссылка на связанную информацию об исходной строке; смотрите описание *GET /api/units/(int:id)/*

DELETE /api/screenshots/(int: *id*)/units/
int: *unit_id* Remove source string association with screenshot.

Параметры

- **id** (*int*) – Идентификатор снимка экрана
- **unit_id** – Source string unit ID

POST /api/screenshots/
Создаёт новый снимок экрана.

Параметры формы

- **file image** – Загруженный файл
- **string name** – Название снимка экрана
- **string project_slug** – Плашка проекта
- **string component_slug** – Плашка компонента

JSON -объект ответа

- **name** (*string*) – название снимка экрана
- **component** (*string*) – URL-адрес связанного объекта компонента
- **file_url** (*string*) – URL-адрес для скачивания файла; смотрите описание *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – ссылка на связанную информацию об исходной строке; смотрите описание *GET /api/units/(int:id)/*

PATCH /api/screenshots/(int: *id*)/
Edit partial information about screenshot.

Параметры

- **id** (*int*) – Идентификатор снимка экрана

JSON -объект ответа

- **name** (*string*) – название снимка экрана
- **component** (*string*) – URL-адрес связанного объекта компонента
- **file_url** (*string*) – URL-адрес для скачивания файла; смотрите описание *GET /api/screenshots/(int:id)/file/*
- **units** (*array*) – ссылка на связанную информацию об исходной строке; смотрите описание *GET /api/units/(int:id)/*

PUT /api/screenshots/(int: *id*)/
Edit full information about screenshot.

Параметры

- **id** (*int*) – Идентификатор снимка экрана

JSON -объект ответа

- **name** (*string*) – название снимка экрана
- **component** (*string*) – URL-адрес связанного объекта компонента
- **file_url** (*string*) – URL-адрес для скачивания файла; смотрите описание *GET /api/screenshots/(int:id)/file/*

- **units** (*array*) – ссылка на связанную информацию об исходной строке; смотрите описание *GET /api/units/(int:id)/*

DELETE /api/screenshots/(int: *id*)/
Delete screenshot.

Параметры

- **id** (*int*) – Идентификатор снимка экрана

1.12.13 Списки компонентов

Добавлено в версии 4.0.

GET /api/component-lists/
Возвращает список списков компонентов.

См.также:

Атрибуты объекта списка компонентов задокументированы в разделе *GET /api/component-lists/(str:slug)/*.

GET /api/component-lists/(str: *slug*)/
Возвращает информацию о списке компонентов.

Параметры

- **slug** (*string*) – Плашка списка компонентов

JSON -объект ответа

- **name** (*string*) – название списка компонентов
- **slug** (*string*) – плашка списка компонентов
- **show_dashboard** (*boolean*) – показывать ли его на панели управления
- **components** (*array*) – ссылка на связанные компоненты; смотрите описание *GET /api/components/(string:project)/(string:component)/*
- **auto_assign** (*array*) – правила автоматического назначения

PUT /api/component-lists/(str: *slug*)/
Изменяет параметры списка компонентов.

Параметры

- **slug** (*string*) – Плашка списка компонентов

JSON -объект запроса

- **name** (*string*) – название списка компонентов
- **slug** (*string*) – плашка списка компонентов
- **show_dashboard** (*boolean*) – показывать ли его на панели управления

PATCH /api/component-lists/(str: *slug*)/
Изменяет параметры списка компонентов.

Параметры

- **slug** (*string*) – Плашка списка компонентов

JSON -объект запроса

- **name** (*string*) – название списка компонентов
- **slug** (*string*) – плашка списка компонентов
- **show_dashboard** (*boolean*) – показывать ли его на панели управления

DELETE /api/component-lists/(**str:** *slug*)**/**
Удаляет список компонентов.

Параметры

- **slug** (*string*) – Плашка списка компонентов

POST /api/component-lists/(**str:** *slug*)**/components/**
Связывает компонент со списком компонентов.

Параметры

- **slug** (*string*) – Плашка списка компонентов

Параметры формы

- **string component_id** – Идентификатор компонента

DELETE /api/component-lists/(**str:** *slug*)**/components/**
str: *component_slug* Отвязывает компонент от списка компонентов.

Параметры

- **slug** (*string*) – Плашка списка компонентов
- **component_slug** (*string*) – Плашка компонента

1.12.14 Обработчики уведомлений

Обработчики уведомлений позволяют внешним приложениям уведомлять Weblate об обновлении репозитория системы контроля версий.

Вы можете использовать конечные точки репозитория для проектов, компонентов и переводов для обновления отдельных репозиториев; документацию смотрите в описании *POST /api/projects/(string:project)/repository/*.

GET /hooks/update/(**string:** *project*)**/**
string: *component***/** Не рекомендуется, начиная с версии 2.6: Пожалуйста, используйте вместо этой конечной точки точку *POST /api/components/(string:project)/(string:component)/repository/*, которая корректно работает с аутентификацией для проектов, ограниченных списками контроля доступа.

Запускает обновление компонента (извлечение кода из системы контроля версий и поиск в нём изменений перевода).

GET /hooks/update/(**string:** *project*)**/**
Не рекомендуется, начиная с версии 2.6: Пожалуйста, используйте вместо этой конечной точки точку *POST /api/projects/(string:project)/repository/*, которая корректно работает с аутентификацией для проектов, ограниченных списками контроля доступа.

Запускает обновление всех компонентов в проекте (извлечение кода из системы контроля версий и поиск в нём изменений перевода).

POST /hooks/github/
Специальный обработчик для обработки уведомлений GitHub и автоматического обновления соответствующих компонентов.

Примечание: GitHub включает прямую поддержку уведомлений Weblate: включите сервисный обработчик Weblate в настройках репозитория и установите URL-адрес в соответствие с URL-адресом вашей установки Weblate.

См.также:

Automatically receiving changes from GitHub Инструкция по настройке интеграции с GitHub

https://docs.github.com/en/github/extending-github/about-webhooks Общая информация о веб-обработчиках GitHub

ENABLE_HOOKS Для включения обработчиков для всего Weblate

POST /hooks/gitlab/

Специальный обработчик для обработки уведомлений GitLab и автоматического обновления соответствующих компонентов.

См.также:

Automatically receiving changes from GitLab Инструкция по настройке интеграции с GitLab

https://docs.gitlab.com/ce/user/project/integrations/webhooks.html Общая информация о веб-обработчиках GitLab

ENABLE_HOOKS Для включения обработчиков для всего Weblate

POST /hooks/bitbucket/

Специальный обработчик для обработки уведомлений Bitbucket и автоматического обновления соответствующих компонентов.

См.также:

Automatically receiving changes from Bitbucket Инструкция по настройке интеграции с Bitbucket

https://confluence.atlassian.com/bitbucket/manage-webhooks-735643732.html Общая информация о веб-обработчиках Bitbucket

ENABLE_HOOKS Для включения обработчиков для всего Weblate

POST /hooks/pagure/

Добавлено в версии 3.3.

Специальный обработчик для обработки уведомлений Pagure и автоматического обновления соответствующих компонентов.

См.также:

Automatically receiving changes from Pagure Инструкция по настройке интеграции с Pagure

https://docs.pagure.org/pagure/usage/using_webhooks.html Общая информация о веб-обработчиках Pagure

ENABLE_HOOKS Для включения обработчиков для всего Weblate

POST /hooks/azure/

Добавлено в версии 3.8.

Специальный обработчик для обработки уведомлений Azure Repos и автоматического обновления соответствующих компонентов.

См.также:

Automatically receiving changes from Azure Repos Инструкция по настройке интеграции с Azure

https://docs.microsoft.com/ru-ru/azure/devops/service-hooks/services/webhooks Общая информация о веб-обработчиках Azure Repos

ENABLE_HOOKS Для включения обработчиков для всего Weblate

POST /hooks/gitea/

Добавлено в версии 3.9.

Специальный обработчик для обработки уведомлений Gitea и автоматического обновления соответствующих компонентов.

См.также:

Automatically receiving changes from Gitea Repos Инструкция по настройке интеграции с Gitea

https://docs.gitea.io/en-us/webhooks/ Общая информация о веб-обработчиках Gitea

ENABLE_HOOKS Для включения обработчиков для всего Weblate

POST /hooks/gitee/

Добавлено в версии 3.9.

Специальный обработчик для обработки уведомлений Gitee и автоматического обновления соответствующих компонентов.

См.также:

Automatically receiving changes from Gitee Repos Инструкция по настройке интеграции с Gitee

https://gitee.com/help/categories/40 Общая информация о веб-обработчиках Gitee

ENABLE_HOOKS Для включения обработчиков для всего Weblate

1.12.15 Экспорты

Чтобы позволить вам и дальше обрабатывать данные, Weblate предоставляет вам различные виды экспорта.

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

Параметры запроса

- **format** (*string*) – Выходной формат: `json` или `csv`

Не рекомендуется, начиная с версии 2.6: Пожалуйста, используйте вместо этой конечной точки точки *GET /api/components/(string:project)/(string:component)/statistics/* или *GET /api/translations/(string:project)/(string:component)/(string:language)/statistics/*; они также разрешает доступ к проектам, контролируемым списками контроля доступа.

Возвращает статистику по указанному компоненту в заданном формате.

Пример запроса:

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

Пример ответа:

```
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/
↪"
    },
]
```


1.12.16 RSS-каналы

Изменения в переводах экспортируются в RSS-каналы.

GET /exports/rss/(**string:** *project*)**/**
string: *component***/string:** *language***/** Возвращает RSS-ленту с последними изменениями в переводе.

GET /exports/rss/(**string:** *project*)**/**
string: *component***/** Возвращает RSS-ленту с последними изменениями в компоненте.

GET /exports/rss/(**string:** *project*)**/**
Возвращает RSS-ленту с последними изменениями в проекте.

GET /exports/rss/language/(**string:** *language*)**/**
Возвращает RSS-ленту с последними изменениями в языке.

GET /exports/rss/
Возвращает RSS-ленту с последними изменениями в экземпляре Weblate.

См.также:

RSS в Википедии

1.13 Клиент Weblate

Добавлено в версии 2.7: Полная поддержка вспомогательной команды wlc существует с версии Weblate 2.7. Если вы используете более старую версию, могут возникнуть некоторые несовместимости с API.

1.13.1 Установка

Клиент Weblate поставляется отдельно и включает в себя Python'ий модуль. Для использования нижеприведённых команд необходимо установить модуль `wlc`:

```
pip3 install wlc
```

1.13.2 Начало работы

Конфигурация wlc хранится в `~/.config/weblate`, пожалуйста, создайте её в соответствии со своим окружением:

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

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

После этого вы сможете вызывать команды на умолчательном сервере:

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

См.также:

Файлы конфигурации

1.13.3 Обзор

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

Какая операция должна быть выполнена на самом деле указывает параметр `command`.

1.13.4 Описание

Weblate Client — это Python'ья библиотека и инструмент командной строки для удалённого управления Weblate с помощью его API. Инструмент командной строки может быть вызван как команда **wlc**, которая построена на основе модуля *wlc*.

Параметры для всего сайта

Программа принимает следующие параметры для всего экземпляра приложения, которые должны быть введены до какой-либо подкоманды.

--format {csv,json,text,html}
Определяет формат вывода.

--url URL
Укажите URL-адрес API. Переопределяет любое значение, найденное в конфигурационном файле, смотрите раздел *Файлы конфигурации*. URL-адрес должен заканчиваться на /api/, например, https://hosted.weblate.org/api/.

--key KEY
Определяет пользовательский ключ API. Переопределяет любое значение, найденное в конфигурационном файле, смотрите раздел *Файлы конфигурации*. Свой ключ вы можете найти в своём профиле на Weblate.

--config PATH
Переопределяет путь к конфигурационному файлу, смотрите раздел *Файлы конфигурации*.

--config-section SECTION
Переопределяет используемый раздел конфигурационного файла, смотрите раздел *Файлы конфигурации*.

Подкоманды

Доступны следующие подкоманды:

version
Распечатывает текущую версию.

list-languages
Перечисляет используемые в Weblate языки.

list-projects
Перечисляет проекты в Weblate.

list-components
Перечисляет компоненты в Weblate.

list-translations
Перечисляет переводы в Weblate.

show
Показывает объект Weblate (перевод, компонент или проект).

ls
Перечисляет объект Weblate (перевод, компонент или проект).

commit
Фиксирует изменения, внесённые в объект Weblate (перевод, компонент или проект).

pull
Извлекает изменения из удалённого репозитория в объект Weblate (перевод, компонент или проект).

push
Отправляет изменения объекта Weblate (перевода, компонента или проекта) в удалённый репозиторий.

reset

Добавлено в версии 0.7: Поддерживается с версии wlc 0.7.

Сбрасывает изменения в объекте Weblate (переводе, компоненте или проекте) для приведения их в соответствие с удалённым репозиторием.

cleanup

Добавлено в версии 0.9: Поддерживается с версии wlc 0.9.

Удаляет любые неотслеживаемые изменения в объекте Weblate (переводе, компоненте или проекте) для приведения их в соответствие с удалённым репозиторием.

repo

Отображает статус репозитория для данного объекта Weblate (перевода, компонента или проекта).

statistics

Отображает подробную статистику по данному объекту Weblate (переводу, компоненту или проекту).

lock-status

Добавлено в версии 0.5: Поддерживается с версии wlc 0.5.

Отображает статус блокировки.

lock

Добавлено в версии 0.5: Поддерживается с версии wlc 0.5.

Блокирует компонент в Weblate от дальнейшего перевода.

unlock

Добавлено в версии 0.5: Поддерживается с версии wlc 0.5.

Разблокирует перевод компонента Weblate.

changes

Добавлено в версии 0.7: Поддерживается с версии wlc 0.7 и Weblate 2.10.

Отображает изменения для данного объекта.

download

Добавлено в версии 0.7: Поддерживается с версии wlc 0.7.

Скачивает файл перевода.

--convert

Преобразовывает формат файла, если не указан, преобразования на сервере не происходит и файл скачивается в том виде, в котором он сохранён в репозитории.

--output

Указывает файл, в который нужно сохранить вывод, если не задать, вывод будет распечатан в стандартный поток вывода.

upload

Добавлено в версии 0.9: Поддерживается с версии wlc 0.9.

Загружает файл перевода.

--overwrite

Перезаписать существующие переводы во время загрузки.

--input

Файл, из которого читается содержимое, если не задать, чтение будет производиться со стандартного потока ввода.

1.13.5 Файлы конфигурации

.weblate Проектный файл конфигурации

~/.config/weblate Пользовательский файл конфигурации

/etc/xdg/weblate Системный файл конфигурации

Программа следует спецификации XDG, поэтому вы можете скорректировать пути размещения конфигурационных файлов переменными окружения `XDG_CONFIG_HOME` или `XDG_CONFIG_DIRS`.

Можно настроить следующие параметры, находящиеся в разделе `[weblate]` (изменить этот раздел вы можете ключом командной строки *`--config-section`*):

key
Ключ API для доступа к Weblate.

url
Адрес API сервера, по умолчанию установлен в `http://127.0.0.1:8000/api/`.

translation
Путь к переводу по умолчанию — компоненту или проекту.

Файл конфигурации является INI-файлом, например:

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

Дополнительно ключи API могут быть сохранены в разделе `[keys]`:

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

Это позволяет вам сохранять ключи в ваших личных настройках, а конфигурацию `.weblate` в хранить в репозитории системы контроля версий, чтобы wlc знала, с каким сервером ей следует общаться.

1.13.6 Примеры

Печать текущей версии программы:

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

Список всех проектов:

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

Также вы можете указать, с каким проектом должен работать wlc:

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

$ wlc show
```

(continues on next page)

(продолжение с предыдущей страницы)

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

При такой настройке закоммитить отложенные изменения в текущем проекте проще простого:

```
$ wlc commit
```

1.14 Python API Weblate'a

1.14.1 Установка

Python API поставляется отдельно, чтобы он был у вас появился, вам нужно установить *клиент Weblate*: `wlc`.

```
pip install wlc
```

1.14.2 wlc

WeblateException

exception wlc.**WeblateException**

Базовый класс для всех исключений.

Weblate

class wlc.**Weblate**(*key=''*, *url=None*, *config=None*)

Параметры

- **key** (*str*) – Ключ пользователя
- **url** (*str*) – URL-адрес сервера API, если не указан, используется адрес по умолчанию
- **config** (wlc.config.WeblateConfig) – Объект конфигурации, переопределяет любые другие параметры.

Класс доступа к API, определения ключа API и, необязательно, URL-адреса API.

get(*path*)

Параметры **path** (*str*) – Путь запроса

Тип результата object

Выполняет единичный GET-вызов API.

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

Параметры **path** (*str*) – Путь запроса

Тип результата object

Выполняет единичный GET-вызов API.

1.14.3 wlc.config

WeblateConfig

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

Параметры **section** (*str*) – Используемый раздел конфигурации

Анализатор файлов конфигурации, следующий спецификации XDG.

load (*path=None*)

Параметры **path** (*str*) – Путь к файлу, из которого загружается конфигурация.

Загружает конфигурацию из файла, если файл не указан, загружает её из конфигурационного файла *wlc* (`~/.config/wlc`), расположенного по пути вашей конфигурации XDG (`/etc/xdg/wlc`).

1.14.4 wlc.main

wlc.main.**main** (*settings=None*, *stdout=None*, *args=None*)

Параметры

- **settings** (*list*) – Параметры для переопределения в виде списка кортежей
- **stdout** (*object*) – Файловый объект стандартного потока вывода печати вывода, по умолчанию используется `sys.stdout`
- **args** (*list*) – Аргументы командной строки для обработки, по умолчанию используется `sys.args`

Главная точка входа для интерфейса командной строки.

@wlc.main.**register_command** (*command*)

Декоратор для регистрации класса *Command* в основном анализаторе, используемом функцией *main()*.

Command

class wlc.main.**Command** (*args*, *config*, *stdout=None*)

Главный класс для вызова команд.

ГЛАВА 2

Документация для администратора

2.1 Инструкции по настройке

2.1.1 Установка Weblate

Установка с помощью Docker'а

С помощью докеризированного развёртывания Weblate вы можете запустить ваш личный экземпляр Weblate за считанные секунды. Все зависимости Weblate уже включены в образ. В качестве базы данных по умолчанию настроена PostgreSQL.

Требования к оборудованию

Weblate должен без проблем работать на всём современном оборудовании, ниже приведена минимальная конфигурация, необходимая для запуска Weblate на одном узле (сам Weblate, база данных и веб-сервер):

- 2 ГБ оперативной памяти
- 2-х ядерный процессор
- 1 ГБ дискового пространства

Чем больше памяти, тем лучше — она используется для кэширования на всех уровнях (на уровне файловой системы, уровне базы данных и уровне Weblate).

Множество одновременно работающих пользователей увеличивают количество необходимых ядер процессора. Для сотен компонентов перевода рекомендуется не менее 4 Гб оперативной памяти.

Примечание: Фактические требования к вашей установке Weblate сильно зависят от размера управляемых ею переводов.

Установка

В следующих примерах предполагается, что у вас есть рабочее окружение Docker'а, в котором установлен `docker-compose`. Для получения конкретных инструкций ознакомьтесь с документацией к Docker'у.

1. Склонируйте репозиторий weblate-docker:

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

2. Создайте файл `docker-compose.override.yml` со своими настройками. Полный список переменных окружения приведён в разделе *Переменные окружения Docker'а*.

```
version: '3'
services:
  weblate:
    ports:
      - 80:8080
    environment:
      WEBLATE_EMAIL_HOST: smtp.example.com
      WEBLATE_EMAIL_HOST_USER: user
      WEBLATE_EMAIL_HOST_PASSWORD: pass
      WEBLATE_SERVER_EMAIL: weblate@example.com
      WEBLATE_DEFAULT_FROM_EMAIL: weblate@example.com
      WEBLATE_SITE_DOMAIN: weblate.example.com
      WEBLATE_ADMIN_PASSWORD: password for the admin user
      WEBLATE_ADMIN_EMAIL: weblate.admin@example.com
```

Примечание: Если переменная *`WEBLATE_ADMIN_PASSWORD`* не установлена, то пользователь-администратор создаётся со случайным паролем, отображаемым при первом запуске.

Приведённый пример настраивает Weblate на прослушку порта 80, чтобы его изменить, отредактируйте отображение портов в файле `docker-compose-override.yml`.

3. Запустите контейнеры Weblate:

```
docker-compose up
```

Наслаждайтесь своим развёрнутым Weblate'ом, он доступен на порту 80 контейнера `weblate`.

Изменено в версии 2.15-2: Недавно настройка изменилась, ранее существовал отдельный контейнер веб-сервера, но с версии 2.15-2 веб-сервер встроен в контейнер Weblate.

Изменено в версии 3.7.1-6: In July 2019 (starting with the 3.7.1-6 tag), the containers are not running as a root user. This has changed the exposed port from 80 to 8080.

См.также:

Вызов команд управления

Контейнер Docker с поддержкой HTTPS

Please see *Установка* for generic deployment instructions, this section only mentions differences compared to it.

Использование собственных SSL-сертификатов

Добавлено в версии 3.8-3.

In case you have own SSL certificate you want to use, simply place the files into the Weblate data volume (see *Docker container volumes*):

- `ssl/fullchain.pem` containing the certificate including any needed CA certificates
- `ssl/privkey.pem` containing the private key

Both of these files must be owned by the same user as the one starting the docker container and have file mask set to `600` (readable and writable only by the owning user).

Additionally, Weblate container will now accept SSL connections on port 4443, you will want to include the port forwarding for HTTPS in docker compose override:

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

If you already host other sites on the same server, it is likely ports `80` and `443` are used by a reverse proxy, such as NGINX. To pass the HTTPS connection from NGINX to the docker container, you can use the following configuration:

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

    server_name <SITE_URL>;
    ssl_certificate /etc/letsencrypt/live/<SITE>/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/<SITE>/privkey.pem;

    location / {
            proxy_set_header HOST $host;
            proxy_set_header X-Forwarded-Proto https;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Host $server_name;
            proxy_pass https://127.0.0.1:<EXPOSED_DOCKER_PORT>;
    }
}
```

Replace `<SITE_URL>`, `<SITE>` and `<EXPOSED_DOCKER_PORT>` with actual values from your environment.

Автоматический выпуск SSL-сертификатов с помощью сервиса Let's Encrypt

In case you want to use Let's Encrypt automatically generated SSL certificates on public installation, you need to add a reverse HTTPS proxy an additional Docker container, https-portal will be used for that. This is made use of in the `docker-compose-https.yml` file. Then create a `docker-compose-https.override.yml` file with your settings:

```
version: '3'
services:
  weblate:
    environment:
      WEBLATE_EMAIL_HOST: smtp.example.com
      WEBLATE_EMAIL_HOST_USER: user
      WEBLATE_EMAIL_HOST_PASSWORD: pass
      WEBLATE_SITE_DOMAIN: weblate.example.com
      WEBLATE_ADMIN_PASSWORD: password for admin user
  https-portal:
    environment:
      DOMAINS: 'weblate.example.com -> http://weblate:8080'
```

Whenever invoking **`docker-compose`** you need to pass both files to it, and then do:

```
docker-compose -f docker-compose-https.yml -f docker-compose-https.override.yml␣
↪build
docker-compose -f docker-compose-https.yml -f docker-compose-https.override.yml up
```

Обновление контейнера Docker'а

Usually it is good idea to only update the Weblate container and keep the PostgreSQL container at the version you have, as upgrading PostgreSQL is quite painful and in most cases does not bring many benefits.

You can do this by sticking with the existing docker-compose and just pull the latest images and then restart:

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

The Weblate database should be automatically migrated on first startup, and there should be no need for additional manual actions.

Примечание: Upgrades across 3.0 are not supported by Weblate. If you are on 2.x series and want to upgrade to 3.x, first upgrade to the latest 3.0.1-x (at time of writing this it is the `3.0.1-7`) image, which will do the migration and then continue upgrading to newer versions.

You might also want to update the `docker-compose` repository, though it's not needed in most case. Please beware of PostgreSQL version changes in this case as it's not straightforward to upgrade the database, see GitHub issue for more info.

Вход под администратором

После настройки контейнера вы можете войти в систему под пользователем *admin* с паролем, указанным в переменной окружения `WEBLATE_ADMIN_PASSWORD`, или случайным паролем, сгенерированным при первом запуске, если он не был установлен явно.

Для сброса пароля пользователя *admin* перезапустите контейнер с переменной `WEBLATE_ADMIN_PASSWORD`, установленной в новое значение пароля.

См.также:

`WEBLATE_ADMIN_PASSWORD`, `WEBLATE_ADMIN_NAME`, `WEBLATE_ADMIN_EMAIL`

Переменные окружения Docker'a

Многие *конфигурационные настройки* Weblate в контейнере Docker могут быть установлены через переменные окружения:

Общие параметры

WEBLATE_DEBUG

Настраивает отладочный режим Django, используя её переменную `DEBUG`.

Пример:

```
environment:
  WEBLATE_DEBUG: 1
```

См.также:

Отключение отладочного режима.

WEBLATE_LOGLEVEL

Настраивает подробность ведения журнала.

WEBLATE_SITE_TITLE

Changes the site-title shown in the header of all pages.

WEBLATE_SITE_DOMAIN

Настраивает домен сайта.

Подсказка: В случае, если он не установлен, используется первый элемент из переменной окружения `WEBLATE_ALLOWED_HOSTS`.

См.также:

Установка правильного домена сайта, `SITE_DOMAIN`

WEBLATE_ADMIN_NAME

WEBLATE_ADMIN_EMAIL

Настраивает имя и электронную почту администратора сайта. Используется как для параметра `ADMINS`, так и для создания пользователя *admin* (подробнее смотрите в описании переменной окружения `WEBLATE_ADMIN_PASSWORD`).

Пример:

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

См.также:

Вход под администратором, *Правильная настройка администраторов*, `ADMINS`

WEBLATE_ADMIN_PASSWORD

Устанавливает пароль для пользователя *admin*.

- Если не установлен и пользователя *admin* не существует, он создаётся со случайным паролем, который показывается при первом запуске контейнера.
- Если не установлен и пользователя *admin* не существует, никакие действия не выполняются.
- Если установлен, при каждом запуске контейнера пользователю *admin* устанавливаются соответствующие пароль `WEBLATE_ADMIN_PASSWORD`, имя `WEBLATE_ADMIN_NAME` и адрес электронной почты `WEBLATE_ADMIN_EMAIL`.

Предупреждение: Хранение пароля в файле конфигурации может представлять собой угрозу безопасности. Подумайте об использовании этой переменной только для начальной установки (или пусть Weblate сгенерирует при начальной загрузке случайный пароль) или для восстановления пароля.

См.также:

Вход под администратором, `WEBLATE_ADMIN_PASSWORD`, `WEBLATE_ADMIN_NAME`, `WEBLATE_ADMIN_EMAIL`

WEBLATE_SERVER_EMAIL

WEBLATE_DEFAULT_FROM_EMAIL

Настраивает адрес для исходящих сообщений электронной почты.

См.также:

Настройка отправки электронной почты

WEBLATE_ALLOWED_HOSTS

Настраивает разрешённые имена HTTP-хостов с помощью параметра `ALLOWED_HOSTS`.

По умолчанию установлен в `*`, что позволяет использовать все имена хостов.

Пример:

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

См.также:

`ALLOWED_HOSTS`, *Настройка разрешённых хостов*, *Установка правильного домена сайта*

WEBLATE_REGISTRATION_OPEN

Настраивает статус открытия регистрации, переключая параметр `REGISTRATION_OPEN`.

Пример:

```
environment:
  WEBLATE_REGISTRATION_OPEN: 0
```

WEBLATE_REGISTRATION_ALLOW_BACKENDS

Настраивает через параметр `REGISTRATION_ALLOW_BACKENDS` методы аутентификации, которые можно использовать для создания новой учётной записи.

Пример:

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

(continues on next page)

(продолжение с предыдущей страницы)

WEBLATE_TIME_ZONE

Настраивает используемый в Weblate часовой пояс, смотрите описание параметра `TIME_ZONE`.

Примечание: Для изменения часового пояса самого контейнера Docker используйте переменную окружения `TZ`.

Пример:

```
environment:
  WEBLATE_TIME_ZONE: Europe/Prague
```

WEBLATE_ENABLE_HTTPS

Заставляет Weblate думать, что он работает за обратным HTTPS-прокси, что принуждает Weblate использовать HTTPS в ссылках писем электронной почты и API или устанавливать у кук флаги безопасности.

Примечание: Этот параметр не включает разрешение на приём контейнером Weblate'а соединений по HTTPS, его нужно настроить отдельно, для примеров смотрите раздел *Контейнер Docker с поддержкой HTTPS*.

Пример:

```
environment:
  WEBLATE_ENABLE_HTTPS: 1
```

См.также:

Установка правильного домена сайта

WEBLATE_IP_PROXY_HEADER

Lets Weblate fetch the IP address from any given HTTP header. Use this when using a reverse proxy in front of the Weblate container.

Enables *`IP_BEHIND_REVERSE_PROXY`* and sets *`IP_PROXY_HEADER`*.

Примечание: The format must conform to Django's expectations. Django transforms raw HTTP header names as follows:

- converts all characters to uppercase
- replaces any hyphens with underscores
- prepends `HTTP_` prefix

So `X-Forwarded-For` would be mapped to `HTTP_X_FORWARDED_FOR`.

Пример:

```
environment:
  WEBLATE_IP_PROXY_HEADER: HTTP_X_FORWARDED_FOR
```

WEBLATE_SECURE_PROXY_SSL_HEADER

A tuple representing a HTTP header/value combination that signifies a request is secure. This is needed when Weblate is running behind a reverse proxy doing SSL termination which does not pass standard HTTPS headers.

Пример:

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

См.также:

SECURE_PROXY_SSL_HEADER

WEBLATE_REQUIRE_LOGIN

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

Пример:

```
environment:
  WEBLATE_REQUIRE_LOGIN: 1
```

WEBLATE_LOGIN_REQUIRED_URLS_EXCEPTIONS

WEBLATE_ADD_LOGIN_REQUIRED_URLS_EXCEPTIONS

WEBLATE_REMOVE_LOGIN_REQUIRED_URLS_EXCEPTIONS

Adds URL exceptions for login required for the whole Weblate installation using *LOGIN_REQUIRED_URLS_EXCEPTIONS*.

You can either replace whole settings, or modify default value using ADD and REMOVE variables.

WEBLATE_GOOGLE_ANALYTICS_ID

Configures ID for Google Analytics by changing *GOOGLE_ANALYTICS_ID*.

WEBLATE_GITHUB_USERNAME

Configures GitHub username for GitHub pull-requests by changing *GITHUB_USERNAME*.

См.также:

GitHub, *Установка hub*

WEBLATE_GITLAB_USERNAME

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

См.также:

GitLab Установка Lab

WEBLATE_GITLAB_HOST

Настраивает хост GitLab'а для GitLab'овских запросов на слияние

См.также:

GitLab Установка Lab

WEBLATE_GITLAB_TOKEN

Настраивает токен доступа GitLab'а для GitLab'овских запросов на слияние

См.также:

GitLab Установка Lab

WEBLATE_SIMPLIFY_LANGUAGES

Настраивает политику упрощения языка, смотрите описание параметра *SIMPLIFY_LANGUAGES*.

WEBLATE_DEFAULT_ACCESS_CONTROL

Настраивает значение по умолчанию для *управления доступом* в новых проектах, смотрите описание параметра *DEFAULT_ACCESS_CONTROL*.

WEBLATE_DEFAULT_RESTRICTED_COMPONENT

Настраивает значение по умолчанию для *ограниченного доступа* в новых компонентах, смотрите описание параметра *DEFAULT_RESTRICTED_COMPONENT*.

WEBLATE_DEFAULT_TRANSLATION_PROPAGATION

Настраивает значение по умолчанию для *разрешения распространения переводов* в новых компонентах, смотрите описание параметра *DEFAULT_TRANSLATION_PROPAGATION*.

WEBLATE_AKISMET_API_KEY
Настраивает API-ключ Akismet, смотрите описание параметра *AKISMET_API_KEY*.

WEBLATE_GPG_IDENTITY
Настраивает GPG для подписи коммитов, смотрите описание параметра *WEBLATE_GPG_IDENTITY*.

См.также:

Signing Git commits with GnuPG

WEBLATE_URL_PREFIX
Настраивает префикс URL-адреса, на котором запущен Weblate, смотрите описание параметра *URL_PREFIX*.

WEBLATE_SILENCED_SYSTEM_CHECKS
Настраивает проверки, которые вы не хотите видеть, смотрите описание параметра SILENCED_SYSTEM_CHECKS.

WEBLATE_CSP_SCRIPT_SRC

WEBLATE_CSP_IMG_SRC

WEBLATE_CSP_CONNECT_SRC

WEBLATE_CSP_STYLE_SRC

WEBLATE_CSP_FONT_SRC
Позволяет настраивать HTTP-заголовок политики безопасности содержимого Content-Security-Policy.

См.также:

Политика безопасности содержимого, *CSP_SCRIPT_SRC*, *CSP_IMG_SRC*, *CSP_CONNECT_SRC*, *CSP_STYLE_SRC*, *CSP_FONT_SRC*

Параметры машинного перевода

WEBLATE_MT_AWS_REGION

WEBLATE_MT_AWS_ACCESS_KEY_ID

WEBLATE_MT_AWS_SECRET_ACCESS_KEY
Настраивает машинный перевод от *AWS*.

```
environment:
  WEBLATE_MT_AWS_REGION: us-east-1
  WEBLATE_MT_AWS_ACCESS_KEY_ID: AKIAIOSFODNN7EXAMPLE
  WEBLATE_MT_AWS_SECRET_ACCESS_KEY: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

WEBLATE_MT_DEEPL_KEY
Включает машинный перевод от *DeepL* и устанавливает параметр *MT_DEEPL_KEY*

WEBLATE_MT_DEEPL_API_VERSION
Настраивает используемую версию API *DeepL*, смотрите описание параметра *MT_DEEPL_API_VERSION*.

WEBLATE_MT_GOOGLE_KEY
Включает машинный перевод от *Google Translate* и устанавливает параметр *MT_GOOGLE_KEY*

WEBLATE_MT_MICROSOFT_COGNITIVE_KEY
Enables *Microsoft Cognitive Services Translator* and sets *MT_MICROSOFT_COGNITIVE_KEY*

WEBLATE_MT_MICROSOFT_ENDPOINT_URL
Enables *Microsoft Cognitive Services Translator* and sets *MT_MICROSOFT_ENDPOINT_URL*

WEBLATE_MT_MICROSOFT_BASE_URL
Enables *Microsoft Cognitive Services Translator* and sets *MT_MICROSOFT_BASE_URL*

WEBLATE_MT_MODERNMT_KEY
Enables *ModernMT* and sets `MT_MODERNMT_KEY`.

WEBLATE_MT_MYMEMORY_ENABLED

Enables *MyMemory* machine translation and sets `MT_MYMEMORY_EMAIL` to `WEBLATE_ADMIN_EMAIL`.

Пример:

```
environment:
  WEBLATE_MT_MYMEMORY_ENABLED: 1
```

WEBLATE_MT_GLOSBE_ENABLED
Включает машинный перевод от *Glosbe*.

```
environment:
  WEBLATE_MT_GLOSBE_ENABLED: 1
```

WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED
Включает машинный перевод от *терминологической службы Майкрософт*.

```
environment:
  WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED: 1
```

WEBLATE_MT_SAP_BASE_URL

WEBLATE_MT_SAP_SANDBOX_APIKEY

WEBLATE_MT_SAP_USERNAME

WEBLATE_MT_SAP_PASSWORD

WEBLATE_MT_SAP_USE_MT
Настраивает машинный перевод от *SAP Translation Hub*.

```
environment:
    WEBLATE_MT_SAP_BASE_URL: "https://example.hana.ondemand.com/translationhub/
↪api/v1/"
    WEBLATE_MT_SAP_USERNAME: "user"
    WEBLATE_MT_SAP_PASSWORD: "password"
    WEBLATE_MT_SAP_USE_MT: 1
```

Параметры аутентификации

LDAP

WEBLATE_AUTH_LDAP_SERVER_URI

WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE

WEBLATE_AUTH_LDAP_USER_ATTR_MAP

WEBLATE_AUTH_LDAP_BIND_DN

WEBLATE_AUTH_LDAP_BIND_PASSWORD

WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS

WEBLATE_AUTH_LDAP_USER_SEARCH

WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER
Конфигурация аутентификации через LDAP.

Пример прямой привязки:

```
environment:
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE: uid=%(user)s,ou=People,dc=example,dc=net
  # map weblate 'full_name' to ldap 'name' and weblate 'email' attribute to
↪'mail' ldap attribute.
  # another example that can be used with OpenLDAP: 'full_name:cn,email:mail'
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
```

Пример поиска и привязки:

```
environment:
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_BIND_DN: CN=ldap,CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_BIND_PASSWORD: password
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
  WEBLATE_AUTH_LDAP_USER_SEARCH: CN=Users,DC=example,DC=com
```

Пример поиска и привязки в Active Directory:

```
environment:
  WEBLATE_AUTH_LDAP_BIND_DN: CN=ldap,CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_BIND_PASSWORD: password
  WEBLATE_AUTH_LDAP_SERVER_URI: ldap://ldap.example.org
  WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS: 0
  WEBLATE_AUTH_LDAP_USER_ATTR_MAP: full_name:name,email:mail
  WEBLATE_AUTH_LDAP_USER_SEARCH: CN=Users,DC=example,DC=com
  WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER: (sAMAccountName=%(user)s)
```

См.также:

Аутентификация через LDAP

GitHub

WEBLATE_SOCIAL_AUTH_GITHUB_KEY

WEBLATE_SOCIAL_AUTH_GITHUB_SECRET

Включает *аутентификацию через GitHub*.

Bitbucket

WEBLATE_SOCIAL_AUTH_BITBUCKET_KEY

WEBLATE_SOCIAL_AUTH_BITBUCKET_SECRET

Включает *аутентификацию через Bitbucket*.

Facebook

WEBLATE_SOCIAL_AUTH_FACEBOOK_KEY

WEBLATE_SOCIAL_AUTH_FACEBOOK_SECRET

Включает *Facebook OAuth 2*.

Google

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_KEY

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_DOMAINS

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_EMAILS
Включает *Google OAuth 2*.

GitLab

WEBLATE_SOCIAL_AUTH_GITLAB_KEY

WEBLATE_SOCIAL_AUTH_GITLAB_SECRET

WEBLATE_SOCIAL_AUTH_GITLAB_API_URL
Включает *GitLab OAuth 2*.

Azure Active Directory

WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_KEY

WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET
Включает аутентификацию через Azure Active Directory, смотрите *Microsoft Azure Active Directory*.

Azure Active Directory с поддержкой Tenant

WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY

WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECRET

WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENANT_ID
Включает аутентификацию через Azure Active Directory с поддержкой Tenant, смотрите *Microsoft Azure Active Directory*.

Keycloak

WEBLATE_SOCIAL_AUTH_KEYCLOAK_KEY

WEBLATE_SOCIAL_AUTH_KEYCLOAK_SECRET

WEBLATE_SOCIAL_AUTH_KEYCLOAK_PUBLIC_KEY

WEBLATE_SOCIAL_AUTH_KEYCLOAK_ALGORITHM

WEBLATE_SOCIAL_AUTH_KEYCLOAK_AUTHORIZATION_URL

WEBLATE_SOCIAL_AUTH_KEYCLOAK_ACCESS_TOKEN_URL
Включает аутентификацию через Keycloak, смотрите документацию.

Поставщики Linux

You can enable authentication using Linux vendors authentication services by setting following variables to any value.

WEBLATE_SOCIAL_AUTH_FEDORA

WEBLATE_SOCIAL_AUTH_OPENSUSE

WEBLATE_SOCIAL_AUTH_UBUNTU

Slack

WEBLATE_SOCIAL_AUTH_SLACK_KEY

SOCIAL_AUTH_SLACK_SECRET
Enables Slack authentication, see *Slack*.

SAML

Self-signed SAML keys are automatically generated on first container startup. In case you want to use own keys, place the certificate and private key in `/app/data/ssl/saml.crt` and `/app/data/ssl/saml.key`.

WEBLATE_SAML_IDP_ENTITY_ID

WEBLATE_SAML_IDP_URL

WEBLATE_SAML_IDP_X509CERT
SAML Identity Provider settings, see *Аутентификация через SAML*.

Другие параметры аутентификации

WEBLATE_NO_EMAIL_AUTH
При установке в любое значение отключает аутентификацию через электронную почту.

Настройки базы данных PostgreSQL

База данных создается в файле `docker-compose.yml`, поэтому эти настройки влияют как на контейнер Weblate, так и на контейнер PostgreSQL.

См.также:

Настройка базы данных для Weblate

POSTGRES_PASSWORD
Пароль PostgreSQL.

POSTGRES_USER
Имя пользователя PostgreSQL.

POSTGRES_DATABASE
Имя базы данных PostgreSQL.

POSTGRES_HOST
Имя хоста или IP-адрес сервера PostgreSQL. По умолчанию равен `database`.

POSTGRES_PORT
Порт сервера PostgreSQL. По умолчанию не установлен (используется значение по умолчанию).

POSTGRES_SSL_MODE
Configure how PostgreSQL handles SSL in connection to the server, for possible choices see SSL Mode Descriptions

Параметры резервного копирования базы данных

См.также:

Дампы данных для резервного копирования

WEBLATE_DATABASE_BACKUP
Configures the daily database dump using `DATABASE_BACKUP`. Defaults to `plain`.

Настройка сервера кэширования

Using Redis is strongly recommended by Weblate and you have to provide a Redis instance when running Weblate in Docker.

См.также:

Включение кэширования

REDIS_HOST
The Redis server hostname or IP address. Defaults to `cache`.

REDIS_PORT
Порт сервера Redis. По умолчанию установлен в `6379`.

REDIS_DB
Нобер базы данных Redis, по умолчанию установлен в `1`.

REDIS_PASSWORD
The Redis server password, not used by default.

REDIS_TLS
Enables using SSL for Redis connection.

REDIS_VERIFY_SSL
Can be used to disable SSL certificate verification for Redis connection.

Настройка почтового сервера

To make outgoing e-mail work, you need to provide a mail server.

Пример конфигурации TLS:

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

Пример конфигурации SSL:

```
environment:
    WEBLATE_EMAIL_HOST: smtp.example.com
    WEBLATE_EMAIL_PORT: 465
    WEBLATE_EMAIL_HOST_USER: user
    WEBLATE_EMAIL_HOST_PASSWORD: pass
    WEBLATE_EMAIL_USE_TLS: 0
    WEBLATE_EMAIL_USE_SSL: 1
```

См.также:

Настройка исходящей почты

WEBLATE_EMAIL_HOST

Mail server hostname or IP address.

См.также:

WEBLATE_EMAIL_PORT, *WEBLATE_EMAIL_USE_SSL*, *WEBLATE_EMAIL_USE_TLS*, EMAIL_HOST

WEBLATE_EMAIL_PORT

Порт почтового сервера, по умолчанию установлен в 25.

См.также:

EMAIL_PORT

WEBLATE_EMAIL_HOST_USER

E-mail authentication user.

См.также:

EMAIL_HOST_USER

WEBLATE_EMAIL_HOST_PASSWORD

E-mail authentication password.

См.также:

EMAIL_HOST_PASSWORD

WEBLATE_EMAIL_USE_SSL

Whether to use an implicit TLS (secure) connection when talking to the SMTP server. In most e-mail documentation, this type of TLS connection is referred to as SSL. It is generally used on port 465. If you are experiencing problems, see the explicit TLS setting *WEBLATE_EMAIL_USE_TLS*.

См.также:

WEBLATE_EMAIL_PORT, *WEBLATE_EMAIL_USE_TLS*, EMAIL_USE_SSL

WEBLATE_EMAIL_USE_TLS

Whether to use a TLS (secure) connection when talking to the SMTP server. This is used for explicit TLS connections, generally on port 587 or 25. If you are experiencing connections that hang, see the implicit TLS setting *WEBLATE_EMAIL_USE_SSL*.

См.также:

WEBLATE_EMAIL_PORT, *WEBLATE_EMAIL_USE_SSL*, EMAIL_USE_TLS

WEBLATE_EMAIL_BACKEND

Configures Django back-end to use for sending e-mails.

См.также:

Настройка отправки электронной почты, EMAIL_BACKEND

Отчёты об ошибках

It is recommended to collect errors from the installation systematically, see *Сбор отчётов об ошибках*.

To enable support for Rollbar, set the following:

ROLLBAR_KEY
Your Rollbar post server access token.

ROLLBAR_ENVIRONMENT
Ваше окружение Rollbar, по умолчанию установлена в `production`.

To enable support for Sentry, set following:

SENTRY_DSN
Your Sentry DSN.

SENTRY_ENVIRONMENT
Your Sentry Environment (optional).

Изменение включённых приложений, проверок, надстроек или автоматических исправлений

Добавлено в версии 3.8-5.

The built-in configuration of enabled checks, addons or autofixes can be adjusted by the following variables:

WEBLATE_ADD_APPS

WEBLATE_REMOVE_APPS

WEBLATE_ADD_CHECK

WEBLATE_REMOVE_CHECK

WEBLATE_ADD_AUTOFIX

WEBLATE_REMOVE_AUTOFIX

WEBLATE_ADD_ADDONS

WEBLATE_REMOVE_ADDONS

Пример:

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

См.также:

CHECK_LIST, *AUTOFIX_LIST*, *WEBLATE_ADDONS*, INSTALLED_APPS

Параметры контейнера

CELERY_MAIN_OPTIONS

CELERY_NOTIFY_OPTIONS

CELERY_TRANSLATE_OPTIONS

CELERY_MEMORY_OPTIONS

CELERY_BACKUP_OPTIONS

CELERY_BEAT_OPTIONS

These variables allow you to adjust Celery worker options. It can be useful to adjust concurrency (`--concurrency 16`) or use different pool implementation (`--pool=gevent`).

By default, the number of concurrent workers matches the number of processors (except the backup worker, which is supposed to run only once).

Пример:

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

См.также:

Celery worker options, *Фоновые задачи с использованием Celery*

UWSGI_WORKERS

Configure how many uWSGI workers should be executed.

It defaults to number of processors + 1.

Пример:

```
environment:
  UWSGI_WORKERS: 32
```

Docker container volumes

There is single data volume exported by the Weblate container. The other service containers (PostgreSQL or Redis) have their data volumes as well, but those are not covered by this document.

The data volume is used to store Weblate persistent data such as cloned repositories or to customize Weblate installation.

The placement of the Docker volume on host system depends on your Docker configuration, but usually it is stored in `/var/lib/docker/volumes/weblate-docker_weblate-data/_data/`. In the container it is mounted as `/app/data`.

См.также:

Docker volumes documentation

Дальнейшая настройка конфигурации

You can further customize Weblate installation in the data volume, see *Docker container volumes*.

Пользовательские файлы конфигурации

You can additionally override the configuration in `/app/data/settings-override.py` (see *Docker container volumes*). This is executed after all environment settings are loaded, so it gets completely set up, and can be used to customize anything.

Замена логотипа и других статических файлов

Добавлено в версии 3.8-5.

The static files coming with Weblate can be overridden by placing into `/app/data/python/customize/static` (see *Docker container volumes*). For example creating `/app/data/python/customize/static/favicon.ico` will replace the favicon.

Подсказка: The files are copied to the corresponding location upon container startup, so a restart of Weblate is needed after changing the content of the volume.

Alternatively you can also include own module (see *Customizing Weblate*) and add it as separate volume to the Docker container, for example:

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

Добавление собственных Python'ьих модулей

Добавлено в версии 3.8-5.

You can place own Python modules in `/app/data/python/` (see *Docker container volumes*) and they can be then loaded by Weblate, most likely by using *Пользовательские файлы конфигурации*.

См.также:

Customizing Weblate

Настройка hub

In order to use the GitHub's pull-request feature, you must initialize your hub configuration by entering the Weblate container and executing an arbitrary Hub command. For example:

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

The username passed for credentials must be the same as `GITHUB_USERNAME`.

См.также:

GitHub, *Установка hub*

Настройка lab

In order to use GitLab's merge-request feature, you must initialize the `lab` configuration by entering the Weblate container and executing the `lab` command. For example:

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

You can also use environment variables to configure `lab` on each container start. Just add `WEBLATE_GITLAB_USERNAME`, `WEBLATE_GITLAB_HOST``and ``WEBLATE_GITLAB_TOKEN` to your env configuration.

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

The `access_token` passed for lab configuration must be same as *`GITLAB_USERNAME`*.

См.также:

GitLab Установка Lab

Select your machine - local or cloud providers

With Docker Machine you can create your Weblate deployment either on your local machine, or on any large number of cloud-based deployments on e.g. Amazon AWS, Greenhost, and many other providers.

Установка на Debian и Ubuntu

Требования к оборудованию

Weblate должен без проблем работать на всём современном оборудовании, ниже приведена минимальная конфигурация, необходимая для запуска Weblate на одном узле (сам Weblate, база данных и веб-сервер):

- 2 ГБ оперативной памяти
- 2-х ядерный процессор
- 1 ГБ дискового пространства

Чем больше памяти, тем лучше — она используется для кэширования на всех уровнях (на уровне файловой системы, уровне базы данных и уровне Weblate).

Множество одновременно работающих пользователей увеличивают количество необходимых ядер процессора. Для сотен компонентов перевода рекомендуется не менее 4 Гб оперативной памяти.

Примечание: Фактические требования к вашей установке Weblate сильно зависят от размера управляемых ею переводов.

Установка

Системные требования

Установите зависимости, необходимые для сборки Python'ьих модулей (смотрите раздел *Требования к программному обеспечению*):

```
apt install \
   libxml2-dev libxslt-dev libfreetype6-dev libjpeg-dev libz-dev libyaml-dev \
   libcairo-dev gir1.2-pango-1.0 libgirepository1.0-dev libacl1-dev libssl-dev \
   build-essential python3-gdbm python3-dev python3-pip python3-virtualenv␣
↪virtualenv git
```

Установите желаемые необязательные зависимости в зависимости от функций, которые вы собираетесь использовать (смотрите раздел *Необязательные зависимости*):

```
apt install tesseract-ocr libtesseract-dev libleptonica-dev
```

При желании установите программное обеспечение для запуска рабочего сервера, смотрите разделы *Запуск сервера*, *Настройка базы данных для Weblate* и *Фоновые задачи с использованием Celery*. В зависимости от размера вашей установки вы можете захотеть запустить эти компоненты на выделенных серверах.

Инструкции по установке на местах:

```
# Web server option 1: NGINX and uWSGI
apt install nginx uwsgi uwsgi-plugin-python3

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

# Caching backend: Redis
apt install redis-server

# Database server: PostgreSQL
apt install postgresql postgresql-contrib

# SMTP server
apt install exim4
```

Python'ьи модули

Подсказка: Для установки Weblate мы используем виртуальное окружение, предоставляемое пакетом virtualenv, чтобы отделить Weblate от вашей системы. Если вы не знакомы с этим понятием, обратитесь к документации пакета.

1. Создайте виртуальное окружение для Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Активируйте виртуальное окружение для Weblate:

   ```
   . ~/weblate-env/bin/activate
   ```

3. Установите Weblate, включая все его зависимости:

   ```
   pip install Weblate
   ```

4. Установите драйвер базы данных:

```
pip install psycopg2-binary
```

5. Установите желаемые необязательные зависимости в зависимости от функций, которые вы собираетесь использовать (некоторые из них могут потребовать дополнительных системных библиотек, проверьте раздел *Необязательные зависимости*):

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

Настройка Weblate

Примечание: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **`weblate`** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`
2. По своему усмотрению подкорректируйте значения в новом файле `settings.py`. Для тестирования вы можете использовать поставляемый пример, но для рабочей среды вы захотите внести в него изменения, смотрите раздел *Настройка конфигурации*.
3. Создайте базу данных и её структуру для Weblate (пример настройки использует PostgreSQL, для установки в рабочую среду ознакомьтесь разделом *Настройка базы данных для Weblate*):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Запуск сервера* and *Обслуживание статических файлов*):

```
weblate collectstatic
```

6. Сожмите файлы JavaScript и CSS (необязательно, смотрите раздел *Сжатие клиентских активов*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Фоновые задачи с использованием Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Запустите сервер разработки (шаги для запуска рабочего сервера смотрите в разделе *Запуск сервера*):

```
weblate runserver
```

После установки

Поздравляем, теперь ваш сервер Weblate запущен и вы можете начать его использовать.

- Теперь вы можете получить доступ к Weblate по адресу `http://localhost:8000/`.
- Войдите в систему с учётными данными администратора, полученными во время установки, или зарегистрируйтесь в качестве нового пользователя.
- Теперь при помощи команды **`weblate`** и когда активно виртуальное окружение Weblate вы можете запускать его команды, смотрите раздел *Команды управления*.
- Остановить тестовый сервер можно комбинацией `Ctrl+C`.

Добавление перевода

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *Project configuration* for more details.

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

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See *Конфигурация компонента* for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see *Поддерживаемые форматы файлов* for more details.

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

Установка на SUSE и openSUSE

Требования к оборудованию

Weblate должен без проблем работать на всём современном оборудовании, ниже приведена минимальная конфигурация, необходимая для запуска Weblate на одном узле (сам Weblate, база данных и веб-сервер):

- 2 ГБ оперативной памяти
- 2-х ядерный процессор
- 1 ГБ дискового пространства

Чем больше памяти, тем лучше — она используется для кэширования на всех уровнях (на уровне файловой системы, уровне базы данных и уровне Weblate).

Множество одновременно работающих пользователей увеличивают количество необходимых ядер процессора. Для сотен компонентов перевода рекомендуется не менее 4 Гб оперативной памяти.

Примечание: Фактические требования к вашей установке Weblate сильно зависят от размера управляемых ею переводов.

Установка

Системные требования

Установите зависимости, необходимые для сборки Python'ьих модулей (смотрите раздел *Требования к программному обеспечению*):

```
zypper install \
   libxslt-devel libxml2-devel freetype-devel libjpeg-devel zlib-devel libyaml-
↪devel \
   cairo-devel typelib-1_0-Pango-1_0 gobject-introspection-devel libacl-devel \
   python3-pip python3-virtualenv python3-devel git
```

Установите желаемые необязательные зависимости в зависимости от функций, которые вы собираетесь использовать (смотрите раздел *Необязательные зависимости*):

```
zypper install tesseract-ocr tesseract-devel leptonica-devel
```

При желании установите программное обеспечение для запуска рабочего сервера, смотрите разделы *Запуск сервера*, *Настройка базы данных для Weblate* и *Фоновые задачи с использованием Celery*. В зависимости от размера вашей установки вы можете захотеть запустить эти компоненты на выделенных серверах.

Инструкции по установке на местах:

```
# Web server option 1: NGINX and uWSGI
zypper install nginx uwsgi uwsgi-plugin-python3

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

# Caching backend: Redis
zypper install redis-server

# Database server: PostgreSQL
zypper install postgresql postgresql-contrib

# SMTP server
zypper install postfix
```

Python'ьи модули

Подсказка: Для установки Weblate мы используем виртуальное окружение, предоставляемое пакетом virtualenv, чтобы отделить Weblate от вашей системы. Если вы не знакомы с этим понятием, обратитесь к документации пакета.

1. Создайте виртуальное окружение для Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Активируйте виртуальное окружение для Weblate:

   ```
   . ~/weblate-env/bin/activate
   ```

3. Установите Weblate, включая все его зависимости:

   ```
   pip install Weblate
   ```

4. Установите драйвер базы данных:

```
pip install psycopg2-binary
```

5. Установите желаемые необязательные зависимости в зависимости от функций, которые вы собираетесь использовать (некоторые из них могут потребовать дополнительных системных библиотек, проверьте раздел *Необязательные зависимости*):

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

Настройка Weblate

Примечание: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **`weblate`** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`
2. По своему усмотрению подкорректируйте значения в новом файле `settings.py`. Для тестирования вы можете использовать поставляемый пример, но для рабочей среды вы захотите внести в него изменения, смотрите раздел *Настройка конфигурации*.
3. Создайте базу данных и её структуру для Weblate (пример настройки использует PostgreSQL, для установки в рабочую среду ознакомьтесь разделом *Настройка базы данных для Weblate*):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Запуск сервера* and *Обслуживание статических файлов*):

```
weblate collectstatic
```

6. Сожмите файлы JavaScript и CSS (необязательно, смотрите раздел *Сжатие клиентских активов*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Фоновые задачи с использованием Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Запустите сервер разработки (шаги для запуска рабочего сервера смотрите в разделе *Запуск сервера*):

```
weblate runserver
```

После установки

Поздравляем, теперь ваш сервер Weblate запущен и вы можете начать его использовать.

- Теперь вы можете получить доступ к Weblate по адресу `http://localhost:8000/`.
- Войдите в систему с учётными данными администратора, полученными во время установки, или зарегистрируйтесь в качестве нового пользователя.
- Теперь при помощи команды **`weblate`** и когда активно виртуальное окружение Weblate вы можете запускать его команды, смотрите раздел *Команды управления*.
- Остановить тестовый сервер можно комбинацией `Ctrl+C`.

Добавление перевода

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *Project configuration* for more details.

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

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See *Конфигурация компонента* for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see *Поддерживаемые форматы файлов* for more details.

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

Установка на RedHat, Fedora и CentOS

Требования к оборудованию

Weblate должен без проблем работать на всём современном оборудовании, ниже приведена минимальная конфигурация, необходимая для запуска Weblate на одном узле (сам Weblate, база данных и веб-сервер):

- 2 ГБ оперативной памяти
- 2-х ядерный процессор
- 1 ГБ дискового пространства

Чем больше памяти, тем лучше — она используется для кэширования на всех уровнях (на уровне файловой системы, уровне базы данных и уровне Weblate).

Множество одновременно работающих пользователей увеличивают количество необходимых ядер процессора. Для сотен компонентов перевода рекомендуется не менее 4 Гб оперативной памяти.

Примечание: Фактические требования к вашей установке Weblate сильно зависят от размера управляемых ею переводов.

Установка

Системные требования

Установите зависимости, необходимые для сборки Python'ьих модулей (смотрите раздел *Требования к программному обеспечению*):

```
dnf install \
   libxslt-devel libxml2-devel freetype-devel libjpeg-devel zlib-devel libyaml-
↪devel \
   cairo-devel pango-devel gobject-introspection-devel libacl-devel \
   python3-pip python3-virtualenv python3-devel git
```

Установите желаемые необязательные зависимости в зависимости от функций, которые вы собираетесь использовать (смотрите раздел *Необязательные зависимости*):

```
dnf install tesseract-langpack-eng tesseract-devel leptonica-devel
```

При желании установите программное обеспечение для запуска рабочего сервера, смотрите разделы *Запуск сервера*, *Настройка базы данных для Weblate* и *Фоновые задачи с использованием Celery*. В зависимости от размера вашей установки вы можете захотеть запустить эти компоненты на выделенных серверах.

Инструкции по установке на местах:

```
# Web server option 1: NGINX and uWSGI
dnf install nginx uwsgi uwsgi-plugin-python3

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

# Caching backend: Redis
dnf install redis

# Database server: PostgreSQL
dnf install postgresql postgresql-contrib

# SMTP server
dnf install postfix
```

Python'ьи модули

Подсказка: Для установки Weblate мы используем виртуальное окружение, предоставляемое пакетом virtualenv, чтобы отделить Weblate от вашей системы. Если вы не знакомы с этим понятием, обратитесь к документации пакета.

1. Создайте виртуальное окружение для Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Активируйте виртуальное окружение для Weblate:

   ```
   . ~/weblate-env/bin/activate
   ```

3. Установите Weblate, включая все его зависимости:

   ```
   pip install Weblate
   ```

4. Установите драйвер базы данных:

```
pip install psycopg2-binary
```

5. Установите желаемые необязательные зависимости в зависимости от функций, которые вы собираетесь использовать (некоторые из них могут потребовать дополнительных системных библиотек, проверьте раздел *Необязательные зависимости*):

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

Настройка Weblate

Примечание: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **`weblate`** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`
2. По своему усмотрению подкорректируйте значения в новом файле `settings.py`. Для тестирования вы можете использовать поставляемый пример, но для рабочей среды вы захотите внести в него изменения, смотрите раздел *Настройка конфигурации*.
3. Создайте базу данных и её структуру для Weblate (пример настройки использует PostgreSQL, для установки в рабочую среду ознакомьтесь разделом *Настройка базы данных для Weblate*):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Запуск сервера* and *Обслуживание статических файлов*):

```
weblate collectstatic
```

6. Сожмите файлы JavaScript и CSS (необязательно, смотрите раздел *Сжатие клиентских активов*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Фоновые задачи с использованием Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Запустите сервер разработки (шаги для запуска рабочего сервера смотрите в разделе *Запуск сервера*):

```
weblate runserver
```

После установки

Поздравляем, теперь ваш сервер Weblate запущен и вы можете начать его использовать.

- Теперь вы можете получить доступ к Weblate по адресу `http://localhost:8000/`.
- Войдите в систему с учётными данными администратора, полученными во время установки, или зарегистрируйтесь в качестве нового пользователя.
- Теперь при помощи команды **`weblate`** и когда активно виртуальное окружение Weblate вы можете запускать его команды, смотрите раздел *Команды управления*.
- Остановить тестовый сервер можно комбинацией `Ctrl+C`.

Добавление перевода

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *Project configuration* for more details.

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

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See *Конфигурация компонента* for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see *Поддерживаемые форматы файлов* for more details.

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

Установка на macOS

Требования к оборудованию

Weblate должен без проблем работать на всём современном оборудовании, ниже приведена минимальная конфигурация, необходимая для запуска Weblate на одном узле (сам Weblate, база данных и веб-сервер):

- 2 ГБ оперативной памяти
- 2-х ядерный процессор
- 1 ГБ дискового пространства

Чем больше памяти, тем лучше — она используется для кэширования на всех уровнях (на уровне файловой системы, уровне базы данных и уровне Weblate).

Множество одновременно работающих пользователей увеличивают количество необходимых ядер процессора. Для сотен компонентов перевода рекомендуется не менее 4 Гб оперативной памяти.

Примечание: Фактические требования к вашей установке Weblate сильно зависят от размера управляемых ею переводов.

Установка

Системные требования

Установите зависимости, необходимые для сборки Python'ьих модулей (смотрите раздел *Требования к программному обеспечению*):

```
brew install pango libjpeg python git libyaml gobject-introspection
pip3 install virtualenv
```

Make sure pip will be able to find the `libffi` version provided by homebrew — this will be needed during the installation build step.

```
export PKG_CONFIG_PATH="/usr/local/opt/libffi/lib/pkgconfig"
```

Установите желаемые необязательные зависимости в зависимости от функций, которые вы собираетесь использовать (смотрите раздел *Необязательные зависимости*):

```
brew install tesseract
```

При желании установите программное обеспечение для запуска рабочего сервера, смотрите разделы *Запуск сервера*, *Настройка базы данных для Weblate* и *Фоновые задачи с использованием Celery*. В зависимости от размера вашей установки вы можете захотеть запустить эти компоненты на выделенных серверах.

Инструкции по установке на местах:

```
# Web server option 1: NGINX and uWSGI
brew install nginx uwsgi

# Web server option 2: Apache with ``mod_wsgi``
brew install httpd

# Caching backend: Redis
brew install redis

# Database server: PostgreSQL
brew install postgresql
```

Python'ьи модули

Подсказка: Для установки Weblate мы используем виртуальное окружение, предоставляемое пакетом virtualenv, чтобы отделить Weblate от вашей системы. Если вы не знакомы с этим понятием, обратитесь к документации пакета.

1. Создайте виртуальное окружение для Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Активируйте виртуальное окружение для Weblate:

   ```
   . ~/weblate-env/bin/activate
   ```

3. Установите Weblate, включая все его зависимости:

   ```
   pip install Weblate
   ```

4. Установите драйвер базы данных:

```
pip install psycopg2-binary
```

5. Установите желаемые необязательные зависимости в зависимости от функций, которые вы собираетесь использовать (некоторые из них могут потребовать дополнительных системных библиотек, проверьте раздел *Необязательные зависимости*):

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

Настройка Weblate

Примечание: Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to **`weblate`** command as `~/weblate-env/bin/weblate`.

1. Copy the file `~/weblate-env/lib/python3.7/site-packages/weblate/settings_example.py` to `~/weblate-env/lib/python3.7/site-packages/weblate/settings.py`
2. По своему усмотрению подкорректируйте значения в новом файле `settings.py`. Для тестирования вы можете использовать поставляемый пример, но для рабочей среды вы захотите внести в него изменения, смотрите раздел *Настройка конфигурации*.
3. Создайте базу данных и её структуру для Weblate (пример настройки использует PostgreSQL, для установки в рабочую среду ознакомьтесь разделом *Настройка базы данных для Weblate*):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see *Запуск сервера* and *Обслуживание статических файлов*):

```
weblate collectstatic
```

6. Сожмите файлы JavaScript и CSS (необязательно, смотрите раздел *Сжатие клиентских активов*):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See *Фоновые задачи с использованием Celery* for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Запустите сервер разработки (шаги для запуска рабочего сервера смотрите в разделе *Запуск сервера*):

```
weblate runserver
```

После установки

Поздравляем, теперь ваш сервер Weblate запущен и вы можете начать его использовать.

- Теперь вы можете получить доступ к Weblate по адресу `http://localhost:8000/`.
- Войдите в систему с учётными данными администратора, полученными во время установки, или зарегистрируйтесь в качестве нового пользователя.
- Теперь при помощи команды **`weblate`** и когда активно виртуальное окружение Weblate вы можете запускать его команды, смотрите раздел *Команды управления*.
- Остановить тестовый сервер можно комбинацией `Ctrl+C`.

Добавление перевода

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See *Project configuration* for more details.

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

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See *Конфигурация компонента* for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see *Поддерживаемые форматы файлов* for more details.

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

Установка из исходников

1. Please follow the installation instructions for your system first:
   - *Установка на Debian и Ubuntu*
   - *Установка на SUSE и openSUSE*
   - *Установка на RedHat, Fedora и CentOS*
2. Grab the latest Weblate sources using Git (or download a tarball and unpack that):

   ```
   git clone https://github.com/WeblateOrg/weblate.git weblate-src
   ```

   Alternatively you can use released archives. You can download them from our website <https://weblate.org/>. Those downloads are cryptographically signed, please see *Проверка подписей выпусков*.

3. Install current Weblate code into the virtualenv:

   ```
   . ~/weblate-env/bin/activate
   pip install -e weblate-src
   ```

4. Copy `weblate/settings_example.py` to `weblate/settings.py`.
5. По своему усмотрению подкорректируйте значения в новом файле `settings.py`. Для тестирования вы можете использовать поставляемый пример, но для рабочей среды вы захотите внести в него изменения, смотрите раздел *Настройка конфигурации*.
6. Create the database used by Weblate, see *Настройка базы данных для Weblate*.
7. Build Django tables, static files and initial data (see *Наполнение базы данных* and *Обслуживание статических файлов*):

```
weblate migrate
weblate collectstatic
weblate compress
weblate compilemessages
```

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

Установка на OpenShift

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

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

В зависимости от ваших настроек и опыта выберите подходящий для вас способ установки:

- *Установка с помощью Docker'а*, рекомендуется для рабочей среды.
- Установка в виртуальное окружение, рекомендуется для рабочей среды:
  - *Установка на Debian и Ubuntu*
  - *Установка на SUSE и openSUSE*
  - *Установка на RedHat, Fedora и CentOS*
  - *Установка на macOS*
- *Установка из исходников*, рекомендуется для разработки.
- *Установка на OpenShift*.

2.1.2 Требования к программному обеспечению

Операционная система

Weblate работает под Linux, FreeBSD и MacOS. Скорее всего он будет работать и под другими Unix-подобными системами.

Работа Weblate под Windows не поддерживается. Тем не менее, работать под ней он всё ещё может и патчи под неё с радостью принимаются.

Другие сервисы

Для своей работы Weblate использует другие сервисы. Вам понадобятся, по крайней мере, следующие запущенные сервисы:

- Сервер базы данных PostgreSQL, смотрите раздел *Настройка базы данных для Weblate*.
- Сервер Redis для кэша и очереди задач, смотрите раздел *Фоновые задачи с использованием Celery*.
- SMTP-сервер для исходящей почты, смотрите раздел *Настройка исходящей почты*.

Зависимости Python'а

Weblate написан на Python'е и поддерживает Python 3.6 или более новую версию. Вы можете установить зависимости с помощью команды pip или из пакетов вашего дистрибутива, полный список зависимостей находится в файле `requirements.txt`.

Основные зависимости:

Django https://www.djangoproject.com/

Celery https://docs.celeryproject.org/

Translate Toolkit https://toolkit.translatehouse.org/

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

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

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

Необязательные зависимости

Для работы некоторых функций Weblate необходимы следующие модули. Полный их список можно найти в файле `requirements-optional.txt`.

`Mercurial` (необязательная зависимость для поддержки репозиториев Mercurial) https://www.mercurial-scm.org/

`phply` (необязательная зависимость для поддержки PHP) https://github.com/viraptor/phply

`tesserocr` (необязательная зависимость для поддержки распознавания текста со снимков экрана) https://github.com/sirfz/tesserocr

`akismet` (необязательная зависимость для защиты предложений от спама) https://github.com/ubernostrum/akismet

`ruamel.yaml` (необязательная зависимость для поддержки *файлов YAML*) https://pypi.org/project/ruamel.yaml/

`Zeep` (необязательная зависимость для поддержки *терминологической службы Майкрософт*) https://docs.python-zeep.org/

`aeidon` (необязательная зависимость для поддержки *файлов субтитров*) https://pypi.org/project/aeidon/

Зависимости серверной части базы данных

Weblate поддерживает PostgreSQL, MySQL и MariaDB, подробнее смотрите в разделе *Настройка базы данных для Weblate* и документации по реализации поддержки соответствующей базы данных.

Другие системные требования

В системе должны быть установлены следующие зависимости:

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

Pango, Cairo и связанные заголовочные файлы и данные интроспекции gir https://cairographics.org/, https://pango.gnome.org/, смотрите раздел *Pango и Cairo*

`hub` (необязательная зависимость для отправки запросов на извлечение в GitHub) https://hub.github.com/

`git-review` (необязательная зависимость для поддержки Gerrit) https://pypi.org/project/git-review/

`git-svn` (необязательная зависимость для поддержки Subversion) https://git-scm.com/docs/git-svn

`tesseract` и его данные (необязательная зависимость для поддержки распознавания текста со снимков экрана
https://github.com/tesseract-ocr/tesseract

Зависимости времени компиляции

Для компиляции некоторых *Python'ьих зависимостей* может потребоваться установка их зависимостей. Это зависит от того, как вы их устанавливаете, так что для получения помощи по установке обратитесь к документации соответствующих пакетов. Зависимости зависимостей вам не понадобятся, если вы используете предварительно собранные «колёса» при их установке через `pip` или устанавливаете их через пакетный менеджер вашего дистрибутива.

Pango и Cairo

Изменено в версии 3.7.

Weblate использует Pango и Cairo для отрисовки растровых виджетов (смотрите раздел *Продвижение перевода*) и проверки отрисовки текста (смотрите раздел *Управление шрифтами*). Для правильной установки их Python'ьих привязок сначала нужно установить системные библиотеки — вам нужны и Cairo, и Pango, которым, в свою очередь, нужна Glib. Все они должны быть установлены с файлами для разработки и данными интроспекции GObject.

2.1.3 Проверка подписей выпусков

Выпуски Weblate подписываются криптографической подписью выпускающего разработчика. В настоящее время им является Михаил Чихарж. Отпечаток его PGP-ключа:

```
63CB 1DF1 EF12 CF2A C0EE 5A32 9C27 B313 42B7 511D
```

а дополнительную идентификационную информацию вы можете получить по адресу <https://keybase.io/nijel>.

Вы должны удостовериться, что подпись соответствует скачанному вами архиву. Таким образом, вы сможете быть уверены, что используете тот же код, который был выпущен. Также вы должны проверить дату подписания, чтобы удостовериться, что вы скачали последнюю версию.

Каждый архив сопровождается файлами `.asc`, содержащими его PGP-подпись. После того, как вы разместите оба файла в одном каталоге, вы сможете проверить подпись:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Ne 3. března 2019, 16:43:15 CET
gpg:                using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: Can't check signature: public key not found
```

Как видите, gpg жалуется на то, что он не знает открытого ключа. На данный момент вы должны сделать одно из следующего:

- Использовать wkd для скачивания ключа:

```
$ gpg --auto-key-locate wkd --locate-keys michal@cihar.com
pub   rsa4096 2009-06-17 [SC]
      63CB1DF1EF12CF2AC0EE5A329C27B31342B7511D
uid           [ultimate] Michal Čihař <michal@cihar.com>
uid           [ultimate] Michal Čihař <nijel@debian.org>
uid           [ultimate] [jpeg image of size 8848]
uid           [ultimate] Michal Čihař (Braiins) <michal.cihar@braiins.cz>
sub   rsa4096 2009-06-17 [E]
sub   rsa4096 2015-09-09 [S]
```

- Скачать связку ключей с сервера Михаила, а затем импортировать её командой:

```
$ gpg --import wmxth3chu9jfxdxywj1skpmhsj311mzm
```

- Скачать и импортировать ключ с одного из серверов ключей:

```
$ gpg --keyserver hkp://pgp.mit.edu --recv-keys␣
↪87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: key 9C27B31342B7511D: "Michal Čihař <michal@cihar.com>" imported
gpg: Total number processed: 1
gpg:              unchanged: 1
```

Это немного улучшит ситуацию — на данный момент вы можете проверить правильность подписи с данного ключа, но всё равно не можете доверять используемому в ключе имени:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Ne 3. března 2019, 16:43:15 CET
gpg:                using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: Good signature from "Michal Čihař <michal@cihar.com>" [ultimate]
gpg:                 aka "Michal Čihař <nijel@debian.org>" [ultimate]
gpg:                 aka "[jpeg image of size 8848]" [ultimate]
gpg:                 aka "Michal Čihař (Braiins) <michal.cihar@braiins.cz>"␣
↪[ultimate]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 63CB 1DF1 EF12 CF2A C0EE  5A32 9C27 B313 42B7 511D
```

Проблема в том, что выдать ключ с таким именем может кто угодно. Вам нужно убедиться, что ключ действительно принадлежит указанному лицу. Руководство GNU по обеспечению конфиденциальности раскрывает эту тему в главе Проверка достоверности ключей. Самый надёжный метод — лично встретиться с разработчиком и обменяться отпечатками ключей, однако вы также можете положиться на сеть доверия. Таким образом, вы можете транзитивно доверять ключу через подписи других людей, которые встречались с разработчиком лично.

Когда достоверность ключа будет установлена, предупреждение больше появляться не будет:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Sun Mar  3 16:43:15 2019 CET
gpg:                using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: Good signature from "Michal Čihař <michal@cihar.com>" [ultimate]
gpg:                 aka "Michal Čihař <nijel@debian.org>" [ultimate]
gpg:                 aka "[jpeg image of size 8848]" [ultimate]
gpg:                 aka "Michal Čihař (Braiins) <michal.cihar@braiins.cz>"␣
↪[ultimate]
```

В случае недействительной подписи (архив был изменён), вы получите явную ошибку, независимо от того, доверяете ли вы ключу или нет:

```
$ gpg --verify Weblate-3.5.tar.xz.asc
gpg: Signature made Sun Mar  3 16:43:15 2019 CET
gpg:                using RSA key 87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: BAD signature from "Michal Čihař <michal@cihar.com>" [ultimate]
```

2.1.4 Разрешения файловой системы

Процесс Weblate должен мочь читать и писать в каталог, в котором он хранит свои данные — `DATA_DIR`. Все файлы в этом каталоге должны принадлежать и иметь разрешение на запись для пользователя, под которым запущен Weblate.

В конфигурации по умолчанию они расположены в том же дереве, что и исходный код Weblate, однако вы можете предпочесть переместить их в более подходящее место, например, в `/var/lib/weblate`.

Weblate попытается автоматически создать эти каталоги, но у него ничего не получится, если у него не будет на это разрешений.

Также вам следует внимательно запускать *команды управления*, поскольку они должны выполняться под тем же пользователем, под которым запущен сам Weblate, в противном случае разрешения на некоторые файлы могут быть выставлены неверно.

См.также:

Обслуживание статических файлов

2.1.5 Настройка базы данных для Weblate

Рекомендуется запускать Weblate с сервером баз данных PostgreSQL.

См.также:

Использование мощного движка базы данных, Базы данных, *Migrating from other databases to PostgreSQL*

PostgreSQL

PostgreSQL, как правило, является лучшим выбором для сайтов, написанных на Django. Это эталонная база данных, используемая для реализации слоя баз данных Django.

Примечание: Weblate использует расширение для триграмм, которое в некоторых случаях должно быть установлено отдельно. Ищите пакет `postgresql-contrib` или аналогичный.

См.также:

PostgreSQL notes

Создание базы данных в PostgreSQL

Обычно рекомендуется запускать Weblate в отдельной базе данных и под отдельной учётной записью:

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

# Create a database user called "weblate"
sudo -u postgres createuser --superuser --pwprompt weblate

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

Подсказка: If you don't want to make the Weblate user a superuser in PostgreSQL, you can omit that. In that case you will have to perform some of the migration steps manually as a PostgreSQL superuser in schema Weblate will use:

```
CREATE EXTENSION IF NOT EXISTS pg_trgm WITH SCHEMA weblate;
```

Настройка Weblate для использования PostgreSQL

Фрагмент настроек `settings.py` для PostgreSQL:

```
DATABASES = {
    'default': {
        # Database engine
        'ENGINE': 'django.db.backends.postgresql',
        # Database name
        'NAME': 'weblate',
        # Database user
        'USER': 'weblate',
        # Database password
        'PASSWORD': 'password',
        # Set to empty string for localhost
        'HOST': 'database.example.com',
        # Set to empty string for default
        'PORT': '',
    }
}
```

MySQL и MariaDB

Weblate также можно использовать с MySQL или MariaDB, ознакомитесь с предостережениями по их использованию вместе с Django в разделах Замечания по MySQL и Замечания по MariaDB его документации.

Подсказка: Some Weblate features will perform better with *PostgreSQL*. This includes searching and translation memory, which both utilize full-text features in the database and PostgreSQL implementation is superior.

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

Для Weblate рекомендуется следующая конфигурация:

- Используйте кодовую таблицу `utf8mb4` для представления символов с более высоких плоскостей Юникода (например, эмодзи).
- Настройте сервер на использование `Innodb_large_prefix` для разрешения более длинных индексов на текстовых полях.
- Установите уровень изоляции в `READ COMMITTED`.
- Режим SQL должен быть установлен в `STRICT_TRANS_TABLES`.

2.1.6 Другие настройки

Настройка исходящей почты

Weblate отправляет электронные письма по разным поводам — для активации учётной записи и по различным уведомлениям, настроенным пользователями. Для этого ему необходим доступ к SMTP-серверу.

Настройка почтового сервера производится с помощью следующих настроек: `EMAIL_HOST`, `EMAIL_HOST_PASSWORD`, `EMAIL_HOST_USER` и `EMAIL_PORT`. Их имена говорят сами за себя, а более подробную информацию вы можете найти в документации Django.

Примечание: Проверить, правильно ли работает исходящая почта, можно с помощью команды управления `sendtestemail` (для получения инструкций о том, как вызывать её в различных окружениях, смотрите раздел *Вызов команд управления*).

Работа за обратным прокси

Некоторые функции Weblate полагаются на возможность получения IP-адреса клиента. Среди этих функций *Ограничение частоты запросов*, *Spam protection* или *Журнал аудита*.

В стандартной конфигурации Weblate достаёт IP-адрес из переменной `REMOTE_ADDR`, которая устанавливается обработчиком WSGI.

Если же вы используете обратный прокси, то это поле, скорее всего, будет содержать его адрес. Вам нужно настроить Weblate на доверие дополнительным HTTP-заголовкам и доставать IP-адрес из них. Эту настройку нельзя включить по умолчанию, поскольку она разрешит подмену IP-адреса для установок, не использующих обратный прокси. Для самых обычных установок будет достаточно включения параметра *`IP_BEHIND_REVERSE_PROXY`*, но вам также может понадобиться подправить параметры *`IP_PROXY_HEADER`* и *`IP_PROXY_OFFSET`*.

См.также:

Spam protection, *Ограничение частоты запросов*, *Журнал аудита*, *`IP_BEHIND_REVERSE_PROXY`*, *`IP_PROXY_HEADER`*, *`IP_PROXY_OFFSET`*, `SECURE_PROXY_SSL_HEADER`

HTTP-прокси

Weblate выполняет команды системы контроля версий, а те принимают конфигурацию прокси из окружения. Рекомендуемый подход заключается в определении настроек прокси в файле `settings.py`:

```
import os
os.environ['http_proxy'] = "http://proxy.example.com:8080"
os.environ['HTTPS_PROXY'] = "http://proxy.example.com:8080"
```

См.также:

Переменные окружения для прокси

2.1.7 Настройка конфигурации

См.также:

Пример конфигурации

Скопируйте `weblate/settings_example.py` в `weblate/settings.py` и подправьте его в соответствии со своими настройками. Скорее всего, вы захотите настроить следующие параметры: `ADMINS`

> Список администраторов сайтов для получения уведомлений, когда что-то идёт не так, например, для получения уведомлений о неудачных слияниях, или об ошибках Django.
>
> **См.также:**
>
> `ADMINS`

`ALLOWED_HOSTS`

> Вы должны установить эту настройку в список хостов, которые должен обслуживать ваш сайт. Например:
>
> ```
> ALLOWED_HOSTS = ['demo.weblate.org']
> ```

Или же вы можете указать символ шаблона:

```
ALLOWED_HOSTS = ['*']
```

См.также:

ALLOWED_HOSTS, *WEBLATE_ALLOWED_HOSTS*, *Настройка разрешённых хостов*

SESSION_ENGINE

Настраивает то, как будут храниться ваши сессии. В случае, если вы оставили механизм хранения по умолчанию — в базе данных — вы должны настроить расписание для команды **weblate clearsessions**, удаляющей из базы данных данные об устаревших сессиях.

Если вы используете в качестве кэша Redis (смотрите раздел *Включение кэширования*), рекомендуется использовать его также и для сессий:

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
```

См.также:

Configuring the session engine, SESSION_ENGINE

DATABASES

Подключение к серверу базы данных, подробнее об этой настройке читайте в документации Django.

См.также:

Настройка базы данных для Weblate, DATABASES, Базы данных

DEBUG

Отключите эту настройку для любого рабочего сервера. При включённом режиме отладки Django в случае возникновения ошибки будет показывать трассировки стека пользователям, при его же отключении ошибки будут отправляться на адреса электронной почты из массива ADMINS (смотрите выше).

Кроме того, отладочный режим замедляет работу Weblate, поскольку в этом случае Django сохраняет внутри себя гораздо больше информации.

См.также:

DEBUG

DEFAULT_FROM_EMAIL

Адрес электронной почты отправителя для исходящих сообщений электронной почты, например, для писем регистрации.

См.также:

DEFAULT_FROM_EMAIL

SECRET_KEY

Ключ, используемый Django для подписывания некоторой информации, сохраняемой в куках, подробнее смотрите в разделе *Секретный ключ Django*.

См.также:

SECRET_KEY

SERVER_EMAIL

Адрес электронной почты, используемый в качестве адреса отправителя для отправки сообщений администратору, например, уведомлений о неудачных слияниях.

См.также:

SERVER_EMAIL

2.1.8 Наполнение базы данных

После того, как ваша конфигурация будет готова, вы можете запустить `weblate migrate` для создания структуры базы данных. Теперь вы можете создавать проекты перевода через интерфейс администратора.

В случае, если вы хотите запустить установку не в интерактивном режиме, вы можете использовать `weblate migrate --noinput`, а затем создать пользователя-администратора при помощи команды `createadmin`.

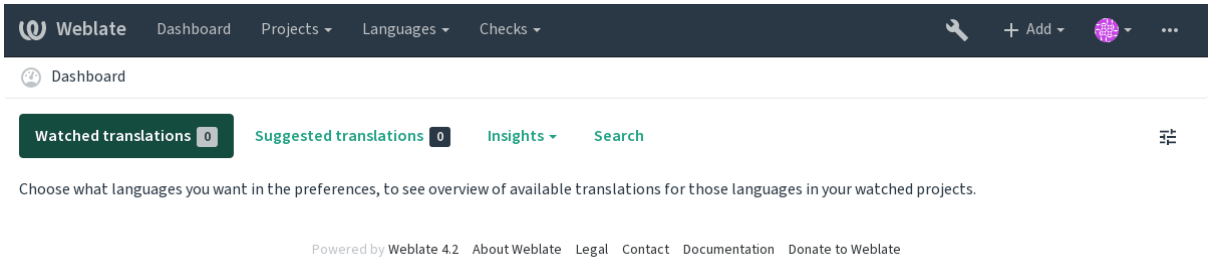
После того, как вы закончите, вы также должны проверить *Отчёт о производительности* в интерфейсе администратора, который может дать вам подсказки по исправлению потенциально неоптимальной конфигурации вашего сайта.

См.также:

Конфигурация, *Управление доступом*

2.1.9 Рабочая среда

Для установки в рабочую среду необходимо провести корректировки, описанные в следующих разделах. Наиболее критичные настройки вызовут предупреждение, которое обозначается восклицательным знаком в верхней панели, если вы вошли в систему под суперпользователем:



Также рекомендуется изучить все проверки, произведённые Django (хотя, возможно, исправлять их все нет необходимости):

```
weblate check --deploy
```

См.также:

Контрольный список развертывания

Отключение отладочного режима

Отключите отладочный режим Django (`DEBUG`), установив:

```
DEBUG = False
```

При включённом отладочном режиме Django сохраняет все выполненные запросы и при возникновении ошибок показывает пользователям обратную трассировку стека, что в рабочей среде не желательно.

См.также:

Настройка конфигурации

Правильная настройка администраторов

Укажите в настройке *ADMINS* правильные адреса администраторов, чтобы определить, кто будет получать электронную почту в случае, если на сервере что-то пойдёт не так, например:

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

См.также:

Настройка конфигурации

Установка правильного домена сайта

Поправьте имя сайта и домен в интерфейсе администратора, иначе ссылки в RSS или регистрационные письма не будут работать. Эта настройка осуществляется с помощью параметра *SITE_DOMAIN*, который должен содержать доменное имя сайта.

Изменено в версии 4.2: До выхода версии 4.2 вместо него использовался фреймворк «сайтов» Django, смотрите раздел Фреймворк «сайтов».

См.также:

Настройка разрешённых хостов, *Правильная настройка HTTPS* *SITE_DOMAIN*, *WEBLATE_SITE_DOMAIN*, *ENABLE_HTTPS*

Правильная настройка HTTPS

Настоятельно рекомендуется запускать Weblate с использованием шифрующего протокола HTTPS. После его включения в настройках необходимо установить параметр *ENABLE_HTTPS*:

```
ENABLE_HTTPS = True
```

Подсказка: Возможно, вы также захотите настроить HSTS, для получения более подробной информации смотрите документ SSL/HTTPS.

См.также:

ENABLE_HTTPS, *Настройка разрешённых хостов*, *Установка правильного домена сайта*

Установка правильного значения SECURE_HSTS_SECONDS

If your site is served over SSL, you have to consider setting a value for `SECURE_HSTS_SECONDS` in the `settings.py` to enable HTTP Strict Transport Security. By default it's set to 0 as shown below.

```
SECURE_HSTS_SECONDS = 0
```

Если для него установлено ненулевое целое значение, то `django.middleware.security.SecurityMiddleware` устанавливает заголовок HTTP Strict Transport Security у всех ответов, которые его ещё не имеют.

Предупреждение: Неправильная настройка этого параметра может привести к необратимому (на некоторое время) выходу из строя вашего сайта. Поэтому сперва ознакомьтесь с документацией по HTTP Strict Transport Security.

Использование мощного движка базы данных

Пожалуйста, используйте для рабочего окружения PostgreSQL, для получения более подробной информации смотрите раздел *Настройка базы данных для Weblate*.

См.также:

Настройка базы данных для Weblate, *Migrating from other databases to PostgreSQL*, *Настройка конфигурации*, Базы данных

Включение кэширования

Если возможно, используйте Redis из Django, настроив конфигурационную переменную `CACHES`, например:

```
CACHES = {
    'default': {
        'BACKEND': 'django_redis.cache.RedisCache',
        'LOCATION': 'redis://127.0.0.1:6379/0',
        # If redis is running on same host as Weblate, you might
        # want to use unix sockets instead:
        # 'LOCATION': 'unix:///var/run/redis/redis.sock?db=0',
        'OPTIONS': {
            'CLIENT_CLASS': 'django_redis.client.DefaultClient',
            'PARSER_CLASS': 'redis.connection.HiredisParser',
        }
    }
}
```

См.также:

Кеширование аватара, Фреймворк кэширования Django

Кеширование аватара

Помимо кэширования силами Django, Weblate выполняет кэширование аватаров. Для этой цели рекомендуется воспользоваться отдельным файловым кэшем:

```
CACHES = {
    'default': {
        # Default caching backend setup, see above
        'BACKEND': 'django_redis.cache.RedisCache',
        'LOCATION': 'unix:///var/run/redis/redis.sock?db=0',
        'OPTIONS': {
            'CLIENT_CLASS': 'django_redis.client.DefaultClient',
            'PARSER_CLASS': 'redis.connection.HiredisParser',
        }
    },
    'avatar': {
        'BACKEND': 'django.core.cache.backends.filebased.FileBasedCache',
        'LOCATION': os.path.join(DATA_DIR, 'avatar-cache'),
        'TIMEOUT': 604800,
        'OPTIONS': {
            'MAX_ENTRIES': 1000,
        },
    }
```

См.также:

`ENABLE_AVATARS`, `AVATAR_URL_PREFIX`, *Аватары*, *Включение кэширования*, Фреймворк кэширования Django

Настройка отправки электронной почты

Weblate'у по разным случаям необходимо отправлять электронные письма и эти письма должны иметь правильный адрес отправителя, поэтому настройте параметры *SERVER_EMAIL* и *DEFAULT_FROM_EMAIL* так, чтобы они соответствовали вашему окружению, например:

```
SERVER_EMAIL = 'admin@example.org'
DEFAULT_FROM_EMAIL = 'weblate@example.org'
```

Примечание: Для отключения отправки писем Weblate'ом установите переменную EMAIL_BACKEND в значение `django.core.mail.backends.dummy.EmailBackend`.

При этом доставка сообщений электронной почты будет отключена *целиком*, включая сообщения для регистрации или сброса пароля.

См.также:

Настройка конфигурации, *Настройка исходящей почты*, EMAIL_BACKEND, DEFAULT_FROM_EMAIL, SERVER_EMAIL

Настройка разрешённых хостов

Django требует, чтобы в параметре *ALLOWED_HOSTS* содержался список доменных имён, которые разрешено обслуживать вашему сайту, если его оставить пустым, любые запросы будут заблокированы.

В случае, если он не настроен на соответствие вашему HTTP-серверу, вы будете получать ошибки вида `Invalid HTTP_HOST header: '1.1.1.1'. You may need to add '1.1.1.1' to ALLOWED_HOSTS.` (`Неверный заголовок HTTP_HOST: '1.1.1.1'. Возможно, вам нужно добавить '1.1.1.1' в ALLOWED_HOSTS.`)

Подсказка: Внутри контейнера Docker он доступен через переменную окружения *WEBLATE_ALLOWED_HOSTS*.

См.также:

ALLOWED_HOSTS, *WEBLATE_ALLOWED_HOSTS*, *Установка правильного домена сайта*

Секретный ключ Django

Параметр *SECRET_KEY* используется Django для подписания файлов кук и вы обязательно должны сгенерировать своё собственное значение, а не использовать приведённое в примере настройки.

Сгенерировать новый ключ вы можете с помощью скрипта `weblate/examples/generate-secret-key`, поставляемого вместе с Weblate.

См.также:

SECRET_KEY

Домашний каталог

Изменено в версии 2.1: Больше он не требуется, теперь Weblate хранит все свои данные в каталоге *DATA_DIR*.

Домашний каталог для пользователя, запустившего Weblate, должен существовать и быть доступен для записи из под этого пользователя. Особенно он необходим, если вы хотите использовать SSH для доступа к частным репозиториям, но доступ к этому каталогу может понадобиться и Git'у (это зависит от используемой вами версии Git'a).

Изменить используемый Weblate каталог вы можете в файле `settings.py`, например, установив его в каталог `configuration` внутри дерева Weblate:

```
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 a non-writable directory for users used for serving web content (such as `apache`, `www-data` or `wwwrun`), so you either have to run Weblate under a different user, or change this setting.

См.также:

Доступ к репозиториям

Загрузка шаблонов

Рекомендуется использовать кэширующий загрузчик шаблонов для Django. Он кэширует обработанные шаблоны, избавляя фреймворк от необходимости разбирать их каждый запрос. Настроить его вы можете с помощью следующего фрагмента кода (самое важное здесь — это настройка `loaders`):

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            os.path.join(BASE_DIR, 'templates'),
        ],
        'OPTIONS': {
            'context_processors': [
                'django.contrib.auth.context_processors.auth',
                'django.template.context_processors.debug',
                'django.template.context_processors.i18n',
                'django.template.context_processors.request',
                'django.template.context_processors.csrf',
                'django.contrib.messages.context_processors.messages',
                'weblate.trans.context_processors.weblate_context',
            ],
            'loaders': [
                ('django.template.loaders.cached.Loader', [
                    'django.template.loaders.filesystem.Loader',
                    'django.template.loaders.app_directories.Loader',
                ]),
            ],
        },
    },
]
```

См.также:

`django.template.loaders.cached.Loader`

Выполнение задач технического обслуживания

Для оптимальной производительности неплохо бы в фоновом режиме выполнять некоторые задачи по техническому обслуживанию. Теперь это делается автоматически с помощью *фоновых задач Celery*, которые включают в себя:

- Проверку работоспособности состояния конфигурации (ежечасно).
- Фиксацию отложенных изменений (ежечасно), смотрите раздел *Lazy commits* и команду `commit_pending`.
- Обновление предупреждений компонента (ежедневно).
- Обновление удалённых веток (по ночам), смотрите параметр `AUTO_UPDATE`.
- Резервное копирование памяти переводов в JSON (ежедневно), смотрите команду `dump_memory`.
- Задачи по обслуживанию полнотекстового индекса и базы данных (ежедневные и еженедельные задачи), смотрите команду `cleanuptrans`.

Изменено в версии 3.2: Начиная с версии 3.2, по умолчанию эти задачи выполняются через Celery и Weblate уже поставляется с правильной конфигурацией, смотрите раздел *Фоновые задачи с использованием Celery*.

Системные локали и кодировки

Системные локали должны быть настроены на поддерживающие UTF-8. В большинстве дистрибутивов Linux это умолчательная их настройка. В случае, если в вашей системе это не так, измените пожалуйста локали на вариант с поддержкой UTF-8.

Например, путём редактирования файла `/etc/default/locale` и прописывания в нём команды `LANG="C.UTF-8"`.

In some cases the individual services have separate configuration for locales. For example when using Apache you might want to set it in `/etc/apache2/envvars`:

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

Использование пользовательского центра сертификации

Во время HTTP-запросов Weblate проверяет SSL-сертификаты. Если вы используете пользовательский центр сертификации, который цепочке сертификатов по умолчанию не является доверенным, вам придётся добавить его сертификат в качестве доверенного.

Предпочтительный подход — сделать это на системном уровне, подробнее смотрите документацию по дистрибутиву (например, в debian это можно сделать, разместив сертификат центра сертификации в каталоге `/usr/local/share/ca-certificates/` и запустив команду **update-ca-certificates**).

Как только это будет сделано, системные инструменты станут доверять сертификату, среди этих инструментов будет и Git.

Для Python'ьего кода вам будет необходимо настроить запросы на использование системной цепочки сертификатов вместо использование цепочки, поставляемой с Python'ом. Этого можно добиться, поместив в `settings.py` следующий фрагмент кода (путь указан для Debian):

```
import os
os.environ["REQUESTS_CA_BUNDLE"] = "/etc/ssl/certs/ca-certificates.crt"
```

Сжатие клиентских активов

Weblate поставляется с кучей JavaScript и CSS файлов. Из соображений производительности хорошо бы их сжать перед отправкой клиенту. В конфигурации по умолчанию это делается на лету ценой небольших накладных расходов. В больших установках рекомендуется включить автономный режим сжатия. Это должно быть сделано в конфигурации, и сжатие должно запускаться при каждом обновлении Weblate.

Переключение конфигурации выполняется простым включением параметра `django.conf.settings.COMPRESS_OFFLINE` и настройкой параметра `django.conf.settings.COMPRESS_OFFLINE_CONTEXT` (последний уже включён в пример конфигурации):

```
COMPRESS_OFFLINE = True
```

При каждом развёртывании вам необходимо сжимать файлы, чтобы они соответствовали текущей версии:

```
weblate compress
```

Подсказка: В официальном образе Docker эта функция уже включена.

См.также:

Распространённые сценарии развёртывания, *Обслуживание статических файлов*

2.1.10 Запуск сервера

Чтобы запустить Weblate, вам понадобится несколько сервисов, рекомендуемая конфигурация состоит из:

- Сервера базы данных (смотрите раздел *Настройка базы данных для Weblate*)
- Сервера кэша (смотрите раздел *Включение кэширования*)
- Веб-сервера на клиентской части для раздачи статических файлов и SSL-терминации (смотрите раздел *Обслуживание статических файлов*)
- Сервера Wsgi для генерации динамического содержимого (смотрите раздел *Пример конфигурации для NGINX и uWSGI*)
- Celery для выполнения фоновых задач (смотрите раздел *Фоновые задачи с использованием Celery*)

Примечание: Между некоторыми сервисами существуют зависимости, например, кэш и база данных должны быть запущены до запуска процессов Celery или uwsgi.

В большинстве случаев вы будете запускать все сервисы на одном (виртуальном) сервере, но в случае, если ваша установка испытывает тяжёлые нагрузки, вы можете их разделить. Единственным ограничением является то, что серверы Celery и Wsgi должны иметь доступ к каталогу `DATA_DIR`.

Запуск веб-сервера

Запуск Weblate ничем не отличается от запуска любой другой программы на базе Django. Как правило, Django выполняется как uWSGI или fcgi (смотрите ниже примеры для различных веб-серверов).

В целях тестирования можно использовать встроенный в Django веб-сервер:

```
weblate runserver
```

Предупреждение: НЕ ИСПОЛЬЗУЙТЕ ЭТОТ СЕРВЕР В РАБОЧЕЙ СРЕДЕ. Он не проходил аудиты безопасности или тесты производительности. Также смотрите документацию Django по команде `runserver`.

Подсказка: Встроенный сервер Django обслуживает статические файлы только с включённым параметром *DEBUG*, поскольку он предназначен только для разработки. Для использования в рабочей среде смотрите установки wsgi в разделах *Пример конфигурации для NGINX и uWSGI*, *Пример конфигурации для Apache*, *Пример конфигурации для Apache и Gunicorn* и *Обслуживание статических файлов*.

Обслуживание статических файлов

Изменено в версии 2.4: До версии 2.4 Weblate некорректно использовал фреймворк статических файлов Django, поэтому его настройка была более сложной.

Django необходимо собрать свои статические файлы в одном каталоге. Для этого необходимо выполнить команду `weblate collectstatic --noinput`. Она скопирует статические файлы в каталог, указанный в параметре `STATIC_ROOT` (по умолчанию это каталог `static` внутри каталога *DATA_DIR*).

Статические файлы рекомендуется раздавать непосредственно с вашего веб-сервера, это следует делать для следующих путей:

/static/ Обслуживает статические файлы для Weblate и административного интерфейса (из каталога, определённого параметром `STATIC_ROOT`).

/media/ Используется для загрузки пользовательских мультимедийных данных (например, снимков экрана).

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

См.также:

Сжатие клиентских активов, Развёртывание Django, Развёртывание статических файлов

Политика безопасности содержимого

Конфигурация Weblate по умолчанию включает промежуточное ПО `weblate.middleware.SecurityMiddleware` , которое устанавливает связанные с безопасностью заголовки HTTP, такие как `Content-Security-Policy` или `X-XSS-Protection`. По умолчанию они настроены для работы с Weblate и его конфигурацией, но для этого может потребоваться настройка вашего окружения.

См.также:

CSP_SCRIPT_SRC, *CSP_IMG_SRC*, *CSP_CONNECT_SRC*, *CSP_STYLE_SRC*, *CSP_FONT_SRC*

Пример конфигурации для NGINX и uWSGI

Для запуска рабочего веб-сервера используйте обёртку wsgi, устанавливаемую вместе с Weblate (в виртуальном окружении она устанавливается в `~/weblate-env/lib/python3.7/site-packages/weblate/wsgi.py`). Не забудьте также поместить ваше виртуальное окружение в пути поиска Python'a (например, используя установку `virtualenv = /home/user/weblate-env` в uWSGI).

Следующая конфигурация запускает Weblate как uWSGI-приложение под веб-сервером NGINX.

Конфигурация для NGINX (пример конфигурации взят из файла `weblate/examples/weblate.nginx.conf`):

```
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
server {
    listen 80;
    server_name weblate;
    # Not used
    root /var/www/html;

    location ~ ^/favicon.ico$ {
        # DATA_DIR/static/favicon.ico
        alias /home/weblate/data/static/favicon.ico;
        expires 30d;
    }

    location /static/ {
        # DATA_DIR/static/
        alias /home/weblate/data/static/;
        expires 30d;
    }

    location /media/ {
        # DATA_DIR/media/
        alias /home/weblate/data/media/;
        expires 30d;
    }

    location / {
        include uwsgi_params;
        # Needed for long running operations in admin interface
        uwsgi_read_timeout 3600;
        # Adjust based to uwsgi configuration:
        uwsgi_pass unix:///run/uwsgi/app/weblate/socket;
        # uwsgi_pass 127.0.0.1:8080;
    }
}
```

Конфигурация для uWSGI (пример конфигурации взят из файла `weblate/examples/weblate.uwsgi.ini`):

```
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
[uwsgi]
plugins       = python3
master        = true
protocol      = uwsgi
socket        = 127.0.0.1:8080
wsgi-file     = /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/wsgi.
↪py

# Add path to Weblate checkout if you did not install
# Weblate by pip
# python-path   = /path/to/weblate

# In case you're using virtualenv uncomment this:
virtualenv = /home/weblate/weblate-env

# Needed for OAuth/OpenID
buffer-size   = 8192
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Reload when consuming too much of memory
reload-on-rss = 250

# Increase number of workers for heavily loaded sites
workers       = 8

# Enable threads for Sentry error submission
enable-threads = true

# Child processes do not need file descriptors
close-on-exec = true

# Avoid default 0000 umask
umask = 0022

# Run as weblate user
uid = weblate
gid = weblate

# Enable harakiri mode (kill requests after some time)
# harakiri = 3600
# harakiri-verbose = true

# Enable uWSGI stats server
# stats = :1717
# stats-http = true

# Do not log some errors caused by client disconnects
ignore-sigpipe = true
ignore-write-errors = true
disable-write-exception = true
```

См.также:

Как совместно использовать Django с uWSGI

Пример конфигурации для Apache

Следующая конфигурация запускает Weblate как WSGI-приложение, вам нужно включить модуль mod_wsgi (пример конфигурации взят из файла `weblate/examples/apache.conf`):

```
#
# VirtualHost for Weblate
#
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
#
<VirtualHost *:80>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        Require all granted
    </Directory>

    # DATA_DIR/media/
    Alias /media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    # Path to your Weblate virtualenv
    WSGIDaemonProcess weblate python-home=/home/weblate/weblate-env
    WSGIProcessGroup weblate
    WSGIApplicationGroup %{GLOBAL}

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

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

</VirtualHost>
```

Примечание: Weblate requires Python 3, so please make sure you are running Python 3 variant of the modwsgi. Usually it is available as a separate package, for example `libapache2-mod-wsgi-py3`.

См.также:

Системные локали и кодировки, How to use Django with Apache and mod_wsgi

Пример конфигурации для Apache и Gunicorn

Следующая конфигурация запускает Weblate в Gunicorn и Apache 2.4 (пример конфигурации взят из файла `weblate/examples/apache.gunicorn.conf`):

```
#
# VirtualHost for Weblate using gunicorn on localhost:8000
#
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
#
<VirtualHost *:443>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    # DATA_DIR/media/
    Alias /media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/https_cert.cert
    SSLCertificateKeyFile /etc/apache2/ssl/https_key.pem
    SSLProxyEngine On

    ProxyPass /favicon.ico !
    ProxyPass /static/ !
    ProxyPass /media/ !

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

См.также:

Как совместно использовать Django с Gunicorn

Запуск Weblate по пути

Изменено в версии 1.3: Поддерживается с версии Weblate 1.3.

Пример конфигурации Apache для обслуживания Weblate по пути ``/weblate`. Снова используя модуль mod_wsgi (пример конфигурации взят из файла `weblate/examples/apache-path.conf`):

```
#
# VirtualHost for Weblate, running under /weblate path
#
# This example assumes Weblate is installed in virtualenv in /home/weblate/weblate-
↪env
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your␣
↪setup.
#
<VirtualHost *:80>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /weblate/favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /weblate/static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>

    # DATA_DIR/media/
    Alias /weblate/media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    # Path to your Weblate virtualenv
    WSGIDaemonProcess weblate python-home=/home/weblate/weblate-env
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    WSGIProcessGroup weblate
    WSGIApplicationGroup %{GLOBAL}

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

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

</VirtualHost>
```

Кроме того, вам нужно будет подправить `weblate/settings.py`:

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

2.1.11 Фоновые задачи с использованием Celery

Добавлено в версии 3.2.

Для обработки фоновых задач Weblate использует Celery. Пример настроек поставляется с готовой конфигурацией, которая обрабатывает все задачи на месте, но для рабочей среды вы захотите изменить их на что-то более разумное.

Типичная установка, использующая в качестве серверной части Redis, выглядит так:

```
CELERY_TASK_ALWAYS_EAGER = False
CELERY_BROKER_URL = 'redis://localhost:6379'
CELERY_RESULT_BACKEND = CELERY_BROKER_URL
```

Также вы должны запустить рабочего Celery для обработки задач и запуска запланированных задач, это можно сделать непосредственно в командной строке (что в основном полезно при отладке или разработке):

```
./weblate/examples/celery start
./weblate/examples/celery stop
```

Запуск Celery в качестве системного сервиса

Скорее всего вы захотите запустить Celery в качестве демона, и как выполнить ваше желание описывается в разделе Демонизация документации Celery. Для наиболее распространённой установки Linux с использованием systemd вы можете воспользоваться примерами перечисленных ниже файлов, поставляемых в каталоге `examples`.

Модуль systemd, размещаемый по пути `/etc/systemd/system/celery-weblate.service`:

```
[Unit]
Description=Celery Service (Weblate)
After=network.target

[Service]
Type=forking
User=weblate
Group=weblate
EnvironmentFile=/etc/default/celery-weblate
WorkingDirectory=/home/weblate
```

(continues on next page)

(продолжение с предыдущей страницы)

```
RuntimeDirectory=celery
RuntimeDirectoryPreserve=restart
LogsDirectory=celery
ExecStart=/bin/sh -c '${CELERY_BIN} multi start ${CELERYD_NODES} \
  -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \
  --logfile=${CELERYD_LOG_FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_OPTS}'
ExecStop=/bin/sh -c '${CELERY_BIN} multi stopwait ${CELERYD_NODES} \
  --pidfile=${CELERYD_PID_FILE}'
ExecReload=/bin/sh -c '${CELERY_BIN} multi restart ${CELERYD_NODES} \
  -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \
  --logfile=${CELERYD_LOG_FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_OPTS}'

[Install]
WantedBy=multi-user.target
```

Конфигурация окружения, размещаемая по пути `/etc/default/celery-weblate`:

```
# Name of nodes to start
CELERYD_NODES="celery notify backup translate"

# Absolute or relative path to the 'celery' command:
CELERY_BIN="/home/weblate/weblate-env/bin/celery"

# App instance to use
# comment out this line if you don't use an app
CELERY_APP="weblate.utils"

# Extra command-line arguments to the worker,
# increase concurency if you get weblate.E019
CELERYD_OPTS="--beat:celery --concurrency:celery=4 --queues:celery=celery --
↪prefetch-multiplier:celery=4 \
    --concurrency:notify=4 --queues:notify=notify --prefetch-multiplier:notify=10 \
    --concurrency:translate=4 --queues:translate=translate --prefetch-
↪multiplier:translate=4 \
    --concurrency:backup=1 --queues:backup=backup  --prefetch-multiplier:backup=2"

# Logging configuration
# - %n will be replaced with the first part of the nodename.
# - %I will be replaced with the current child process index
#   and is important when using the prefork pool to avoid race conditions.
CELERYD_PID_FILE="/var/run/celery/weblate-%n.pid"
CELERYD_LOG_FILE="/var/log/celery/weblate-%n%I.log"
CELERYD_LOG_LEVEL="INFO"

# Internal Weblate variable to indicate we're running inside Celery
CELERY_WORKER_RUNNING="1"
```

Конфигурация ротации журналов, размещаемая по пути `/etc/logrotate.d/celery`:

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

Примечание: Процесс Celery должен выполняться под тем же пользователем, под которым работает Weblate и процесс WSGI, иначе файлы в каталоге *DATA_DIR* будут сохраняться с разными владельцами,

что приведёт к проблемам во время выполнения.

Периодические задачи при помощи Celery beat

Weblate поставляется со встроенной конфигурацией планировщика задач. Вы, однако, можете определить в `settings.py` дополнительные задачи, для примера смотрите раздел *Lazy commits*.

Предполагается, что задачи будут выполняться демоном Celery beat. В случае, если он не работает должным образом, он может быть не запущен или его база данных была повреждена. В этом случае проверьте журналы запуска Celery, чтобы выяснить первопричину проблемы.

Мониторинг состояния Celery

Вы можете использовать команду *`celery_queues`* для просмотра текущей длины очереди задач Celery. В случае, если очередь станет слишком длинной, вы также увидите ошибку конфигурации в интерфейсе администратора.

> **Предупреждение:** Ошибки Celery по умолчанию записываются только в журнал Celery и пользователю не видны. В случае, если вы хотите видеть сводку по таким ошибкам, рекомендуется настроить *сбор отчётов об ошибках*.

См.также:

Конфигурация и настройки по умолчанию, Руководство по рабочим, Демонизация, Руководство по мониторингу и управлению, *`celery_queues`*

2.1.12 Мониторинг Weblate

Weblate предоставляет URL-адрес `/healthz/`, который можно использоваться для простой проверки его работоспособности, например, при использовании Kubernetes.

2.1.13 Сбор отчётов об ошибках

Weblate, как и любое другое программное обеспечение, может дать сбой. Для сбора полезной информации о сбоях мы рекомендуем использовать услуги сторонних сервисов. Особенно они полезны в случае сбоя в задачах Celery, которые в противном случае просто записали бы ошибку в журнал, и вы не получили бы о ней никакого уведомления. Weblate поддерживает следующие сервисы:

Sentry

В Weblate встроена поддержка Sentry. Для его использования достаточно установить параметр *`SENTRY_DSN`* в `settings.py`:

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

Rollbar

Weblate имеет встроенную поддержку Rollbar. Для его использования достаточно следовать инструкциям из документа`Уведомления Rollbar для Python'а <https://docs.rollbar.com/docs/python/>`_.

Вкратце, вам необходимо дописать в `settings.py`:

```
# Add rollbar as last middleware:
MIDDLEWARE = [
    # … other middleware classes …
    'rollbar.contrib.django.middleware.RollbarNotifierMiddleware',
]

# Configure client access
ROLLBAR = {
    'access_token': 'POST_SERVER_ITEM_ACCESS_TOKEN',
    'client_token': 'POST_CLIENT_ITEM_ACCESS_TOKEN',
    'environment': 'development' if DEBUG else 'production',
    'branch': 'master',
    'root': '/absolute/path/to/code/root',
}
```

Всё остальное интегрируется автоматически, теперь у вас будут собираться ошибки как со стороны сервера, так и со стороны клиента.

2.1.14 Перенос Weblate на другой сервер

Перенос Weblate на другой сервер должен быть достаточно простой операцией, однако он хранит данные в нескольких местах, переносить которые следует с некоторой осторожностью. Лучше всего для переноса на время остановить Weblate.

Перенос базы данных

В зависимости от серверной части вашей базы данных, у вас может быть несколько вариантов переноса базы данных. Самый простой из них — это сделать дамп базы на старом сервере и импортировать его на новый. В качестве альтернативы вы можете использовать репликацию, если ваша база данных её поддерживает.

Наилучшим подходом будет использование родных инструментов для работы с базами данных, так как они обычно наиболее эффективны (например, **mysqldump** или **pg_dump**). Если вы выполнить перенос между разными базами данных, то единственным вариантом может оказаться использование команд управления Django для создания дампа и импорта базы данных:

```
# Export current data
weblate dumpdata > /tmp/weblate.dump
# Import dump
weblate loaddata /tmp/weblate.dump
```

Перенос репозиториев системы контроля версий

Также необходимо перенести репозитории системы контроля версий, хранящиеся в каталоге *DATA_DIR*. Вы можете просто скопировать их, либо воспользоваться командой **rsync** для более эффективного выполнения переноса.

Прочие заметки

Не забудьте переместить и другие сервисы, которые Weblate мог использовать, например, Redis, задачи Cron или модули пользовательской аутентификации.

2.2 Weblate deployments

Weblate can be easily installed in your cloud. Please find detailed guide for your platform:

- *Установка с помощью Docker'а*
- *Установка на OpenShift*

2.2.1 Helm Chart

You can install Weblate on Kubernetes using Helm. See <https://github.com/WeblateOrg/helm/tree/master/charts/weblate> for the detailed instructions.

2.2.2 Bitnami Weblate stack

Bitnami provides a Weblate stack for many platforms at <https://bitnami.com/stack/weblate>. The setup will be adjusted during installation, see <https://bitnami.com/stack/weblate/README.txt> for more documentation.

2.2.3 Weblate in YunoHost

The self-hosting project YunoHost provides a package for Weblate. Once you have your YunoHost installation, you may install Weblate as any other application. It will provide you with a fully working stack with backup and restoration, but you may still have to edit your settings file for specific usages.

You may use your administration interface, or this button (it will bring you to your server):

It also is possible to use the commandline interface:

```
yunohost app install https://github.com/YunoHost-Apps/weblate_ynh
```

2.3 Upgrading Weblate

2.3.1 Docker image upgrades

The official Docker image (see *Установка с помощью Docker'а*) has all upgrade steps integrated. There are no manual step besides pulling latest version.

2.3.2 Generic upgrade instructions

Before upgrading, please check the current *Требования к программному обеспечению* as they might have changed. Once all requirements are installed or updated, please adjust your `settings.py` to match changes in the configuration (consult `settings_example.py` for correct values).

Always check *Version specific instructions* before upgrade. In case you are skipping some versions, please follow instructions for all versions you are skipping in the upgrade. Sometimes it's better to upgrade to some intermediate version to ensure a smooth migration. Upgrading across multiple releases should work, but is not as well tested as single version upgrades.

Примечание: It is recommended to perform a full database backup prior to upgrade so that you can roll back the database in case upgrade fails, see *Резервное копирование и перенос Weblate*.

1. Stop wsgi and Celery processes. The upgrade can perform incompatible changes in the database, so it is always safer to avoid old processes running while upgrading.
2. Upgrade Weblate code.

   For pip installs it can be achieved by:

   ```
   pip install -U Weblate
   ```

   With Git checkout you need to fetch new source code and update your installation:

   ```
   cd weblate-src
   git pull
   # Update Weblate inside your virtualenv
   . ~/weblate-env/bin/pip install -e .
   # Install dependencies directly when not using virtualenv
   pip install --upgrade -r requirements.txt
   ```

3. Upgrade configuration file, refer to `settings_example.py` or *Version specific instructions* for needed steps.
4. Upgrade database structure:

   ```
   weblate migrate --noinput
   ```

5. Collect updated static files (see *Запуск сервера* and *Обслуживание статических файлов*):

   ```
   weblate collectstatic --noinput
   ```

6. Сожмите файлы JavaScript и CSS (необязательно, смотрите раздел *Сжатие клиентских активов*):

   ```
   weblate compress
   ```

7. If you are running version from Git, you should also regenerate locale files every time you are upgrading. You can do this by invoking:

   ```
   weblate compilemessages
   ```

8. Убедитесь, что ваша установка в порядке (также смотрите раздел *Рабочая среда*):

   ```
   weblate check --deploy
   ```

9. Restart celery worker (see *Фоновые задачи с использованием Celery*).

2.3.3 Version specific instructions

Upgrade from 2.x

If you are upgrading from 2.x release, always first upgrade to 3.0.1 and the continue upgrading in the 3.x series. Upgrades skipping this step are not supported and will break.

См.также:

Обновление с версии 2.20 до версии 3.0 в документации Weblate 3.0

Upgrade from 3.x

If you are upgrading from 3.x release, always first upgrade to 4.0.4 and the continue upgrading in the 4.x series. Upgrades skipping this step are not supported and will break.

См.также:

Обновление с версии 3.11 до версии 4.0 в документации Weblate 4.0

Upgrade from 4.0 to 4.1

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- There are several changes in `settings_example.py`, most notable middleware changes, please adjust your settings accordingly.
- There are new file formats, you might want to include them in case you modified the *`WEBLATE_FORMATS`*.
- There are new quality checks, you might want to include them in case you modified the *`CHECK_LIST`*.
- There is change in `DEFAULT_THROTTLE_CLASSES` setting to allow reporting of rate limiting in the API.
- There are some new and updated requirements.
- There is a change in `INSTALLED_APPS`.
- The *DeepL* machine translation now defaults to v2 API, you might need to adjust *`MT_DEEPL_API_VERSION`* in case your current DeepL subscription does not support that.

См.также:

Generic upgrade instructions

Upgrade from 4.1 to 4.2

Please follow *Generic upgrade instructions* in order to perform update.

Notable configuration or dependencies changes:

- Upgrade from 3.x releases is not longer supported, please upgrade to 4.0 or 4.1 first.
- There are some new and updated requirements.
- There are several changes in `settings_example.py`, most notable new middleware and changed application ordering.
- The keys for JSON based formats no longer include leading dot. The strings are adjusted during the database migration, but external components might need adjustment in case you rely on keys in exports or API.
- The Celery configuration was changed to no longer use `memory` queue. Please adjust your startup scripts and `CELERY_TASK_ROUTES` setting.

- The Weblate domain is now configured in the settings, see `SITE_DOMAIN` (or `WEBLATE_SITE_DOMAIN`). You will have to configure it before running Weblate.

См.также:

Generic upgrade instructions

2.3.4 Upgrading from Python 2 to Python 3

Weblate no longer supports Python older than 3.5. In case you are still running on older version, please perform migration to Python 3 first on existing version and upgrade later. See Upgrading from Python 2 to Python 3 in the Weblate 3.11.1 documentation.

2.3.5 Migrating from other databases to PostgreSQL

If you are running Weblate on other dabatase than PostgreSQL, you should migrate to PostgreSQL as that will be the only supported database backend in the 4.0 release. The following steps will guide you in migrating your data between the databases. Please remember to stop both web and Celery servers prior to the migration, otherwise you might end up with inconsistent data.

Создание базы данных в PostgreSQL

Обычно рекомендуется запускать Weblate в отдельной базе данных и под отдельной учётной записью:

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

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

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

Migrating using Django JSON dumps

The simplest approach for migration is to utilize Django JSON dumps. This works well for smaller installations. On bigger sites you might want to use pgloader instead, see *Migrating to PotsgreSQL using pgloader*.

1. Add PostgeSQL as additional database connection to the `settings.py`:

```
DATABASES = {
    'default': {
        # Database engine
        'ENGINE': 'django.db.backends.mysql',
        # Database name
        'NAME': 'weblate',
        # Database user
        'USER': 'weblate',
        # Database password
        'PASSWORD': 'password',
        # Set to empty string for localhost
        'HOST': 'database.example.com',
        # Set to empty string for default
        'PORT': '',
        # Additional database options
        'OPTIONS': {
            # In case of using an older MySQL server, which has MyISAM as a␣
↪default storage
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            # 'init_command': 'SET storage_engine=INNODB',
            # Uncomment for MySQL older than 5.7:
            # 'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
            # If your server supports it, see the Unicode issues above
            'charset': 'utf8mb4',
            # Change connection timeout in case you get MySQL gone away error:
            'connect_timeout': 28800,
        }
    },
    'postgresql': {
        # Database engine
        'ENGINE': 'django.db.backends.postgresql',
        # Database name
        'NAME': 'weblate',
        # Database user
        'USER': 'weblate',
        # Database password
        'PASSWORD': 'password',
        # Set to empty string for localhost
        'HOST': 'database.example.com',
        # Set to empty string for default
        'PORT': '',
    }
}
```

2. Run migrations and drop any data inserted into the tables:

```
weblate migrate --database=postgresql
weblate sqlflush --database=postgresql | weblate dbshell --database=postgresql
```

3. Dump legacy database and import to PostgreSQL

```
weblate dumpdata --all --output weblate.json
weblate loaddata weblate.json --database=postgresql
```

4. Adjust `DATABASES` to use just PostgreSQL database as default, remove legacy connection.

Weblate should be now ready to run from the PostgreSQL database.

Migrating to PotsgreSQL using pgloader

The pgloader is a generic migration tool to migrate data to PostgreSQL. You can use it to migrate Weblate database.

1. Adjust your `settings.py` to use PostgeSQL as a database.
2. Migrate the schema in the PostgreSQL database:

```
weblate migrate
weblate sqlflush | weblate dbshell
```

3. Run the pgloader to transfer the data. The following script can be used to migrate the database, but you might want to learn more about pgloader to understand what it does and tweak it to match your setup:

```
LOAD DATABASE
     FROM      mysql://weblate:password@localhost/weblate
     INTO postgresql://weblate:password@localhost/weblate

WITH include no drop, truncate, create no tables, create no indexes, no␣
↪foreign keys, disable triggers, reset sequences, data only
```

(continues on next page)

(продолжение с предыдущей страницы)

```
ALTER SCHEMA 'weblate' RENAME TO 'public'
;
```

2.3.6 Migrating from Pootle

As Weblate was originally written as replacement from Pootle, it is supported to migrate user accounts from Pootle. You can dump the users from Pootle and import them using `importusers`.

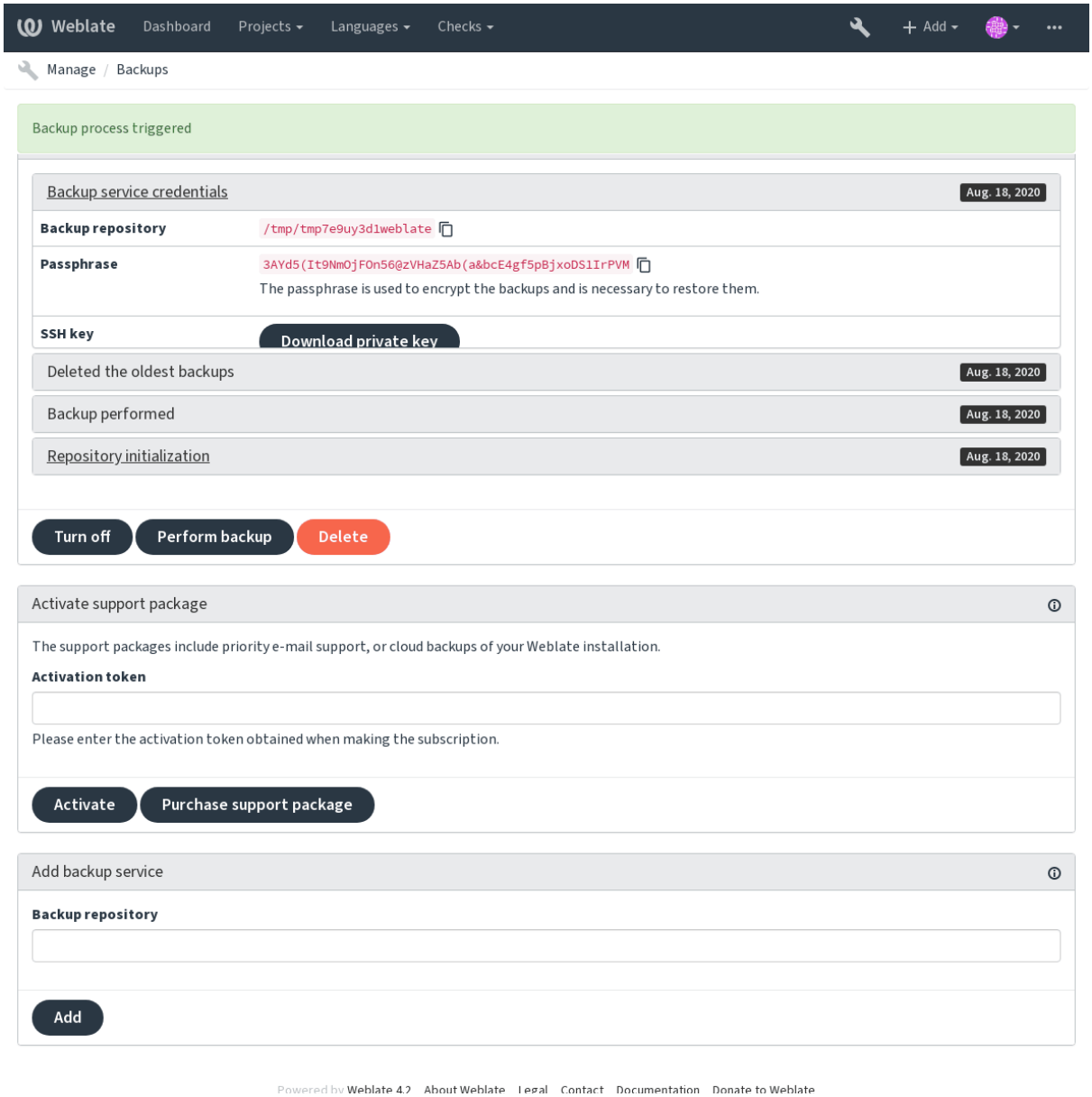
2.4 Резервное копирование и перенос Weblate

2.4.1 Автоматическое резервное копирование

Добавлено в версии 3.9.

Weblate has built-in support for creating service backups using BorgBackup. Borg creates space-effective encrypted backups which can be safely stored in the cloud. The backups can be controlled in the management interface on the *Backups* tab.

Предупреждение: Only PostgreSQL database is included in the automated backups. Other database engines have to be backed up manually. You are recommended to migrate to PostgreSQL as that will be the only supported database in the 4.0 release. See *Migrating from other databases to PostgreSQL*.



Использование хранилища резервных копий, предоставляемого Weblate

The easiest approach to backup your Weblate instance is to purchase backup service at weblate.org. The process of activating can be performed in few steps:

1. Purchase backup service on https://weblate.org/support/#backup.
2. Enter obtained key in the management interface, see *Integrating support*.
3. Weblate will connect to the cloud service and obtain access information for the backups.
4. Turn on the new backup configuration on the *Backups* tab.
5. Backup Borg credentials in order to be able to restore the backups, see *Ключ шифрования Borg*.

Подсказка: The manual step of turning on is there for your safety. Without your consent no data is sent to the backup repository obtained through the registration process.

Использование собственного хранилища резервных копий

You can also use your own storage for the backups. SSH can be used to store backups on the remote destination, the target server needs to have BorgBackup installed.

См.также:

Раздел Общие сведения в документации Borg

Ключ шифрования Borg

BorgBackup creates encrypted backups and without a passphrase you will not be able to restore the backup. The passphrase is generated when adding new backup service and you should copy it and keep it in a secure place.

In case you are using *Использование хранилища резервных копий, предоставляемого Weblate*, please backup your private SSH key as well — it is used to access your backups.

См.также:

borg init

Восстановление из резервной копии Borg

1. Восстановите доступ к своему репозиторию резервных копий и подготовьте свою парольную фразу для резервного копирования.
2. Получите список существующих на сервере резервных копий с помощью команды `borg list РЕПОЗИТОРИЙ`.
3. Восстановите нужную резервную копию в текущий каталог с помощью команды `borg extract РЕПОЗИТОРИЙ::АРХИВ`.
4. Восстановите базу данных из SQL-дампа, размещённого в каталоге `backup` каталога данных Weblate (смотрите раздел *Дампы данных для резервного копирования*).
5. Скопируйте в правильное место настройки и каталог данных Weblate.

Сеанс Borg может выглядеть так:

```
$ borg list /tmp/xxx
Enter passphrase for key /tmp/xxx:
2019-09-26T14:56:08                  Thu, 2019-09-26 14:56:08␣
↪[de0e0f13643635d5090e9896bdaceb92a023050749ad3f3350e788f1a65576a5]
$ borg extract /tmp/xxx::2019-09-26T14:56:08
Enter passphrase for key /tmp/xxx:
```

См.также:

borg list, borg extract

2.4.2 Ручное резервное копирование

Depending on what you want to save, back up the type data Weblate stores in each respective place.

Подсказка: In case you are doing manual backups, you might want to silent Weblate warning about lack of backups by adding `weblate.I028` to `SILENCED_SYSTEM_CHECKS` in `settings.py` or *`WEBLATE_SILENCED_SYSTEM_CHECKS`* for Docker.

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

База данных

The actual storage location depends on your database setup.

The database is the most important storage. Set up regular backups of your database, without it all your translation setup will be gone.

Встроенные средства базы данных для резервного копирования

The recommended approach is to do dump of the database using database native tools such as **pg_dump** or **mysqldump**. It usually performs better than Django backup and restores complete tables with all data.

You can restore this backup in newer Weblate release, it will perform any necessary migrations when running in `migrate`. Please consult *Upgrading Weblate* on more detailed information how to perform upgrade between versions.

Резервное копирование базы данных средствами Django

Alternatively you can backup database using Django's `dumpdata` command. That way the backup is database agnostic and can be used in case you want to change database backend.

Prior to restoring you need to be running exactly same Weblate version as was used when doing backups. This is necessary as the database structure does change between releases and you would end up corrupting the data in some way. After installing the same version, run all database migrations using `migrate`.

Once this is done, some entries will be already created in the database and you will have them in the database backup as well. The recommended approach is to delete such entries manually using management shell (see *Вызов команд управления*):

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

Файлы

If you have enough backup space, simply backup the whole *DATA_DIR*. This is safe bet even if it includes some files you don't want. The following sections describe in detail what you should back up and what you can skip.

Дампы данных для резервного копирования

Хранятся в *DATA_DIR* `/backups`.

Weblate dumps various data here, and you can include these files for more complete backups. The files are updated daily (requires a running Celery beats server, see *Фоновые задачи с использованием Celery*). Currently, this includes:

- Weblate settings as `settings.py` (there is also expanded version in `settings-expanded.py`).
- PostgreSQL database backup as `database.sql`.

The database backups are by default saved as plain text, but they can also be compressed or entirely skipped by using *DATABASE_BACKUP*.

Репозитории системы контроля версий

Хранятся в *DATA_DIR* /vcs.

The version control repositories contain a copy of your upstream repositories with Weblate changes. If you have push on commit enabled for all your translation components, all Weblate changes are included upstream and you do not have to backup the repositories on the Weblate side. They can be cloned again from the upstream locations with no data loss.

Ключи SSH и GPG

Хранятся в *DATA_DIR* /ssh и *DATA_DIR* /home.

If you are using SSH or GPG keys generated by Weblate, you should back up these locations, otherwise you will lose the private keys and you will have to regenerate new ones.

Загруженные пользователем файлы

Хранятся в *DATA_DIR* /media.

You should back up user uploaded files (e.g. *Visual context for strings*).

Командная строка для ручного резервного копирования

Using a cron task, you can set up a bash command to be executed on a daily basis, for instance:

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

The string between quotes after XZ_OPT allows you to choose your xz options, for instance the amount of memory used for compression; see https://linux.die.net/man/1/xz

You can adjust the list of folders and files to your needs. For instance, to avoid saving the translation memory (in backups folder), you could use:

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

2.4.3 Задачи Celery

The Celery tasks queue might contain some info, but is usually not needed for a backup. At most you will lose updates that have not yet been processed to translation memory. It is recommended to perform the fulltext or repository updates upon restoring anyhow, so there is no problem in losing these.

См.также:

Фоновые задачи с использованием Celery

2.4.4 Восстановление ручных резервных копий

1. Восстановите все данные, сохранённые вами в резервной копии.
2. Обновите все репозитории с помощью команды `updategit`.

```
weblate updategit --all
```

2.4.5 Перенос установки Weblate

Переместите свою установку на другую систему, следуя приведённым выше инструкциям по резервному копированию и восстановлению из резервной копии.

См.также:

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

2.5 Аутентификация

2.5.1 Регистрация пользователя

The default setup for Weblate is to use python-social-auth, a form on the website to handle registration of new users. After confirming their e-mail a new user can contribute or authenticate by using one of the third party services.

You can also turn off registration of new users using `REGISTRATION_OPEN`.

The authentication attempts are subject to *Ограничение частоты запросов*.

2.5.2 Authentication backends

The built-in solution of Django is used for authentication, including various social options to do so. Using it means you can import the user database of other Django-based projects (see *Migrating from Pootle*).

Django can additionally be set up to authenticate against other means too.

См.также:

Параметры аутентификации describes how to configure authentication in the official Docker image.

2.5.3 Социальная аутентификация

Thanks to Welcome to Python Social Auth's documentation!, Weblate support authentication using many third party services such as GitLab, Ubuntu, Fedora, etc.

Please check their documentation for generic configuration instructions in Django Framework.

Примечание: By default, Weblate relies on third-party authentication services to provide a validated e-mail address. If some of the services you want to use don't support this, please enforce e-mail validation on the Weblate side by configuring FORCE_EMAIL_VALIDATION for them. For example:

```
SOCIAL_AUTH_OPENSUSE_FORCE_EMAIL_VALIDATION = True
```

См.также:

Конвейер

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

См.также:

Python Social Auth backend

Аутентификация OpenID

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

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

См.также:

OpenID

Аутентификация GitHub

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

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.github.GithubOAuth2',
    'social_core.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
SOCIAL_AUTH_GITHUB_KEY = 'GitHub Client ID'
SOCIAL_AUTH_GITHUB_SECRET = 'GitHub Client Secret'
SOCIAL_AUTH_GITHUB_SCOPE = ['user:email']
```

The GitHub should be configured to have callback URL as `https://example.com/accounts/complete/github/`.

Примечание: Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see *Установка правильного домена сайта*.

См.также:

GitHub

Аутентификация Bitbucket

You need to register an application on Bitbucket and then tell Weblate all its secrets:

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.bitbucket.BitbucketOAuth',
    'social_core.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
SOCIAL_AUTH_BITBUCKET_KEY = 'Bitbucket Client ID'
SOCIAL_AUTH_BITBUCKET_SECRET = 'Bitbucket Client Secret'
SOCIAL_AUTH_BITBUCKET_VERIFIED_EMAILS_ONLY = True
```

Примечание: Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see *Установка правильного домена сайта*.

См.также:

Bitbucket

Google OAuth 2

To use Google OAuth 2, you need to register an application on <https://console.developers.google.com/> and enable the Google+ API.

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

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.google.GoogleOAuth2',
    'social_core.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = 'Client ID'
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = 'Client secret'
```

Примечание: Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see *Установка правильного домена сайта*.

См.также:

Google

Facebook OAuth 2

As per usual with OAuth 2 services, you need to register your application with Facebook. Once this is done, you can set up Weblate to use it:

The redirect URL is `https://WEBLATE SERVER/accounts/complete/facebook/`

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

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

Примечание: Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see *Установка правильного домена сайта*.

См.также:

Facebook

GitLab OAuth 2

For using GitLab OAuth 2, you need to register an application on <https://gitlab.com/profile/applications>.

The redirect URL is `https://WEBLATE SERVER/accounts/complete/gitlab/` and ensure you mark the *read_user* scope.

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.gitlab.GitLabOAuth2',
    'social_core.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
SOCIAL_AUTH_GITLAB_KEY = 'Application ID'
SOCIAL_AUTH_GITLAB_SECRET = 'Secret'
SOCIAL_AUTH_GITLAB_SCOPE = ['read_user']

# If you are using your own GitLab
# SOCIAL_AUTH_GITLAB_API_URL = 'https://gitlab.example.com/'
```

Примечание: Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see *Установка правильного домена сайта*.

См.также:

GitLab

Microsoft Azure Active Directory

Weblate can be configured to use common or specific tenants for authentication.

The redirect URL is `https://WEBLATE SERVER/accounts/complete/azuread-oauth2/` for common and `https://WEBLATE SERVER/accounts/complete/azuread-tenant-oauth2/` for tenant-specific authentication.

```
# Azure AD common

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.azuread.AzureADOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# OAuth2 keys
SOCIAL_AUTH_AZUREAD_OAUTH2_KEY = ""
SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET = ""
```

```
# Azure AD Tenant

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.azuread_tenant.AzureADTenantOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# OAuth2 keys
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY = ""
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECRET = ""
# Tenant ID
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENANT_ID = ""
```

Примечание: Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see *Установка правильного домена сайта*.

См.также:

Microsoft Azure Active Directory

Slack

For using Slack OAuth 2, you need to register an application on <https://api.slack.com/apps>.

The redirect URL is `https://WEBLATE SERVER/accounts/complete/slack/`.

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    'social_core.backends.slack.SlackOAuth2',
    'social_core.backends.email.EmailAuth',
    'weblate.accounts.auth.WeblateUserBackend',
)

# Social auth backends setup
SOCIAL_AUTH_SLACK_KEY = ''
SOCIAL_AUTH_SLACK_SECRET = ''
```

Примечание: Weblate provided callback URL during the authentication includes configured domain. In case you get errors about URL mismatch, you might want to fix this, see *Установка правильного домена сайта*.

См.также:

Slack

Отключение аутентификации по паролю

E-mail and password authentication can be turned off by removing `social_core.backends.email.EmailAuth` from `AUTHENTICATION_BACKENDS`. Always keep `weblate.accounts.auth.WeblateUserBackend` there, it is needed for core Weblate functionality.

Совет: You can still use password authentication for the admin interface, for users you manually create there. Just navigate to `/admin/`.

For example authentication using only the openSUSE Open ID provider can be achieved using the following:

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

2.5.4 Аутентификация по паролю

The default `settings.py` comes with a reasonable set of `AUTH_PASSWORD_VALIDATORS`:

- Passwords can't be too similar to your other personal info.
- Passwords must contain at least 10 characters.
- Passwords can't be a commonly used password.
- Passwords can't be entirely numeric.
- Passwords can't consist of a single character or only whitespace.
- Passwords can't match a password you have used in the past.

You can customize this setting to match your password policy.

Additionally you can also install django-zxcvbn-password which gives quite realistic estimates of password difficulty and allows rejecting passwords below a certain threshold.

2.5.5 Аутентификация через SAML

Добавлено в версии 4.1.1.

Please follow the Python Social Auth instructions for configuration. Notable differences:

- Weblate supports single IDP which has to be called `weblate` in `SOCIAL_AUTH_SAML_ENABLED_IDPS`.
- The SAML XML metadata URL is `/accounts/metadata/saml/`.
- Following settings are automatically filled in: `SOCIAL_AUTH_SAML_SP_ENTITY_ID`, `SOCIAL_AUTH_SAML_TECHNICAL_CONTACT`, `SOCIAL_AUTH_SAML_SUPPORT_CONTACT`

Example configuration:

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

# Social auth backends setup
SOCIAL_AUTH_SAML_SP_PUBLIC_CERT = "-----BEGIN CERTIFICATE-----"
SOCIAL_AUTH_SAML_SP_PRIVATE_KEY = "-----BEGIN PRIVATE KEY-----"
SOCIAL_AUTH_SAML_ENABLED_IDPS = {
    "weblate": {
        "entity_id": "https://idp.testshib.org/idp/shibboleth",
        "url": "https://idp.testshib.org/idp/profile/SAML2/Redirect/SSO",
        "x509cert": "MIIEDjCCAvagAwIBAgIBADA ... 8Bbnl+ev0peYzxFyF5sQA==",
        "attr_name": "full_name",
        "attr_username": "username",
        "attr_email": "email",
    }
}
```

См.также:

Настройка SAML в Docker'е, SAML

2.5.6 Аутентификация через LDAP

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

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

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

Предупреждение: With django-auth-ldap older than 1.3.0 the *Автоматические прикрепления группы* will not work properly for newly created users.

Примечание: There are some incompatibilities in the Python LDAP 3.1.0 module, which might prevent you from using that version. If you get error AttributeError: „module“ object has no attribute „_trace_level“, downgrading python-ldap to 3.0.0 might help.

Once you have the package installed, you can hook it into the Django authentication:

```
# Add LDAP backed, keep Django one if you want to be able to login
# even without LDAP for admin account
AUTHENTICATION_BACKENDS = (
    'django_auth_ldap.backend.LDAPBackend',
    'weblate.accounts.auth.WeblateUserBackend',
)

# LDAP server address
AUTH_LDAP_SERVER_URI = 'ldaps://ldap.example.net'

# DN to use for authentication
AUTH_LDAP_USER_DN_TEMPLATE = 'cn=%(user)s,o=Example'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Depending on your LDAP server, you might use a different DN
# like:
# AUTH_LDAP_USER_DN_TEMPLATE = 'ou=users,dc=example,dc=com'

# List of attributes to import from LDAP upon login
# Weblate stores full name of the user in the full_name attribute
AUTH_LDAP_USER_ATTR_MAP = {
    'full_name': 'name',
    # Use the following if your LDAP server does not have full name
    # Weblate will merge them later
    # 'first_name': 'givenName',
    # 'last_name': 'sn',
    # Email is required for Weblate (used in VCS commits)
    'email': 'mail',
}

# Hide the registration form
REGISTRATION_OPEN = False
```

Примечание: You should remove `'social_core.backends.email.EmailAuth'` from the `AUTHENTICATION_BACKENDS` setting, otherwise users will be able to set their password in Weblate, and authenticate using that. Keeping `'weblate.accounts.auth.WeblateUserBackend'` is still needed in order to make permissions and facilitate anonymous users. It will also allow you to sign in using a local admin account, if you have created it (e.g. by using `createadmin`).

Using bind password

If you can not use direct bind for authentication, you will need to use search, and provide a user to bind for the search. For example:

```
import ldap
from django_auth_ldap.config import LDAPSearch

AUTH_LDAP_BIND_DN = ""
AUTH_LDAP_BIND_PASSWORD = ""
AUTH_LDAP_USER_SEARCH = LDAPSearch("ou=users,dc=example,dc=com",
    ldap.SCOPE_SUBTREE, "(uid=%(user)s)")
```

Интеграция с Active Directory

```
import ldap
from django_auth_ldap.config import LDAPSearch, NestedActiveDirectoryGroupType

AUTH_LDAP_BIND_DN = "CN=ldap,CN=Users,DC=example,DC=com"
AUTH_LDAP_BIND_PASSWORD = "password"

# User and group search objects and types
AUTH_LDAP_USER_SEARCH = LDAPSearch("CN=Users,DC=example,DC=com", ldap.SCOPE_
↪SUBTREE, "(sAMAccountName=%(user)s)")

# Make selected group a superuser in Weblate
AUTH_LDAP_USER_FLAGS_BY_GROUP = {
    # is_superuser means user has all permissions
    "is_superuser": "CN=weblate_AdminUsers,OU=Groups,DC=example,DC=com",
}
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Map groups from AD to Weblate
AUTH_LDAP_GROUP_SEARCH = LDAPSearch("OU=Groups,DC=example,DC=com", ldap.SCOPE_
↪SUBTREE, "(objectClass=group)")
AUTH_LDAP_GROUP_TYPE = NestedActiveDirectoryGroupType()
AUTH_LDAP_FIND_GROUP_PERMS = True

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

См.также:

Аутентификация в Django через LDAP, Аутентификация

2.5.7 Аутентификация через CAS

CAS authentication can be achieved using a package such as *django-cas-ng*.

Step one is disclosing the e-mail field of the user via CAS. This has to be configured on the CAS server itself, and requires you run at least CAS v2 since CAS v1 doesn't support attributes at all.

Step two is updating Weblate to use your CAS server and attributes.

To install *django-cas-ng*:

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

Once you have the package installed you can hook it up to the Django authentication system by modifying the `settings.py` file:

```
# Add CAS backed, keep the Django one if you want to be able to sign in
# even without LDAP for the admin account
AUTHENTICATION_BACKENDS = (
    'django_cas_ng.backends.CASBackend',
    'weblate.accounts.auth.WeblateUserBackend',
)

# CAS server address
CAS_SERVER_URL = 'https://cas.example.net/cas/'

# Add django_cas_ng somewhere in the list of INSTALLED_APPS
INSTALLED_APPS = (
    ...,
    'django_cas_ng'
)
```

Finally, a signal can be used to map the e-mail field to the user object. For this to work you have to import the signal from the *django-cas-ng* package and connect your code with this signal. Doing this in settings file can cause problems, therefore it's suggested to put it:

- In your app config's `django.apps.AppConfig.ready()` method
- In the project's `urls.py` file (when no models exist)

```
from django_cas_ng.signals import cas_user_authenticated
from django.dispatch import receiver
@receiver(cas_user_authenticated)
def update_user_email_address(sender, user=None, attributes=None, **kwargs):
    # If your CAS server does not always include the email attribute
    # you can wrap the next two lines of code in a try/catch block.
    user.email = attributes['email']
    user.save()
```

См.также:

Django CAS NG

2.5.8 Настройка сторонней аутентификации Django

Generally any Django authentication plugin should work with Weblate. Just follow the instructions for the plugin, just remember to keep the Weblate user backend installed.

См.также:

Аутентификация через LDAP, *Аутентификация через CAS*

Typically the installation will consist of adding an authentication backend to `AUTHENTICATION_BACKENDS` and installing an authentication app (if there is any) into `INSTALLED_APPS`:

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

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

2.6 Управление доступом

Изменено в версии 3.0: До версии 3.0 система привилегий Weblate была основана на системе привилегий Django, однако теперь она построена специально для Weblate. Если вы используете более старую версию, пожалуйста, ознакомьтесь с документацией к этой старой версии, информация, данная здесь, не будет к ней применима.

Weblate поставляется с тонко настраиваемой системой привилегий для назначения пользователям прав как в рамках всего экземпляра приложения, так и в рамках отдельных его частей.

Система разрешений построена на основе групп и ролей, в которой роли определяют набор разрешений, а группы назначают их пользователям и переводчикам, для получения более подробной информации смотрите раздел *Пользователи, роли, группы и разрешения*.

После установки создаётся стандартный набор групп; вы можете использовать их, чтобы назначить пользователям роли в рамках всего экземпляра приложения (смотрите раздел *Группы и роли по умолчанию*). Кроме того, если включён *По-проектный контроль доступа*, вы можете назначать пользователей конкретным проектам перевода. Более мелкозернистую конфигурацию поможет обеспечить *Настраиваемый контроль доступа*

2.6.1 Распространённые настройки

Блокировка Weblate

Для полной блокировки вашей установки Weblate вы можете использовать настройку `LOGIN_REQUIRED_URLS`, чтобы заставить пользователей входить в систему и настройку `REGISTRATION_OPEN` для предотвращения новых регистраций.

Разрешения для всего сайта

Для управления разрешениями для всего экземпляра просто добавьте пользователей в группы *Пользователи* (это уже делается по умолчанию механизмом *автоматического назначения групп*), *Рецензенты* и *Управляющие*. Оставляйте все проекты в статусе *Публичный* (смотрите раздел *По-проектный контроль доступа*).

По-проектные разрешения

Переведите ваши проекты в статусы *Защищенный* или *Частный* и управляйте пользователями отдельно для каждого проекта через интерфейс Weblate.

Добавление разрешений к языкам, компонентам или проектам

Дополнительно вы можете предоставить разрешения любому пользователю на основе проекта, компонента или набора языков. Для этого создайте новую группу (например, *Переводчики на чешский*) и сконфигурируйте её для данного ресурса. Членам этой группы будут предоставлены любые назначенные разрешения для выбранных ресурсов.

Это будет отлично работать без дополнительной настройки, если использовать по-проектные разрешения. Для разрешений на весь экземпляр вы, вероятно, захотите удалить эти разрешения из группы *Пользователи* или изменить автоматическое назначение всем пользователям этой группы (смотрите раздел *Автоматические прикрепления группы*).

См.также:

Проверка разрешений

2.6.2 По-проектный контроль доступа

Примечание: При включении списков контроля доступа всем пользователям запрещается доступ к чему бы то ни было в пределах данного проекта, если только вы не добавите для них соответствующие разрешения.

Вы можете ограничить доступ пользователей к отдельным проектам. Эта функция включается в блоке *Управление доступом* конфигурации каждого соответствующего проекта. Она автоматически создаёт для этого проекта несколько групп, смотрите раздел *Предопределённые группы*.

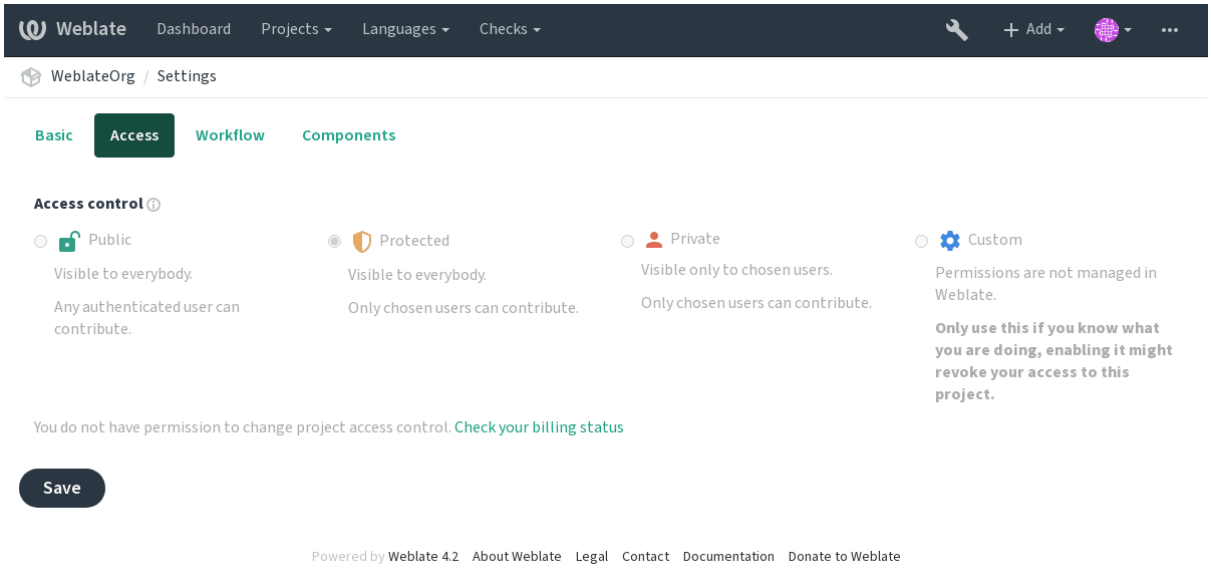
Для настройки :guilabel: *Управление доступом* возможны следующие варианты выбора:

Публичный Проект видят и могут переводить все

Защищённый Проект видят все, но переводить могут только выбранные пользователи

Частный Видеть и переводить проект могут только выбранные пользователи

Настраиваемый Weblate не занимается управлением пользователями, смотрите раздел *Настраиваемый контроль доступа*.



Чтобы разрешить доступ к этому проекту, вы должны добавить привилегию либо непосредственно данному пользователю, либо группе пользователей в административном интерфейсе Django, либо с помощью системы управления пользователями на странице проекта, как описано в разделе *По-проектная настройка доступа*.

Примечание: Даже при включённом списке контроля доступа о вашем проекте будет доступна следующая краткая информация:

- Статистика для всего экземпляра, в том числе по каждому проекту в отдельности.
- Сводка по языкам для всего экземпляра, в том числе по каждому проекту в отдельности.

2.6.3 Автоматические прикрепления группы

Вы можете настроить Weblate для автоматического добавления пользователей в группы на основе их адреса электронной почты. Это автоматическое назначение происходит только в момент создания учётной записи.

Оно может быть настроено в административном интерфейсе Django отдельно для каждой группы (в разделе *Аутентификация*).

Примечание: Weblate при миграциях всегда настраивает автоматическое назначение групп *Пользователи* и *Просматривающие*. В случае, если вы хотите его отключить, просто задайте для них регулярное выражение `^$`, которое никогда ничего не найдёт.

2.6.4 Пользователи, роли, группы и разрешения

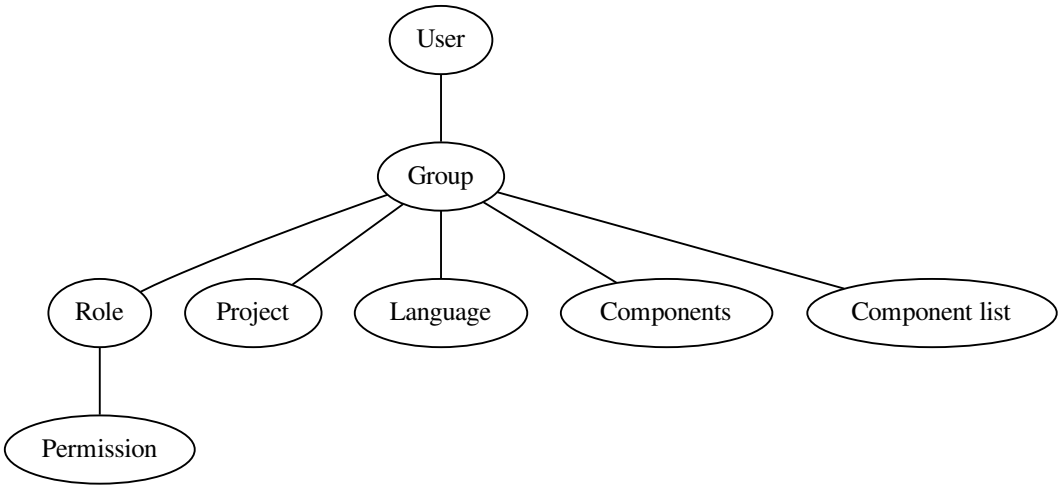
Модели аутентификации состоят из нескольких объектов:

Разрешение Индивидуальные разрешения, определённые Weblate. Вы не можете назначать отдельные разрешения, назначить их можно только через назначение ролей.

Роль Роль определяет набор разрешений. Она позволяет повторно использовать эти наборы в нескольких местах, и упрощает их администрирование.

Пользователь Пользователи могут состоять в нескольких группах.

Группа Группы соединяют роли, пользователей и объекты аутентификации (проекты, языки и списки компонентов).



Проверка разрешений

Всякий раз, когда происходит проверка разрешения для выяснения того, может ли пользователь выполнить данное действие, она производится в соответствующей области, и проверяет следующие вещи в указанном порядке:

1. С компонентом или проектом сравнивается *Список компонентов* в поиске соответствия.
2. С компонентом или проектом сравниваются *Компоненты* в поиске соответствия.
3. С проектом сравниваются *Проекты* в поиске соответствия.

Как вы можете видеть, предоставление пользователю доступа к компоненту автоматически предоставляет ему доступ и к содержащему компонент проекту.

Примечание: Используется только первое сработавшее правило. Таким образом, если задать *Список компонентов*, *Компоненты* и *Проект*, то применится только *Список компонентов*.

Если производится проверка разрешения на перевод, выполняется дополнительный шаг:

4. *Languages* are matched against the scope of translations if set, if not set, this does not match any language.

Подсказка: Для автоматического включения всех языков или проектов вы можете использовать переключатели *Выбор языка* или *Выбор проекта*.

Проверка доступа к проекту

Пользователь должен быть членом группы, связанной с проектом или с каким-либо компонентом внутри проекта. Чтобы получить доступ к проекту, никаких специальных разрешений не требуется, достаточно только членства в группе (это используется в умолчательной группе *Просматривающие*, смотрите раздел *Группы и роли по умолчанию*).

Проверка доступа к компоненту

Пользователь может получить доступ к неограниченному компоненту, как только получит доступ к содержащему компонент проекту. Если включён *Restricted access*, для доступа к компоненту потребуется явное разрешение на компонент (или на содержащий его список компонентов).

2.6.5 Управление пользователями и группами

Пользователями и группами можно управлять через административный интерфейс Django, доступный по адресу `/admin/`.

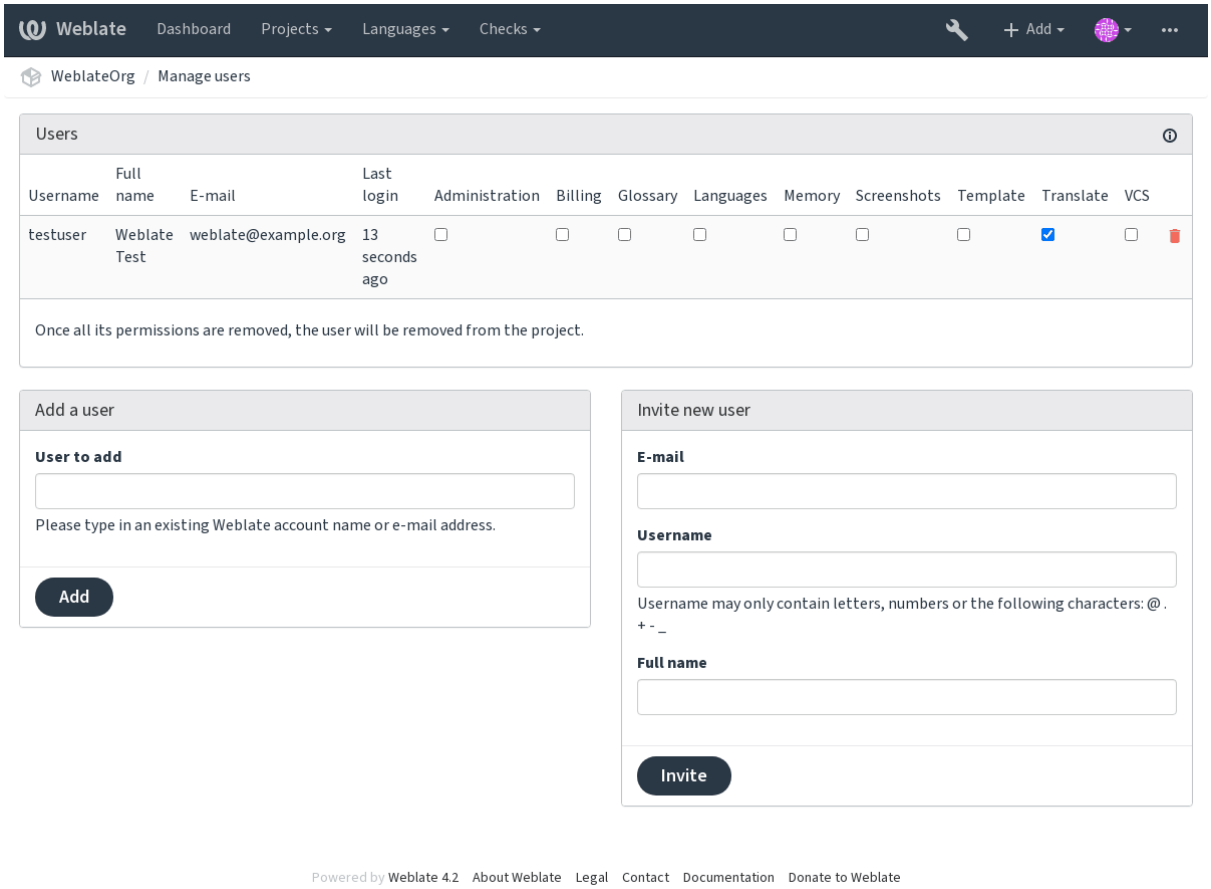
По-проектная настройка доступа

Примечание: Эта возможность работает только для проектов под управлением списка контроля доступа, смотрите раздел *По-проектный контроль доступа*.

Пользователи с привилегией *Может управлять списками контроля доступа для проекта* (смотрите раздел *Управление доступом*) также могут управлять пользователями в проектах с включённым на странице проекта контролем доступа. Интерфейс позволяет вам:

- Добавлять существующих пользователей в проект
- Приглашать новых пользователей в проект
- Изменять разрешения пользователей
- Отзывать доступы пользователей

Управление пользователями доступно через меню *Управлять* проекта:



См.также:

По-проектный контроль доступа

Предопределённые группы

Weblate поставляется с предопределённым набором групп для проекта, которые вы можете назначить пользователям.

`Administration`
Даёт все разрешения, доступные в проекте.

`Glossary`
Может управлять глоссарием (добавлять, удалять или загружать записи в глоссарий).

`Languages`
Может управлять языками перевода — добавлять или удалять переводы.

`Screenshots`
Может управлять снимками экрана — добавлять или удалять их, а также связывать их с исходными строками.

`Template`
Может редактировать шаблоны перевода в *одноязычных компонентах* и информацию об исходной строке.

`Translate`
Может переводить проект и загружать переводы, сделанные в автономном режиме.

`VCS`
Может управлять системой контроля версий и имеет доступ к экспортированному репозиторию.

Review
Может одобрять переводы во время рецензирования.

Billing
Имеет доступ к платёжной информации (смотрите раздел *Биллинг*).

2.6.6 Настраиваемый контроль доступа

При выборе в графе *Управление доступом* пункта *Настраиваемый*, Weblate прекратит управление доступом для данного проекта и всеми пользователями и группами можно будет управлять с помощью административного интерфейса Django. Этот пункт можно использовать для определения более сложного контроля доступа, либо для настройки политики общего доступа для всех проектов на одном экземпляре Weblate. Если вы хотите включить его для всех проектов по умолчанию, пожалуйста, задайте параметр `DEFAULT_ACCESS_CONTROL`.

> **Предупреждение:** При включении этого режима Weblate удалит все *по-проектные списки контроля доступа*, которые он создал для этого проекта. Если вы включаете его, не имея прав администратора, полученных от экземпляра приложения, вы мгновенно потеряете доступ к управлению проектом.

2.6.7 Группы и роли по умолчанию

Список привилегий

Биллинг (смотрите раздел *Биллинг*) Просмотр платёжной информации [*Администрирование*, *Биллинг*]

Изменения Скачивание изменений [*Администрирование*]

Комментарии Публикация комментариев [*Администрирование*, *Правка источника*, *Опытный пользователь*, *Рецензирование строк*, *Перевод*]

Удаление комментариев [*Администрирование*]

Компонент Правка параметров компонентов [*Администрирование*]

Блокировка компонента от перевода [*Администрирование*]

Глоссарий Добавление записей в глоссарий [*Администрирование*, *Управление глоссарием*, *Опытный пользователь*]

Правка записей глоссария [*Администрирование*, *Управление глоссарием*, *Опытный пользователь*]

Удаление записей из глоссария [*Администрирование*, *Управление глоссарием*, *Опытный пользователь*]

Загрузка записей в глоссарий [*Администрирование*, *Управление глоссарием*, *Опытный пользователь*]

Автоматизированный перевод Использование автоматизированного перевода [*Администрирование*, *Опытный пользователь*]

Проекты Правка параметров проекта [*Администрирование*]

Управление доступом к проекту [*Администрирование*]

Отчёты Скачивание отчётов [*Администрирование*]

Снимки экрана Добавление снимков экрана [*Администрирование*, *Управление снимками экрана*]

Правка снимков экрана [*Администрирование*, *Управление снимками экрана*]

Удаление снимков экрана [*Администрирование*, *Управление снимками экрана*]

Исходные строки Правка исходных строк [*Администрирование*, *Правка источника*]

Строки Добавление новых строк [*Администрирование*]

Игнорирование неудачных проверок [*Администрирование*, *Правка источника*, *Опытный пользователь*, *Рецензирование строк*, *Перевод*]

Правка строк [*Администрирование*, *Правка источника*, *Опытный пользователь*, *Рецензирование строк*, *Перевод*]

Рецензирование строк [*Администрирование*, *Рецензирование строк*]

Правка строк, когда предложения являются обязательными [*Администрирование*, *Рецензирование строк*]

Правка исходных строк [*Администрирование*, *Правка источника*, *Опытный пользователь*]

Предложения Принятие предложения [*Администрирование*, *Правка источника*, *Опытный пользователь*, *Рецензирование строк*, *Перевод*]

Добавление предложений [*Добавление предложений*, *Администрирование*, *Правка источника*, *Опытный пользователь*, *Рецензирование строк*, *Перевод*]

Удаление предложений [*Администрирование*]

Голосование по предложениям [*Администрирование*, *Правка источника*, *Опытный пользователь*, *Рецензирование строк*, *Перевод*]

Переводы Начало нового перевода [*Администрирование*, *Управление языками*, *Опытный пользователь*]

Производство автоматического перевода [*Администрирование*, *Управление языками*]

Удаление существующих переводов [*Администрирование*, *Управление языками*]

Начало перевода на новый язык [*Администрирование*, *Управление языками*]

Загрузки Задание автора загружаемого перевода [*Администрирование*]

Перезапись существующих строк при загрузке новых [*Администрирование*, *Правка источника*, *Опытный пользователь*, *Рецензирование строк*, *Перевод*]

Загрузка переведённых строк [*Администрирование*, *Правка источника*, *Опытный пользователь*, *Рецензирование строк*, *Перевод*]

Система контроля версий (VCS) Доступ к внутреннему репозиторию [*Доступ к репозиторию*, *Администрирование*, *Управление репозиторием*, *Опытный пользователь*]

Коммит изменений во внутренний репозиторий [*Администрирование*, *Управление репозиторием*]

Отправка изменений из внутреннего репозитория [*Администрирование*, *Управление репозиторием*]

Сброс изменений во внутреннем репозитории [*Администрирование*, *Управление репозиторием*]

Просмотр расположения вышестоящего репозитория [*Доступ к репозиторию*, *Администрирование*, *Управление репозиторием*, *Опытный пользователь*]

Обновление внутреннего репозитория [*Администрирование*, *Управление репозиторием*]

Привилегии для всего сайта Использование интерфейса управления

Добавление определений языка

Управление определениями языка

Добавление групп

Управление группами

Добавление пользователей

Управление пользователями

Управление объявлениями

Управление памятью переводов

Примечание: Привилегии для всего сайта не предоставляются никаким умолчательным ролям. Они мощны и достаточно близки к статусу суперпользователя — большинство из них влияют на все проекты вашей установки Weblate.

Список групп

Следующие группы создаются после установки (или после выполнения команды `setupgroups`):

Гости Определяет разрешения для неавторизованных пользователей.

Эта группа содержит только анонимных пользователей (смотрите настройку `ANONYMOUS_USER_NAME`).

Вы можете удалить роли из этой группы, чтобы ограничить разрешения для неавторизованных пользователей.

Роли по умолчанию: *Добавление предложений*, *Доступ к репозиторию*

Просматривающие Эта роль обеспечивает видимость публичных проектов для всех пользователей. По умолчанию членами этой группы являются все пользователи.

По умолчанию все пользователи являются членами этой группы, используя *Автоматические прикрепления группы*.

Роли по умолчанию: нет

Пользователи Стандартная группа для всех пользователей.

По умолчанию все пользователи являются членами этой группы, используя *Автоматические прикрепления группы*.

Роли по умолчанию: *Опытный пользователь*

Рецензенты Группа для рецензентов (смотрите раздел *Рабочие процессы перевода*).

Роли по умолчанию: *Рецензирование строк*

Управляющие Группа для администраторов.

Роли по умолчанию: *Администрирование*

Предупреждение: Никогда не удаляйте предопределённые в Weblate группы и пользователей, поскольку это может привести к неожиданным проблемам. Если вы не хотите использовать эти функции, просто удалите из них все привилегии.

2.7 Переводческие проекты

2.7.1 Translation organization

Weblate organizes translatable VCS content of project/components into a tree-like structure.

- The bottom level object is *Project configuration*, which should hold all translations belonging together (for example translation of an application in several versions and/or accompanying documentation).
- On the level above, *Конфигурация компонента*, which is actually the component to translate, you define the VCS repository to use, and the mask of files to translate.
- Above *Конфигурация компонента* there are individual translations, handled automatically by Weblate as translation files (which match the mask defined in *Конфигурация компонента*) appear in the VCS repository.

Weblate supports a wide range of translation formats (both bilingual and monolingual ones) supported by Translate Toolkit, see *Поддерживаемые форматы файлов*.

Примечание: You can share cloned VCS repositories using *Внутренние URL-адреса Weblate*. Using this feature is highly recommended when you have many components sharing the same VCS. It improves performance and decreases required disk space.

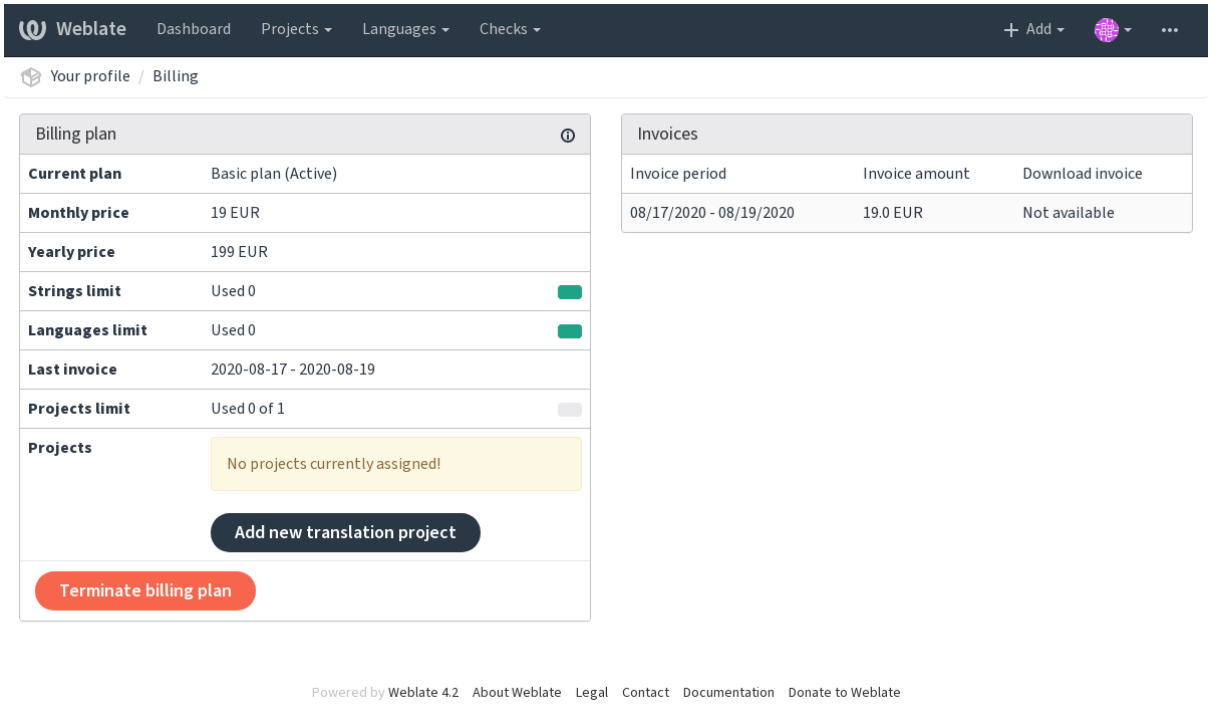
2.7.2 Adding translation projects and components

Изменено в версии 3.2: An interface for adding projects and components is included, and you no longer have to use *The Django admin interface*.

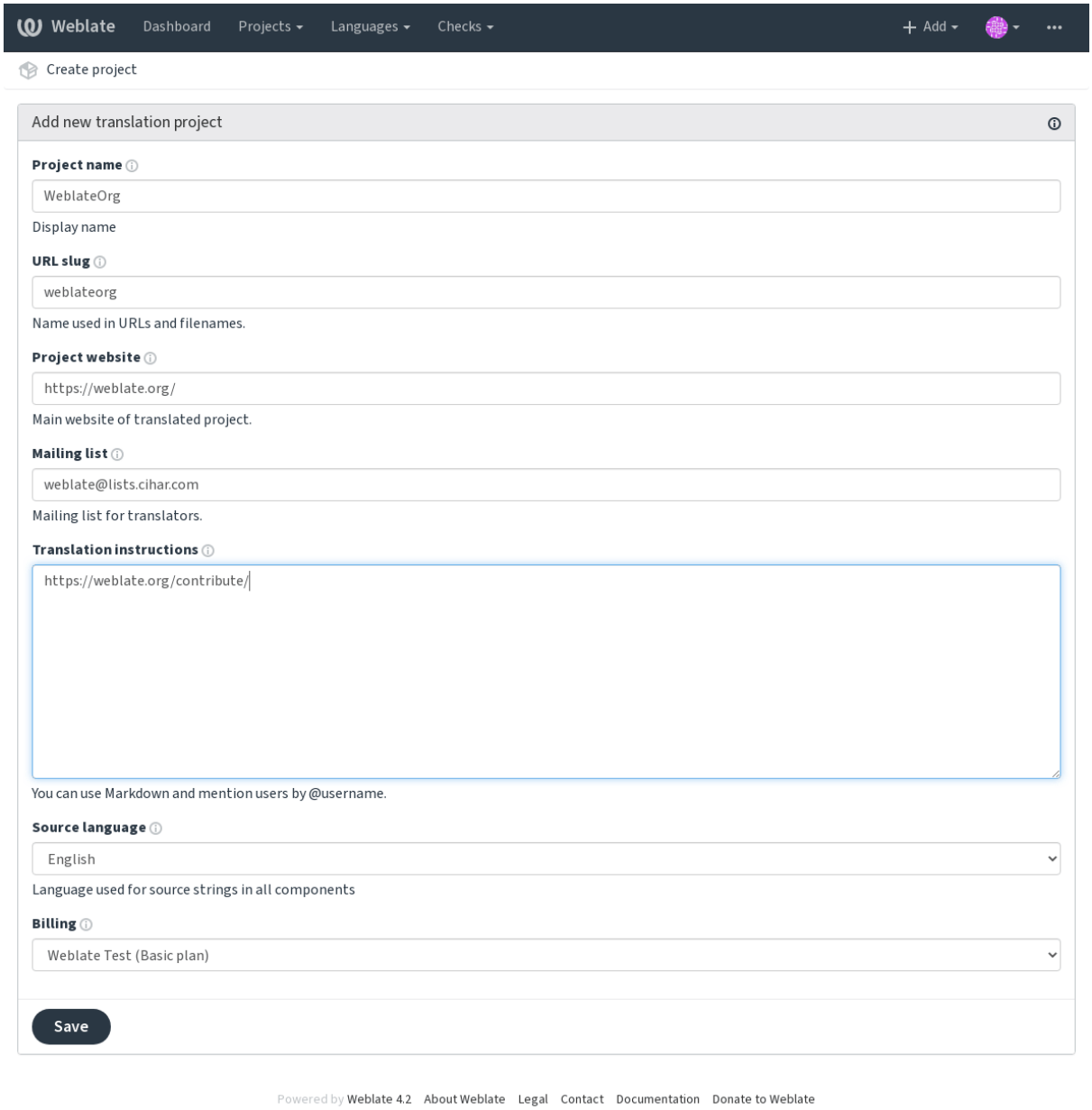
Изменено в версии 3.4: The process of adding components is now multi staged, with automated discovery of most parameters.

Based on your permissions, new translation projects and components can be created. It is always permitted for superusers, and if your instance uses billing (e.g. like https://hosted.weblate.org/ see *Биллинг*), you can also create those based on your plans allowance from the user account that manages billing.

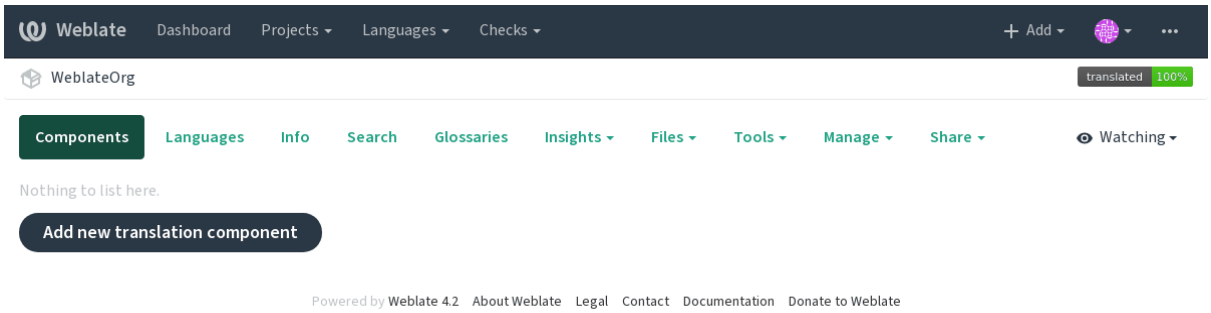
You can view your current billing plan on a separate page:



The project creation can be initiated from there, or using the menu in the navigation bar, filling in basic info about the translation project to complete addition of it:



После создания проекта вы попадаете непосредственно на страницу проекта:



Creating a new translation component can be initiated via a single click there. The process of creating a component is multi-staged and automatically detects most translation parameters. There are several approaches to creating component:

Из системы контроля версий Создаёт компонент из удаленного репозитория системы контроля версий.

Из уже существующего компонента Создаёт дополнительный компонент на основе уже существующего,

только с другими файлами.

Дополнительная ветка Создаёт дополнительный компонент на основе уже существующего, только для другой ветки.

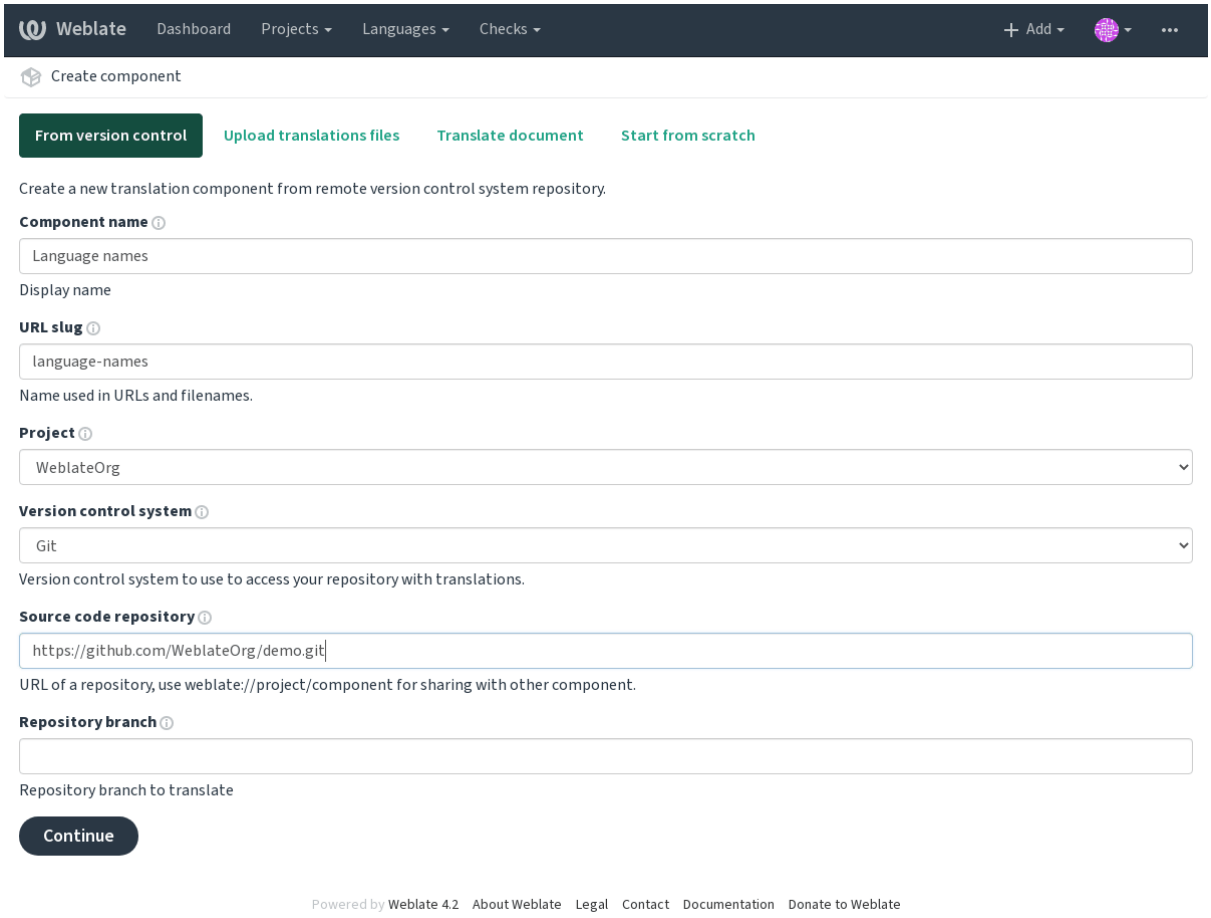
Загрузить файлы перевода Upload translation files to Weblate in case you do not have version control or do not want to integrate it with Weblate. You can later update the content using the web interface or API.

Перевести документ Загрузите одиночный документ и переведите его.

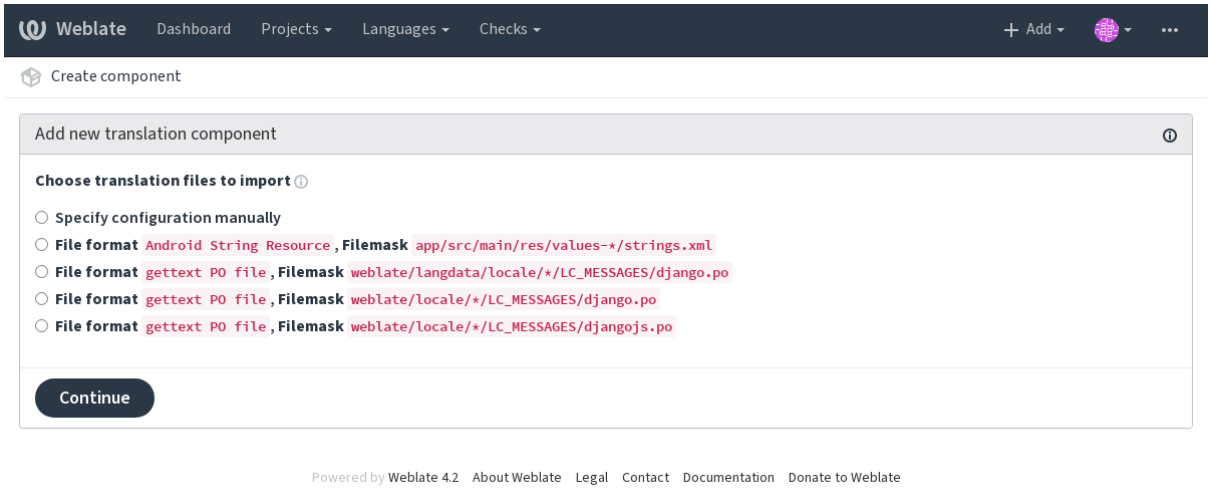
Начать с нуля Create blank translation project and add strings manually.

Once you have existing translation components, you can also easily add new ones for additional files or branches using same repository.

First you need to fill in name and repository location:



On the next page, you are presented with a list of discovered translatable resources:



As a last step, you review the translation component info and fill in optional details:

Weblate Dashboard Projects Languages Checks + Add

Create component

Add new translation component

Project

WeblateOrg

Component name

Language names

Display name

URL slug

language-names

Name used in URLs and filenames.

Version control system

Git

Version control system to use to access your repository containing translations. You can also choose additional integration with third party providers to submit merge requests.

Source code repository

https://github.com/WeblateOrg/demo.git

URL of a repository, use weblate://project/component to share it with other component.

Repository branch

Repository branch to translate

Repository push URL

URL of a push repository, pushing is turned off if empty.

Push branch

Branch for pushing changes, leave empty to use repository branch

Repository browser

https://github.com/WeblateOrg/demo/blob/{{branch}}/{{filename}}#L{{line}}

Link to repository browser, use {{branch}} for branch, {{filename}} and {{line}} as filename and line placeholders.

File format

gettext PO file

Filemask

weblate/langdata/locale/*/LC_MESSAGES/django.po

Path of files to translate relative to repository root, use * instead of language code, for example: po/*.po or locale/*/LC_MESSAGES/django.po.

Monolingual base language file

Filename of translation base file, containing all strings and their source; it is recommended for monolingual translation formats.

☑ **Edit base file**

Whether users will be able to edit the base file for monolingual translations.

Intermediate language file

Filename of intermediate translation file. In most cases this is a translation file provided by developers and is used when creating actual source strings.

Template for new translations

weblate/langdata/locale/django.pot

Filename of file used for creating new translations. For gettext choose .pot file.

Translation license

GNU General Public License v3.0 or later

Adding new translation

Create new language file

How to handle requests for creating new translations.

Language code style

Default based on the file format

Customize language code used to generate the filename for translations created by Weblate.

Language filter

^(cs|he|hu)$

Regular expression used to filter translation when scanning for filemask.

You will be able to edit more options in the component settings after creating it.

Save

Powered by Weblate 4.2 About Weblate Legal Contact Documentation Donate to Weblate

См.также:

The Django admin interface, *Project configuration*, *Конфигурация компонента*

2.7.3 Project configuration

Create a translation project and then add a new component for translation in it. The project is like a shelf, in which real translations are stacked. All components in the same project share suggestions and their dictionary; the translations are also automatically propagated through all components in a single project (unless turned off in the component configuration), see Memory Management.

These basic attributes set up and inform translators of a project:

Название проекта

Verbose project name, used to display the project name.

Плашка проекта

Название проекта, подходящее для указания его в URL-адресе.

Веб-сайт проекта

URL where translators can find more info about the project.

Список рассылки

Список рассылки, в котором переводчики могут обсуждать или комментировать переводы.

Инструкции для переводчиков

URL to more site with more detailed instructions for translators.

Set Language-Team header

Whether Weblate should manage the `Language-Team` header (this is a *GNU gettext* only feature right now).

Использовать совместную память переводов

Использовать ли совместно используемую память переводов, см. подробнее *Совместная память переводов*.

Вносить вклад в совместную память переводов

Whether to contribute to shared translation memory, see *Совместная память переводов* for more details.

Управление доступом

Configure per project access control, see *По-проектный контроль доступа* for more details.

Значение по умолчанию можно изменить установкой параметра `DEFAULT_ACCESS_CONTROL`.

Включить рецензии

Enable review workflow for translations, see *Выделенные рецензенты*.

Включить рецензии исходных строк

Enable review workflow for source strings, see *Рецензирование исходных строк*.

Включить обработчики

Whether unauthenticated *Обработчики уведомлений* are to be used for this repository.

Исходный язык

Language used for source strings in all components. Change this if you are translating from something else than English.

Подсказка: In case you are translating bilingual files from English, but want to be able to do fixes in the English translation as well, you might want to choose *English (Developer)* as a source language. To avoid conflict between name of the source language and existing translation.

For monolingual translations, you can use intermediate translation in this case, see *Файл промежуточного языка*.

См.также:

Файл промежуточного языка, *Шлюз качества для исходных строк*, *Двуязычные и одноязычные форматы*, *Определения языка*

Псевдонимы языка

Define language codes mapping when importing translations into Weblate. Use this when language codes are inconsistent in your repositories and you want to get a consistent view in Weblate.

The typical use case might be mapping American English to English: `en_US:en`

Multiple mappings to be separated by comma: `en_GB:en,en_US:en`

Подсказка: The language codes are mapped when matching the translation files and the matches are case sensitive, so make sure you use the source language codes in same form as used in the filenames.

См.также:

Разбор кодов языков

2.7.4 Конфигурация компонента

A component is a grouping of something for translation. You enter a VCS repository location and file mask for which files you want translated, and Weblate automatically fetches from this VCS, and finds all matching translatable files.

You can find some examples of typical configurations in the *Поддерживаемые форматы файлов*.

Примечание: It is recommended to keep translation components to a reasonable size - split the translation by anything that makes sense in your case (individual apps or addons, book chapters or websites).

Weblate easily handles translations with 10000s of strings, but it is harder to split work and coordinate among translators with such large translation components.

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

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

Название компонента

Verbose component name, used to display the component name.

Плашка компонента

Название компонента, подходящее для указания его в URL-адресе.

Проект компонента

Project configuration, которому принадлежит компонент.

Система контроля версий

Используемая система контроля версий, подробнее смотрите в разделе *Интеграция с системой контроля версий*.

Репозиторий исходного кода

VCS repository used to pull changes.

См.также:

See *Доступ к репозиториям* for more details on specifying URLs.

Подсказка: This can either be a real VCS URL or `weblate://project/component` indicating that the repository should be shared with another component. See *Внутренние URL-адреса Weblate* for more details.

URL для отправки в репозиторий

Repository URL used for pushing. This setting is used only for *Git* and *Mercurial* and push support is turned off for these when this is empty.

См.также:

See *Доступ к репозиториям* for more details on how to specify a repository URL and *Pushing changes from Weblate* for more details on pushing changes from Weblate.

Просмотрщик репозитория

URL of repository browser used to display source files (location of used messages). When empty, no such links will be generated. You can use *Template markup*.

For example on GitHub, use something like: `https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename}}#L{{line}}`

In case your paths are relative to different folder, you might want to strip leading directory by `parentdir` filter (see *Template markup*): `https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename|parentdir}}#L{{line}}`

URL экспортированного репозитория

URL where changes made by Weblate are exported. This is important when *Непрерывная локализация* is not used, or when there is a need to manually merge changes. You can use *Git exporter* to automate this for Git repositories.

Ветка репозитория

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

Ветка для отправки

Branch for pushing changes, leave empty to use *Ветка репозитория*.

Примечание: This is currently only supported for Git and GitHub, it is ignored for other VCS integrations.

Маска файла

Mask of files to translate, including path. It should include one «*» replacing language code (see *Определения языка* for info on how this is processed). In case your repository contains more than one translation file (e.g. more gettext domains), you need to create a component for each of them.

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

In case your filename contains special characters such as `[`, `]`, these need to be escaped as `[[]` or `[]]`.

См.также:

Двуязычные и одноязычные форматы, *Что означает сообщение «Есть больше файлов для одного языка (en)»?*

Одноязычный базовый файл

Base file containing string definitions for *Одноязычные компоненты*.

См.также:

Двуязычные и одноязычные форматы, *Что означает сообщение «Есть больше файлов для одного языка (en)»?*

Правка базового файла

Whether to allow editing the base file for *Одноязычные компоненты*.

Файл промежуточного языка

Intermediate language file for *Одноязычные компоненты*. In most cases this is a translation file provided by developers and is used when creating actual source strings.

When set, the source translation is based on this file, but all others are based on *Одноязычный базовый файл*. In case the string is not translated in source translation, translating to other languages is prohibited. This provides *Шлюз качества для исходных строк*.

См.также:

Шлюз качества для исходных строк, *Двуязычные и одноязычные форматы*, *Что означает сообщение «Есть больше файлов для одного языка (en)»?*

Шаблон для новых переводов

Base file used to generate new translations, e.g. `.pot` file with gettext.

Подсказка: In many monolingual formats Weblate starts with blank file by default. Use this in case you want to have all strings present with empty value when creating new translation.

См.также:

Добавление новых переводов, *Добавление нового перевода*, *Двуязычные и одноязычные форматы*, *Что означает сообщение «Есть больше файлов для одного языка (en)»?*

Формат файла

Translation file format, see also *Поддерживаемые форматы файлов*.

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.

Разрешить распространение перевода

Вы можете отключить распространение переводов на этот компонент из других компонентов этого же проекта. Это сильно зависит от того, что вы переводите, иногда желательно использовать разные переводы для одинаковых строк.

It's usually a good idea to turn this off for monolingual translations, unless you are using the same IDs across the whole project.

Значение по умолчанию может быть изменено параметром `DEFAULT_TRANSLATION_PROPAGATION`.

Включить предложения

Whether translation suggestions are accepted for this component.

Голосование по предложению

Turns on votecasting for suggestions, see *Голосование по предложению*.

Автоматическое принятие предложений

Automatically accept voted suggestions, see *Голосование по предложению*.

Флаги перевода

Customization of quality checks and other Weblate behavior, see *Настройка поведения*.

Принудительные проверки

List of checks which can not be ignored, see *Enforcing checks*.

Лицензия перевода

License of the translation (does not need to be the same as the source code license).

Соглашение участника

Пользовательское соглашение, которое должно быть одобрено, прежде чем пользователь сможет переводить этот компонент.

Добавление нового перевода

How to handle requests for creation of new languages. Available options:

Связаться с сопровождающими User can select desired language and the project maintainers will receive a notification about this. It is up to them to add (or not) the language to the repository.

Указать URL на инструкции по переводу User is presented a link to page which describes process of starting new translations. Use this in case more formal process is desired (for example forming a team of people before starting actual translation).

Создать новый язык Пользователь может выбрать язык и Weblate автоматически создаст для него файл, так что можно сразу начинать переводить.

Запретить добавление новых переводов There will be no option for user to start new translation.

См.также:

Добавление новых переводов.

Стиль кода языка

Customize language code used to generate the filename for translations created by Weblate, see *Добавление новых переводов* for more details.

Стиль слияния

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

Default value can be changed by `DEFAULT_MERGE_STYLE`.

Commit, add, delete, merge and addon messages

Message used when committing a translation, see *Template markup*.

Default value can be changed by `DEFAULT_ADD_MESSAGE`, `DEFAULT_ADDON_MESSAGE`, `DEFAULT_COMMIT_MESSAGE`, `DEFAULT_DELETE_MESSAGE`, `DEFAULT_MERGE_MESSAGE`.

Имя произведшего коммит

Name of the committer used for Weblate commits, the author will always be the real translator. On some VCSs this might be not supported.

Default value can be changed by `DEFAULT_COMMITER_NAME`.

E-mail делающего коммит

Email of committer used for Weblate commits, the author will always be the real translator. On some VCSs this might be not supported. The default value can be changed in `DEFAULT_COMMITER_EMAIL`.

Отправлять при коммите

Whether committed changes should be automatically pushed to the upstream repository. When enabled, the push is initiated once Weblate commits changes to its internal repository (see *Lazy commits*). To actually enable pushing *Repository push URL* has to be configured as well.

Возраст изменений для коммита

Sets how old changes (in hours) are to get before they are committed by background task or `commit_pending` management command. All changes in a component are committed once there is at least one older than this period.

Default value can be changed by `COMMIT_PENDING_HOURS`.

Блокировать при ошибке

Enables locking the component on repository error (failed pull, push or merge). Locking in this situation avoids adding another conflict which would have to be resolved manually.

The component will be automatically unlocked once there are no repository errors left.

Языковой фильтр

Regular expression used to filter the translation when scanning for filemask. This can be used to limit the list of languages managed by Weblate.

Примечание: You need to list language codes as they appear in the filename.

Some examples of filtering:

Filter description	Регулярное выражение
Selected languages only	`^(cs|de|es)$`
Exclude languages	`^(?!(it|fr)$).+$`
Exclude non language files	`^(?!(blank)$).+$`
Include all files (default)	`^[^.]+$`

Регулярное выражение вариантов

Regular expression used to determine the variants of a string, see *Варианты строк*.

Примечание: Most of the fields can be edited by project owners or managers, in the Weblate interface.

См.также:

Поддерживает ли Weblate другие системы контроля версий, кроме Git и Mercurial?, *Предупреждения компонента перевода*

Приоритет

Переводчикам вначале предлагаются компоненты с большим приоритетом.

Restricted access

By default the component is visible to anybody who has access to the project, even if the person can not perform any changes in the component. This makes it easier to keep translation consistency within the project.

Enable this in case you want to grant access to this component explicitly - the project level permissions will not apply and you will have to specify component or component list level permission in order to grant access.

Default value can be changed by `DEFAULT_RESTRICTED_COMPONENT`.

Подсказка: This applies to project managers as well - please make sure you will not loose access to the component after toggling the status.

2.7.5 Template markup

Weblate uses simple markup language in several places where text rendering is needed. It is based on The Django template language, so it can be quite powerful.

Currently it is used in:

- Commit message formatting, see *Конфигурация компонента*
- **Several addons**
  - *Обнаружение компонента*
  - *Генератор статистики*
  - *Executing scripts from addon*

There following variables are available in the component templates:

`{{ language_code }}` Код языка

`{{ language_name }}` Название языка

`{{ component_name }}` Название компонента

`{{ component_slug }}` Плашка компонента

`{{ project_name }}` Название проекта

`{{ project_slug }}` Плашка проекта

`{{ url }}` Translation URL

`{{ filename }}` Имя файла перевода

`{{ stats }}` Translation stats, this has further attributes, examples below.

`{{ stats.all }}` Total strings count

`{{ stats.fuzzy }}` Count of strings needing review

`{{ stats.fuzzy_percent }}` Percent of strings needing review

`{{ stats.translated }}` Translated strings count

`{{ stats.translated_percent }}` Translated strings percent

`{{ stats.allchecks }}` Number of strings with failing checks

`{{ stats.allchecks_percent }}` Percent of strings with failing checks

`{{ author }}` Author of current commit, available only in the commit scope.

`{{ addon_name }}` Name of currently executed addon, available only in the addon commit message.

The following variables are available in the repository browser or editor templates:

`{{branch}}` current branch

`{{line}}` line in file

`{{filename}}` filename, you can also strip leading parts using the `parentdir` filter, for example `{{filename|parentdir}}`

You can combine them with filters:

```
{{ component|title }}
```

You can use conditions:

```
{% if stats.translated_percent > 80 %}Well translated!{% endif %}
```

There is additional tag available for replacing characters:

```
{% replace component "-" " " %}
```

You can combine it with filters:

```
{% replace component|capfirst "-" " " %}
```

There are also additional filter to manipulate with filenames:

```
Directory of a file: {{ filename|dirname }}
File without extension: {{ filename|stripext }}
File in parent dir: {{ filename|parentdir }}
It can be used multiple times:  {{ filename|parentdir|parentdir }}
```

…and other Django template features.

2.7.6 Importing speed

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

Optimize configuration

Конфигурация по умолчанию полезна для тестирования и отладки Weblate, в то же время для установки в рабочую среду в неё необходимо внести некоторые коррективы. Многие из параметров довольно сильно влияют на производительность. Для получения более подробной информации обратитесь к разделу *Рабочая среда*, особенно обратите внимание на следующие темы:

- Configure Celery for executing background tasks (see *Фоновые задачи с использованием Celery*)
- *Включение кэширования*
- *Использование мощного движка базы данных*
- *Отключение отладочного режима*

Check resource limits

If you are importing huge translations or repositories, you might be hit by resource limitations of your server.

- Check the amount of free memory, having translation files cached by the operating system will greatly improve performance.
- Disk operations might be bottleneck if there is a lot of strings to process—the disk is pushed by both Weblate and the database.
- Additional CPU cores might help improve performance of background tasks (see *Фоновые задачи с использованием Celery*).

Disable unneeded checks

Some quality checks can be quite expensive, and if not needed, can save you some time during import if omitted. See `CHECK_LIST` for info on configuration.

2.7.7 Automatic creation of components

In case your project has dozen of translation files (e.g. for different gettext domains, or parts of Android apps), you might want to import them automatically. This can either be achieved from the command line by using `import_project` or `import_json`, or by installing the *Обнаружение компонента* addon.

To use the addon, you first need to create a component for one translation file (choose the one that is the least likely to be renamed or removed in future), and install the addon on this component.

For the management commands, you need to create a project which will contain all components and then run `import_project` or `import_json`.

См.также:

Команды управления, *Обнаружение компонента*

2.8 Определения языка

Для правильного представления различных переводов необходима информация о названии языка, направлении текста, определения форм множественного числа и код языка. В поставку включены определения примерно для 350 языков.

2.8.1 Разбор кодов языков

При разборе переводов Weblate пытается сопоставить код языка (обычно по ISO 639-1) с любым существующим объектом языка.

You can further adjust this mapping at project level by *Псевдонимы языка*.

If no exact match can be found, an attempt will be made to best fit it into an existing language (e.g. ignoring the default country code for a given language—choosing `cs` instead of `cs_CZ`).

Should that also fail, a new language definition will be created using the defaults (left to right text direction, one plural) and naming of the language as *xx_XX (generated)*. You might want to change this in the admin interface later, (see *Изменение определений языка*) and report it to the issue tracker (see *Внесение вклада в Weblate*).

Подсказка: In case you see something unwanted as a language, you might want to adjust *Языковой фильтр* to ignore such file when parsing translations.

2.8.2 Изменение определений языка

Вы можете изменить определения языка в интерфейсе по языкам (по адресу `/languages/`).

While editing, make sure all fields are correct (especially plurals and text direction), otherwise translators will be unable to properly edit those translations.

2.8.3 Определения языка

Each language consists of following fields:

Код языка

Code identifying the language. Weblate prefers two letter codes as defined by ISO 639-1, but uses ISO 639-2 or ISO 639-3 codes for languages that do not have two letter code. It can also support extended codes as defined by BCP 47.

См.также:

Разбор кодов языков

Название языка

Visible name of the language. The language names included in Weblate are also being localized depending on user interface language.

Направление текста

Determines whether language is written right to left or left to right. This property is autodetected correctly for most of the languages.

Количество форм множественного числа

Количество используемых в языке форм множественного числа.

Формула множественного числа

Совместимая с Gettext формула множественного числа, используемая для определения того, какая форма множественного числа используется для данного количества.

См.также:

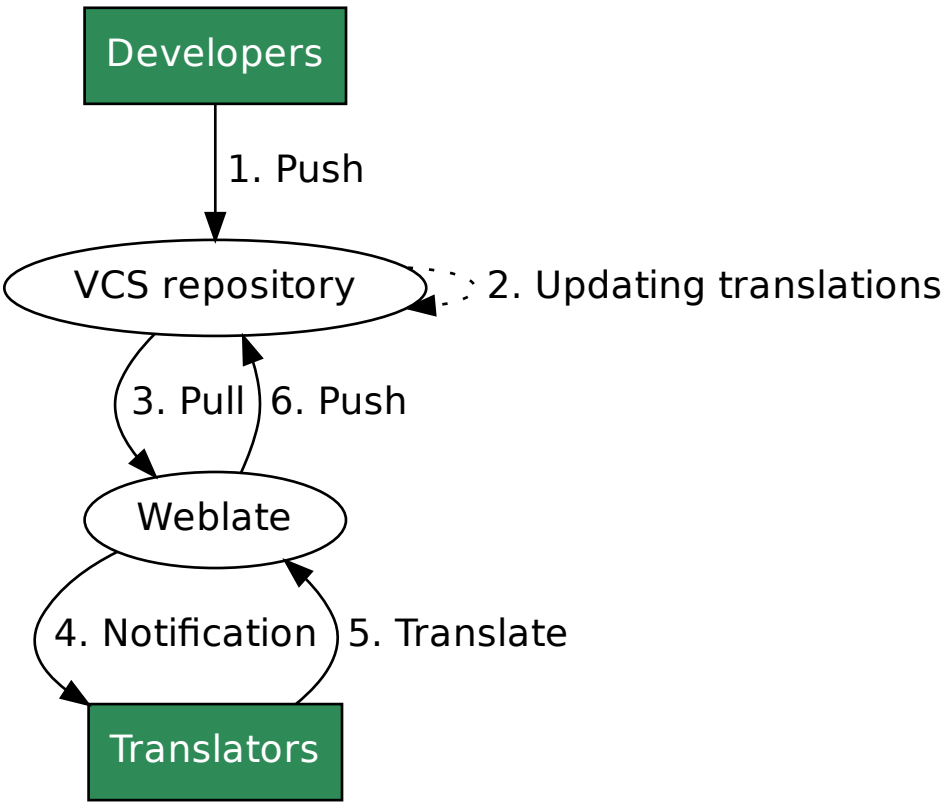
Формы множественного числа, GNU gettext utilities: Plural forms, Language Plural Rules by the Unicode Consortium

2.9 Непрерывная локализация

There is infrastructure in place so that your translation closely follows development. This way translators can work on translations the entire time, instead of working through huge amount of new text just prior to release.

This is the process:

1. Developers make changes and push them to the VCS repository.
2. Optionally the translation files are updated (this depends on the file format, see *Почему Weblate до сих пор показывает старые строки перевода, хотя я уже обновил шаблон?*).
3. Weblate pulls changes from the VCS repository, see *Updating repositories*.
4. Once Weblate detects changes in translations, translators are notified based on their subscription settings.
5. Translators submit translations using the Weblate web interface, or upload offline changes.
6. Once the translators are finished, Weblate commits the changes to the local repository (see *Lazy commits*) and pushes them back if it has permissions to do so (see *Pushing changes from Weblate*).



2.9.1 Updating repositories

You should set up some way of updating backend repositories from their source.

- Use *Обработчики уведомлений* to integrate with most of common code hosting services
- Manually trigger update either in the repository management or using API or *Клиент Weblate*
- Enable `AUTO_UPDATE` to automatically update all components on your Weblate instance
- Execute `updategit` (with selection of project or *–all* to update all)

Whenever Weblate updates the repository, the post update addons will be triggered, see *Надстройки*.

Avoiding merge conflicts

The merge conflicts from Weblate arise when same file was changed both in Weblate and outside it. There are two approaches to deal with that - avoid edits outside Weblate or integrate Weblate into your updating process, so that it flushes changes prior to updating the files outside Weblate.

The first approach is easy with monolingual files - you can add new strings within Weblate and leave whole editing of the files there. For bilingual files, there is usually some kind of message extraction process to generate translatable files from the source code. In some cases this can be split into two parts - one for the extraction generates template (for example gettext POT is generated using **`xgettext`**) and then further process merges it into actual translations (the gettext PO files are updated using **`msgmerge`**). You can perform the second step within Weblate and it will make sure that all pending changes are included prior to this operation.

The second approach can be achieved by using API to force Weblate to push all pending changes and lock the translation while you are doing changes on your side.

The script for doing updates can look like this:

```
# Lock Weblate translation
wlc lock
# Push changes from Weblate to upstream repository
wlc push
# Pull changes from upstream repository to your local copy
git pull
# Update translation files, this example is for Django
./manage.py makemessages --keep-pot -a
git commit -m 'Locale updates' -- locale
# Push changes to upstream repository
git push
# Tell Weblate to pull changes (not needed if Weblate follows your repo
# automatically)
wlc pull
# Unlock translations
wlc unlock
```

If you have multiple components sharing same repository, you need to lock them all separately:

```
wlc lock foo/bar
wlc lock foo/baz
wlc lock foo/baj
```

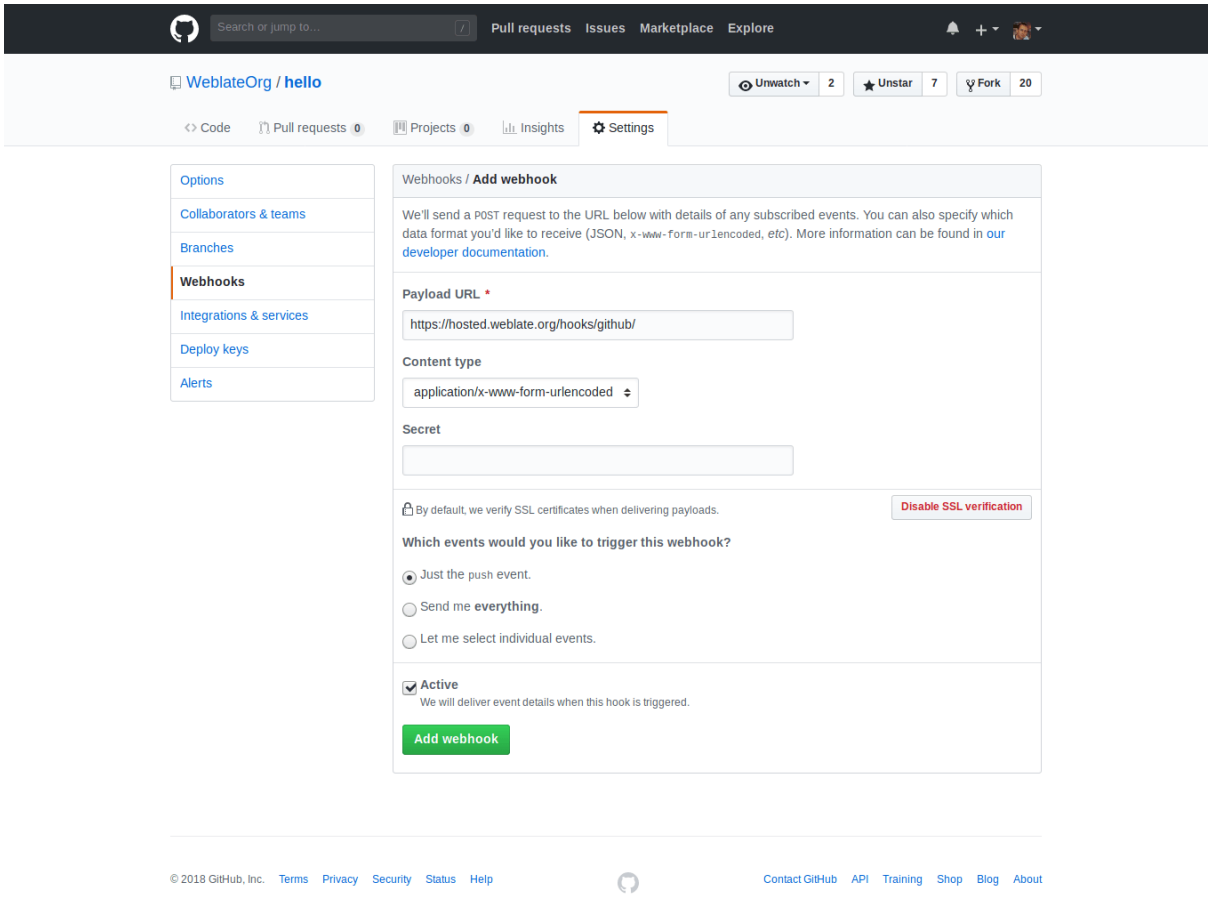
Примечание: The example uses *Клиент Weblate*, which needs configuration (API keys) to be able to control Weblate remotely. You can also achieve this using any HTTP client instead of wlc, e.g. curl, see API.

Automatically receiving changes from GitHub

Weblate comes with native support for GitHub.

If you are using Hosted Weblate, the recommended approach is to install the Weblate app, that way you will get the correct setup without having to set much up. It can also be used for pushing changes back.

To receive notifications on every push to a GitHub repository, add the Weblate Webhook in the repository settings (*Webhooks*) as shown on the image below:



For the payload URL, append `/hooks/github/` to your Weblate URL, for example for the Hosted Weblate service, this is `https://hosted.weblate.org/hooks/github/`.

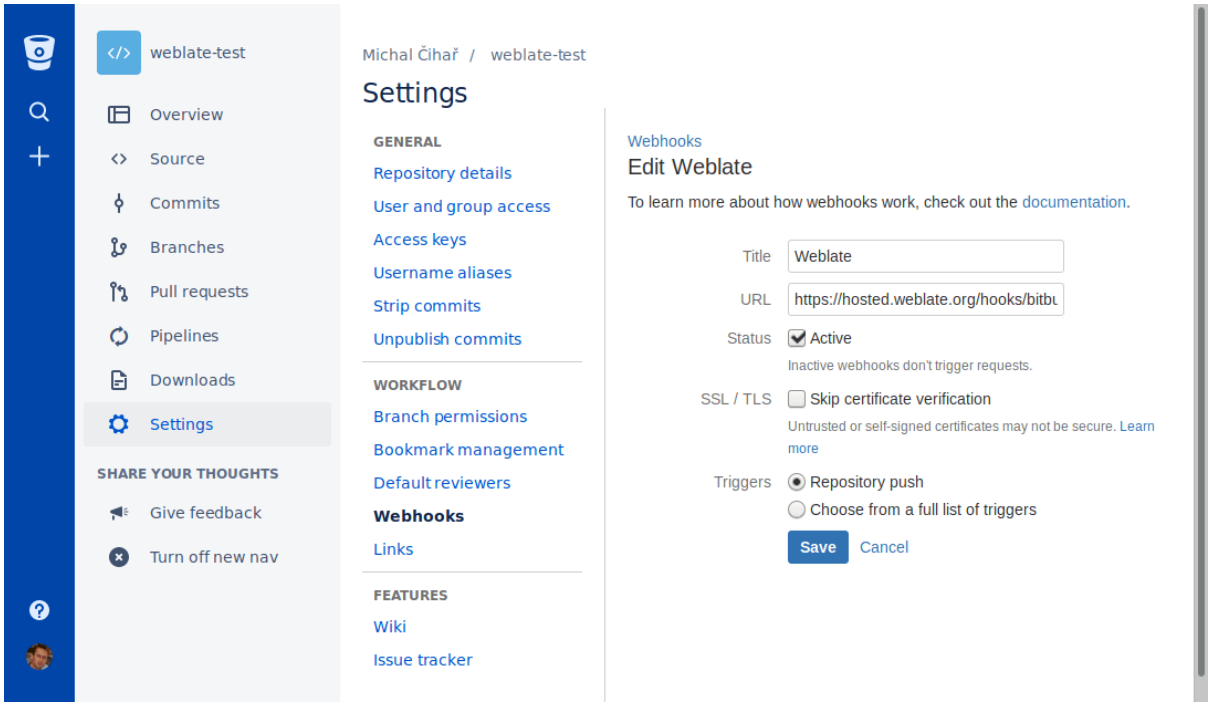
You can leave other values at default settings (Weblate can handle both content types and consumes just the *push* event).

См.также:

`POST /hooks/github/`, *Доступ к репозиториям из Hosted Weblate*

Automatically receiving changes from Bitbucket

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



См.также:

`POST /hooks/bitbucket/`, *Доступ к репозиториям из Hosted Weblate*

Automatically receiving changes from GitLab

Weblate has support for GitLab hooks, add a project webhook with destination to `/hooks/gitlab/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/gitlab/`).

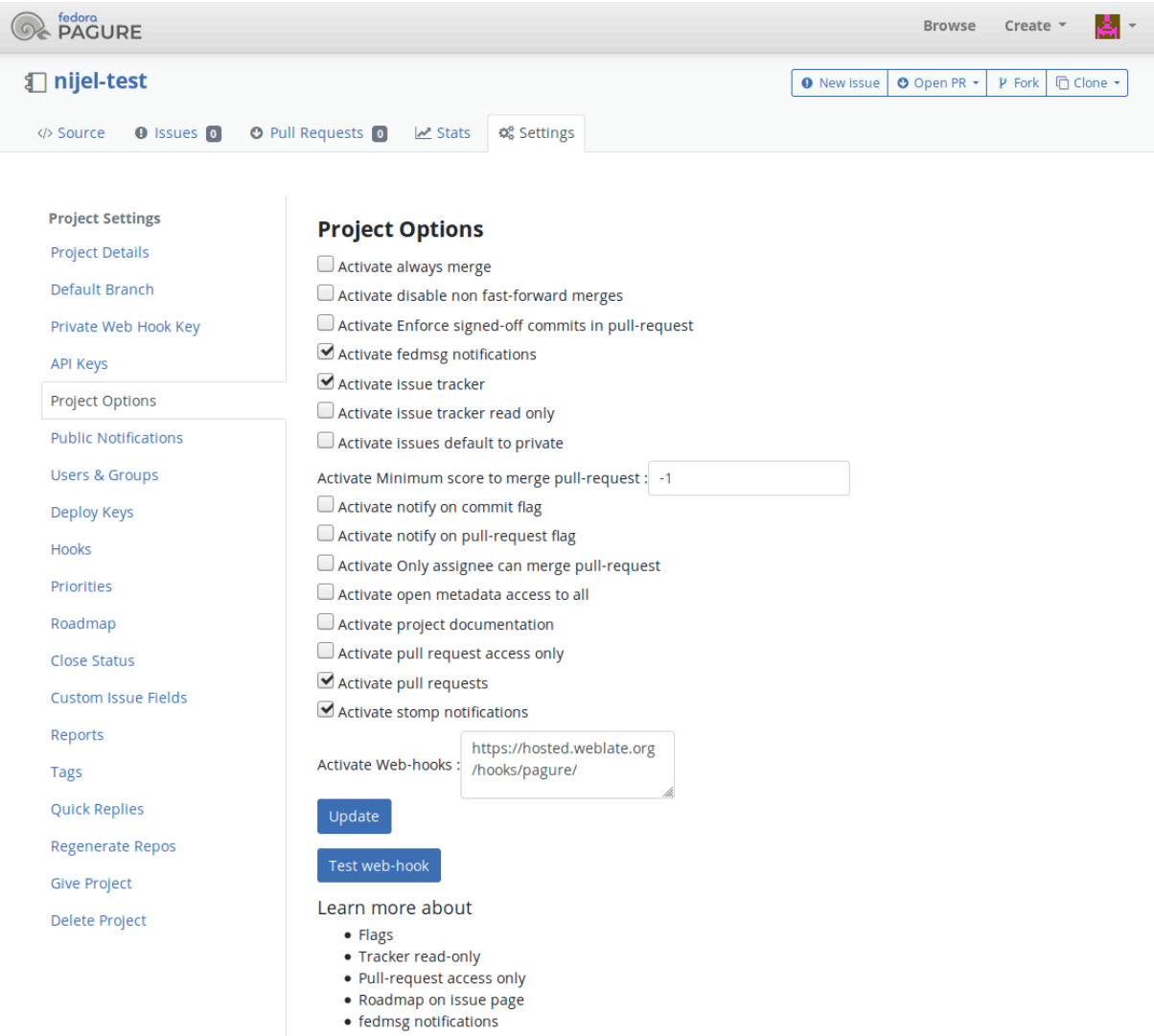
См.также:

`POST /hooks/gitlab/`, *Доступ к репозиториям из Hosted Weblate*

Automatically receiving changes from Pagure

Добавлено в версии 3.3.

Weblate has support for Pagure hooks, add a webhook with destination to `/hooks/pagure/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/pagure/`). This can be done in *Activate Web-hooks* under *Project options*:



См.также:

`POST /hooks/pagure/`, *Доступ к репозиториям из Hosted Weblate*

Automatically receiving changes from Azure Repos

Добавлено в версии 3.8.

Weblate has support for Azure Repos web hooks, add a webhook for *Code pushed* event with destination to `/hooks/azure/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/azure/`). This can be done in *Service hooks* under *Project settings*.

См.также:

Web hooks in Azure DevOps manual, `POST /hooks/azure/`, *Доступ к репозиториям из Hosted Weblate*

Automatically receiving changes from Gitea Repos

Добавлено в версии 3.9.

Weblate has support for Gitea webhooks, add a *Gitea Webhook* for *Push events* event with destination to `/hooks/gitea/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/gitea/`). This can be done in *Webhooks* under repository *Settings*.

См.также:

Webhooks in Gitea manual, `POST /hooks/gitea/`, *Доступ к репозиториям из Hosted Weblate*

Automatically receiving changes from Gitee Repos

Добавлено в версии 3.9.

Weblate has support for Gitee webhooks, add a *WebHook* for *Push* event with destination to `/hooks/gitee/` URL on your Weblate installation (for example `https://hosted.weblate.org/hooks/gitee/`). This can be done in *WebHooks* under repository *Management*.

См.также:

Webhooks in Gitee manual, `POST /hooks/gitee/`, *Доступ к репозиториям из Hosted Weblate*

Automatically updating repositories nightly

Weblate automatically fetches remote repositories nightly to improve performance when merging changes later. You can optionally turn this into doing nightly merges as well, by enabling `AUTO_UPDATE`.

2.9.2 Pushing changes from Weblate

Each translation component can have a push URL set up (see *URL для отправки в репозиторий*), and in that case Weblate will be able to push change to the remote repository. Weblate can be also be configured to automatically push changes on every commit (this is default, see *Отправлять при коммите*). If you do not want changes to be pushed automatically, you can do that manually under *Repository maintenance* or using API via `wlc push`.

The push options differ based on the *Интеграция с системой контроля версий* used, more details are found in that chapter.

In case you do not want direct pushes by Weblate, there is support for *GitHub*, *GitLab* pull requests or *Gerrit* reviews, you can activate these by choosing *GitHub*, *GitLab* or *Gerrit* as *Система контроля версий* in *Конфигурация компонента*.

Overall, following options are available with Git, GitHub and GitLab:

Desired setup	*Система контроля версий*	*URL для отправки в репозиторий*	*Ветка для отправки*
No push	*Git*	*empty*	*empty*
Push directly	*Git*	SSH URL	*empty*
Отправка в отдельную ветку	*Git*	SSH URL	Branch name
GitHub pull request from fork	*GitHub*	*empty*	*empty*
GitHub pull request from branch	*GitHub*	SSH URL[1]	Branch name
GitLab merge request from fork	*GitLab*	*empty*	*empty*
GitLab merge request from branch	*GitLab*	SSH URL[1]	Branch name

Примечание: You can also enable automatic pushing of changes after Weblate commits, this can be done in *Отправлять при коммите*.

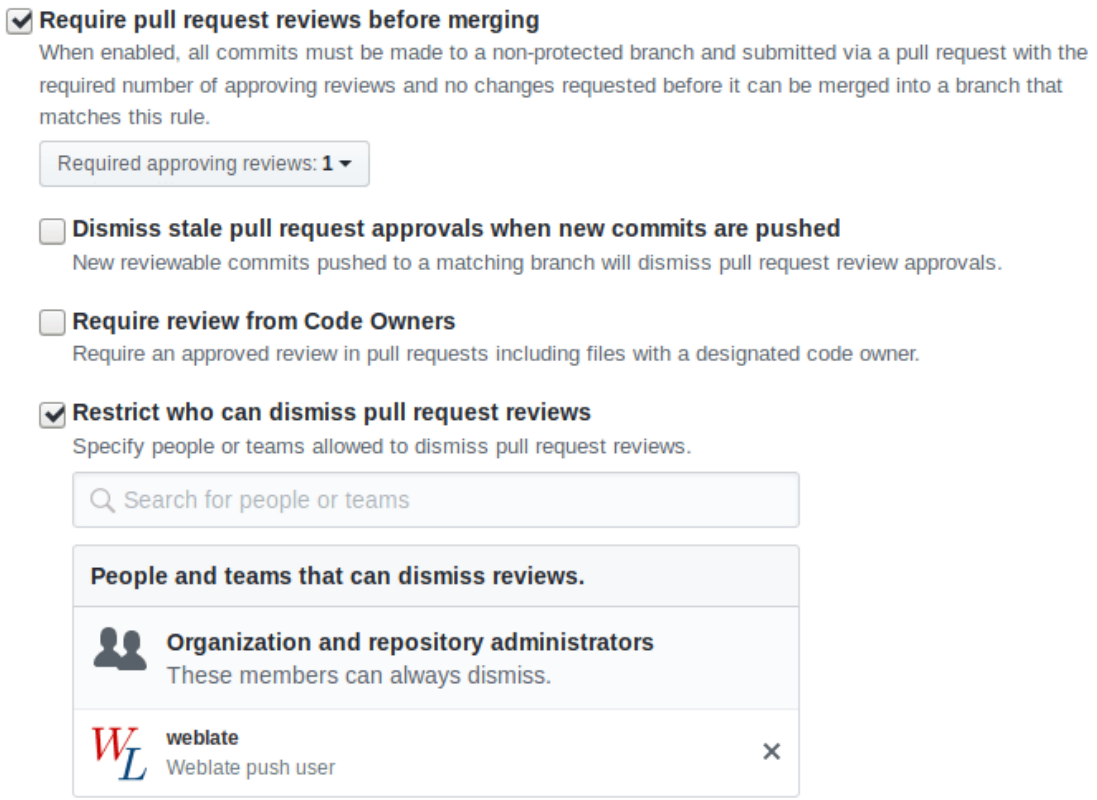
См.также:

See *Доступ к репозиториям* for setting up SSH keys, and *Lazy commits* for info about when Weblate decides to commit changes.

Protected branches

If you are using Weblate on protected branch, you can configure it to use pull requests and perform actual review on the translations (what might be problematic for languages you do not know). An alternative approach is to waive this limitation for the Weblate push user.

For example on GitHub this can be done in the repository configuration:



2.9.3 Merge or rebase

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

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

[1] Can be empty in case *Репозиторий исходного кода* supports pushing.

2.9.4 Interacting with others

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

См.также:

API

2.9.5 Lazy commits

The behaviour of Weblate is to group commits from the same author into one commit if possible. This greatly reduces the number of commits, however you might need to explicitly tell it to do the commits in case you want to get the VCS repository in sync, e.g. for merge (this is by default allowed for the Managers group, see *Управление доступом*).

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

- Somebody else changes an already changed string.
- A merge from upstream occurs.
- An explicit commit is requested.
- Change is older than period defined as *Age of changes to commit* on *Конфигурация компонента*.

Подсказка: Commits are created for every component. So in case you have many components you will still see lot of commits. You might utilize *Склеивать Git-коммиты* addon in that case.

If you want to commit changes more frequently and without checking of age, you can schedule a regular task to perform a commit:

```
CELERY_BEAT_SCHEDULE = {
    # Unconditionally commit all changes every 2 minutes
    "commit": {
        "task": "weblate.trans.tasks.commit_pending",
        # Ommiting hours will honor per component settings,
        # otherwise components with no changes older than this
        # won't be committed
        "kwargs": {"hours": 0},
        # How frequently to execute the job in seconds
        "schedule": 120,
    }
}
```

2.9.6 Processing repository with scripts

The way to customize how Weblate interacts with the repository is *Надстройки*. Consult *Executing scripts from addon* for info on how to execute external scripts through addons.

2.9.7 Keeping translations same across components

Once you have multiple translation components, you might want to ensure that the same strings have same translation. This can be achieved at several levels.

Распространение перевода

При включённой функции распространения перевода (которая включена по умолчанию, смотрите раздел *Конфигурация компонента*), все новые переводы автоматически копируются во все компоненты с совпадающими строками. Такие переводы должным образом засчитываются текущему пользователю-переводчику во всех компонентах.

Примечание: Распространение перевода требует, чтобы в одноязычных форматах файлов перевода ключи строк совпадали, так что об этом следует помнить при создании ключей перевода.

Consistency check

The *Противоречия* check fires whenever the strings are different. You can utilize this to review such differences manually and choose the right translation.

Автоматический перевод

Automatic translation based on different components can be way to synchronize the translations across components. You can either trigger it manually (see *Автоматический перевод*) or make it run automatically on repository update using addon (see *Автоматический перевод*).

2.10 Лицензирование переводов

You can specify which license translations are contributed under. This is especially important to do if translations are open to the public, to stipulate what they can be used for.

You should specify *Конфигурация компонента* license info. You should avoid requiring a contributor license agreement, though it is possible.

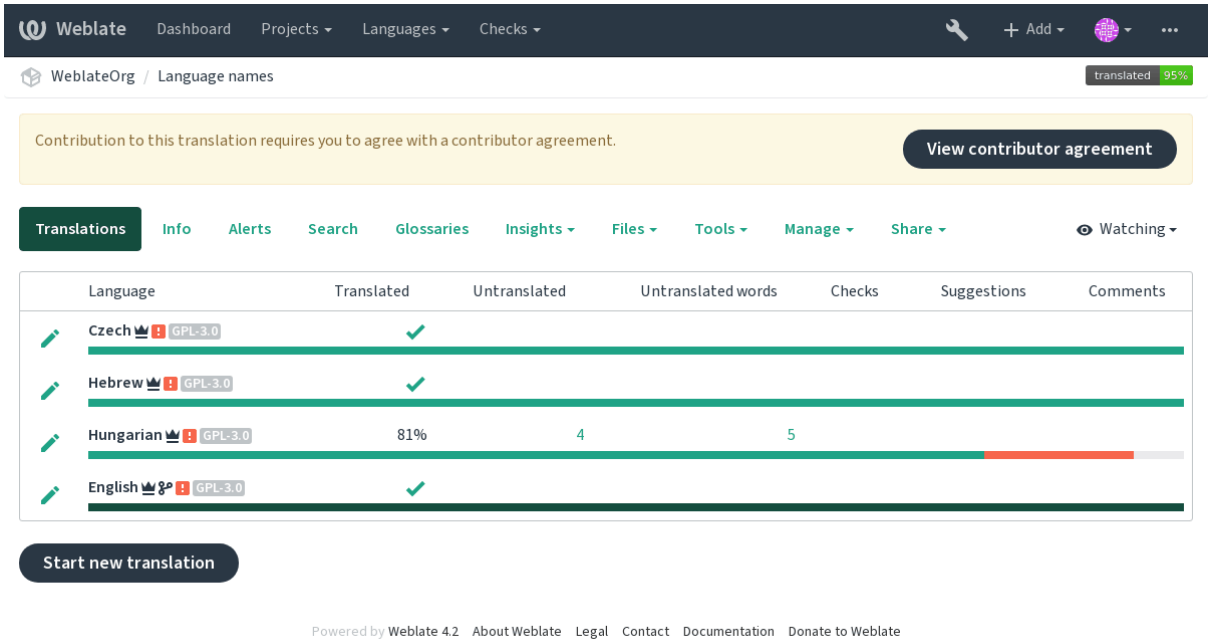
2.10.1 Информация о лицензии

Upon specifying license info (license name and URL), this info is shown in the translation info section of the respective *Конфигурация компонента*.

Usually this is best place to post licensing info if no explicit consent is required. If your project or translation is not libre you most probably need prior consent.

2.10.2 Соглашение участника

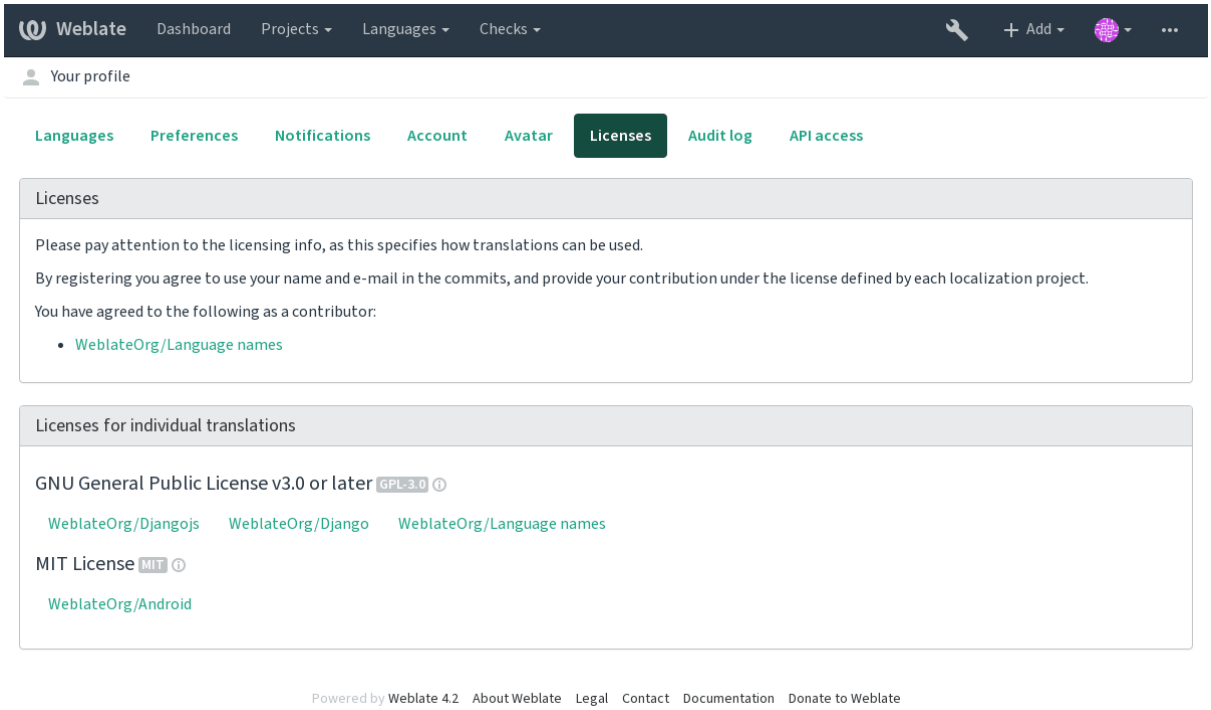
If you specify a contributor license agreement, only users who have agreed to it will be able to contribute. This is a clearly visible step when accessing the translation:



The entered text is formatted into paragraphs and external links can be included. HTML markup can not be used.

2.10.3 Пользовательские лицензии

Any user can review all translation licenses of all public projects on the instance from their profile:



2.11 Процесс перевода

2.11.1 Голосование по предложению

Everyone can add suggestions by default, to be accepted by signed in users. Suggestion voting can be used to make use of a string when more than signed in user agrees, by setting up the *Конфигурация компонента* configuration with *Suggestion voting* to turn on voting, and *Autoaccept suggestions* to set a threshold for accepted suggestions (this includes a vote from the user making the suggestion if it is cast).

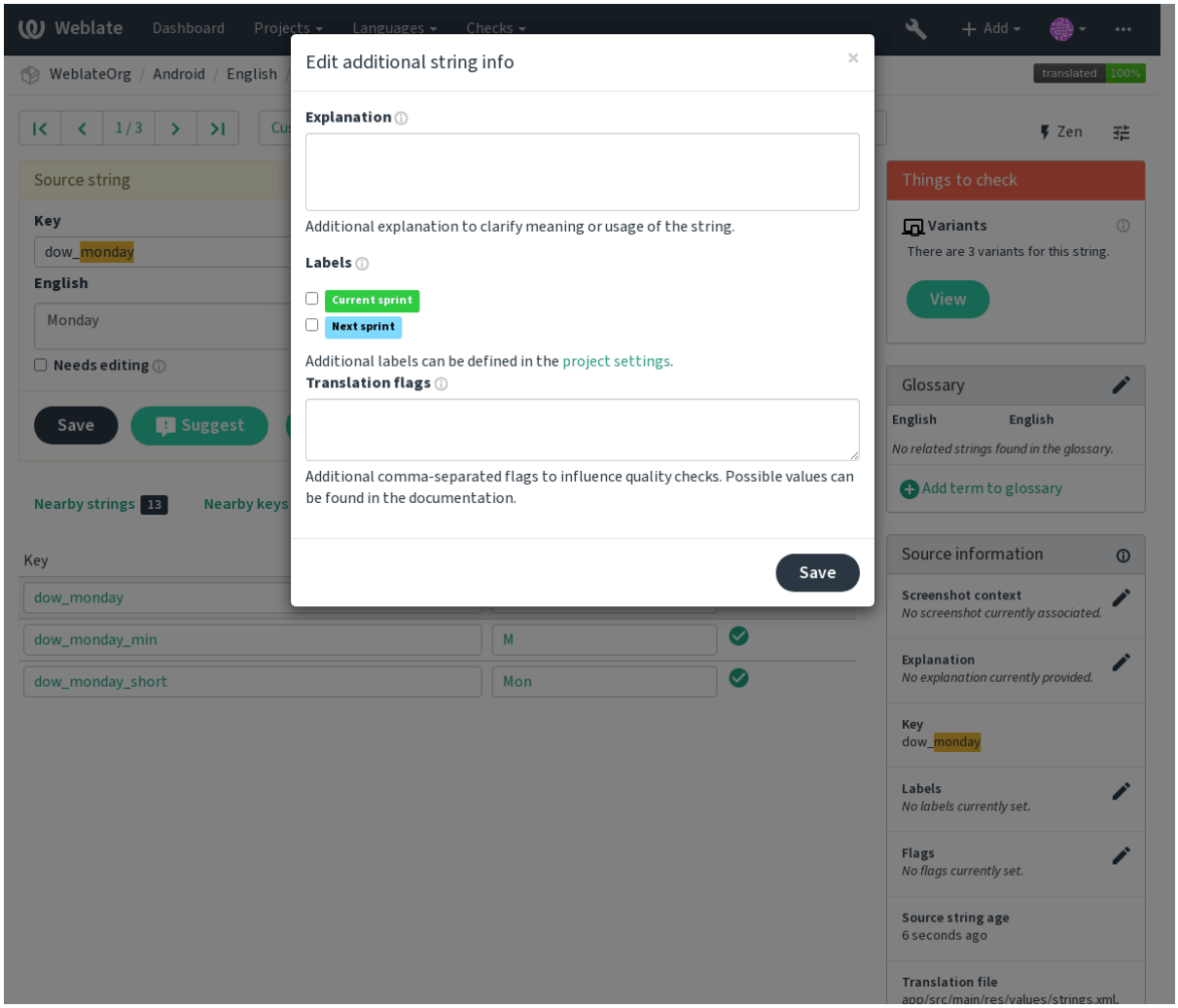
Примечание: Once automatic acceptance is set up, normal users lose the privilege to directly save translations or accept suggestions. This can be overridden with the *Can override suggestion state* privilege (see *Управление доступом*).

You can combine these with *Управление доступом* into one of the following setups:

- Users suggest and vote for suggestions and a limited group controls what is accepted. - Turn on voting. - Turn off automatic acceptance. - Don't let users save translations.
- Users suggest and vote for suggestions with automatic acceptance once the defined number of them agree. - Turn on voting. - Set the desired number of votes for automatic acceptance.
- Optional voting for suggestions. (Can optionally be used by users when they are unsure about a translation by making multiple suggestions.) - Only turn on voting.

2.11.2 Additional info on source strings

Enhance the translation process with info available in the translation files. This includes explanation, string priority, check flags, or providing visual context. All these features can be set in the *Рецензирование строк*:



Access this directly from the translation interface by clicking the «Edit» icon next to *Screenshot context* or *Flags*.

Weblate Dashboard Projects Languages Checks Add

WeblateOrg / Django / Czech / Translate translated 96%

11 / 26 All strings Position and priority Zen

Translation

Explanation

Help text for automatic translation tool

English

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

Czech Clone source NBS

Automatický překlad prostřednictvím strojového překladu používá aktivní enginy strojového překladu pro získání nejlepších možných překladů a použije je na tento projekt.

Needs editing 169/1570

Save Suggest Skip

Nearby strings 26 Comments Machinery Other languages History

Context	English	Czech	State
	Files	Soubory	
	Automatic translation	Automatický překlad	
	Add new translation string	Add new translation string	
	Translation status	Stav překladu	
	Singular %(count)s word **Plural** %(count)s words	**One** %(count)s slovo **Few** %(count)s slova **Other** %(count)s slov	
	Other components	Další součásti	
	Translation file	Soubor s překladem	
	Download	Stáhnout	
	Browse all translation changes	Procházet všechny změny v překladu.	
	Automatic translation takes existing translations in this project and applies them to the current component. It can be used to push translations to a different branch, to fix inconsistent translations or to translate a new component using translation memory.	Automatický překlad použije stávající překlady v projektu na tuto součást. Může být užitečný pro sloučení překladů z jiné větve, opravu nekonzistentních překladů nebo překlad nové součásti pomocí překladové paměti.	
	Automatic translation via machine translation uses active machine translation engines to get the best possible translations and applies them in this project.	Automatický překlad prostřednictvím strojového překladu používá aktivní enginy strojového překladu pro získání nejlepších možných překladů a použije je na tento projekt.	
	You can add new translation string here, it will automatically appear in all translations.	Zde můžete přidat nový řetězec k překladu, automaticky se objeví ve všech jazycích.	
	The uploaded file will be merged with the current translation. In case you want to overwrite already translated strings, don't forget to enable it.	Nahraný soubor bude sloučen se stávajícími překlady. Pokud chcete přepsat již přeložené řetězce, nezapomeňte to povolit.	
	The uploaded file will be merged with the current translation.	Nahraný soubor bude sloučen se stávajícími překlady.	
	The fulltext search might not work properly as the fulltext index for this translation is not yet up to date.	Fulltextové vyhledávání nemusí fungovat správně, protože fulltextový index pro tento překlad ještě není plně zpracován.	
	Review	Kontrola	
	Review translations touched by other users.	Zkontrolovat překlady od ostatních uživatelů.	
	Start review	Začít kontrolu	
	Percent	Procenta	
	Total	Celkem	
	Failing check	Neúspěšných kontrol	
	Last activity	Poslední aktivita	
	Last change	Poslední změna	
	Last author	Poslední autor	
Question for a mathematics-based CAPTCHA, the %s is an arithmetic problem	What is %s?	Kolik to je?	
	The string uses three dots (...) instead of an ellipsis character (…)		

Glossary

English	Czech	
machine translation	strojový překlad	WeblateOrg
project	projekt	WeblateOrg

Add term to glossary

Source information

Screenshot context
No screenshot currently associated.

Explanation
Help text for automatic translation tool

Labels
No labels currently set.

Flags
No flags currently set.

Source string location
weblate/templates/translation.html:212

Source string age
2 seconds ago

Translation file
weblate/locale/cs/LC_MESSAGES/django.po, string 11

Powered by Weblate 4.2 About Weblate Legal Contact Documentation Donate to Weblate

Strings prioritization

Добавлено в версии 2.0.

String priority can be changed to offer higher priority strings for translation earlier by using the `priority` flag

Подсказка: This can be used to order the flow of translation in a logical manner.

См.также:

Проверки качества

Флаги перевода

Добавлено в версии 2.4.

Изменено в версии 3.3: Previously called *Quality checks flags*, it no longer configures only checks.

The default set of translation flags is determined by the translation *Конфигурация компонента* and the translation file. However, you might want to use it to customize this per source string.

См.также:

Проверки качества

Объяснение

Изменено в версии 4.1: In previous version this has been called extra context.

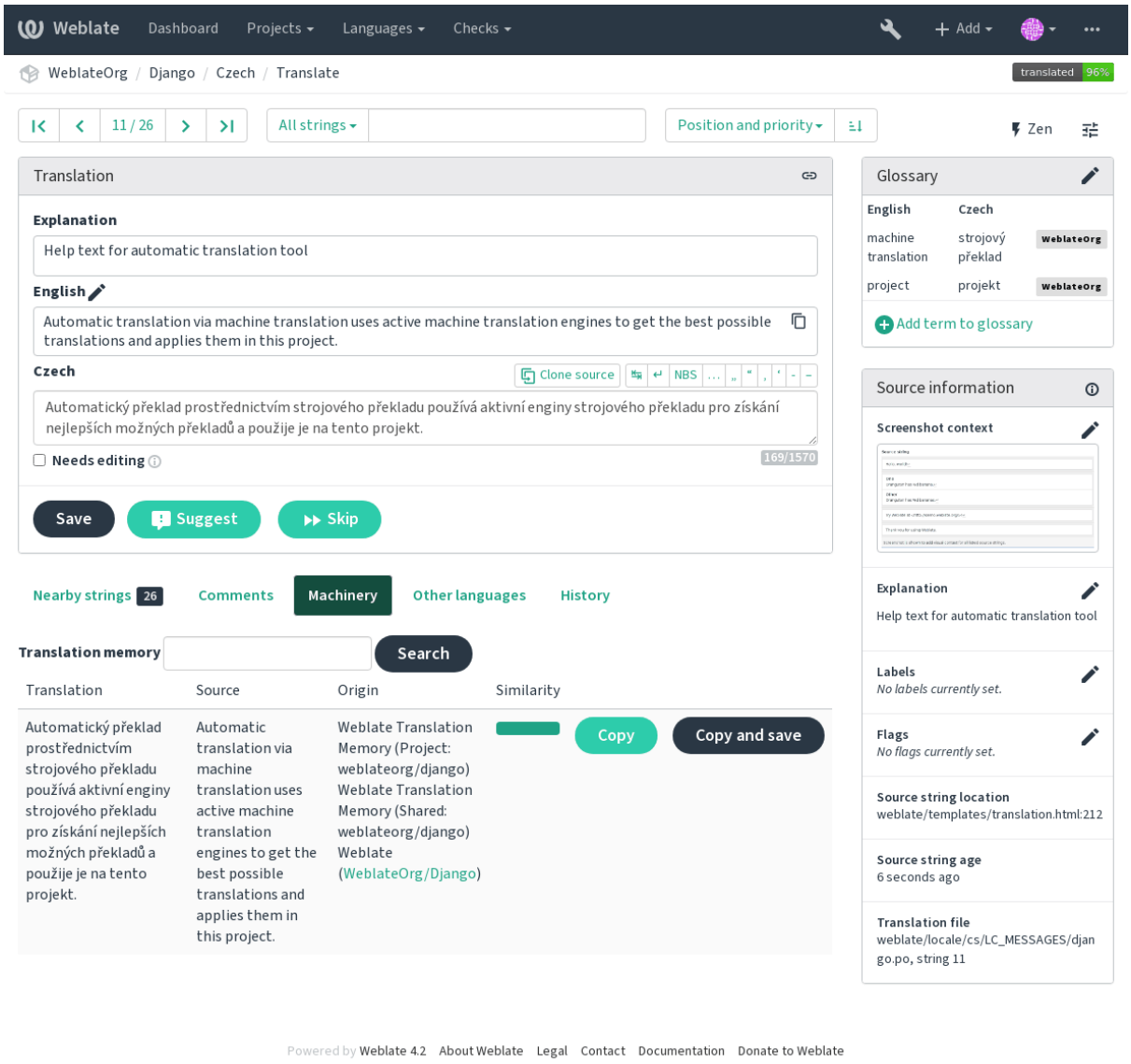
Use the explanation to clarify scope or usage of the translation. You can use Markdown to include links and other markup.

Visual context for strings

Добавлено в версии 2.9.

You can upload a screenshot showing a given source string in use within your program. This helps translators understand where it is used, and how it should be translated.

The uploaded screenshot is shown in the translation context sidebar:



In addition to *Рецензирование строк*, screenshots have a separate management interface under the *Tools* menu. Upload screenshots, assign them to source strings manually, or use optical character recognition to do so.

Once a screenshot is uploaded, this interface handles management and source string association:

Weblate Dashboard Projects Languages Checks Add

WeblateOrg / Django / Screenshots / Automatic translation

Screenshot has been uploaded, you can now assign it to source strings.

Assigned source strings

Source string	Context	Location	Assigned screenshots	Actions
No source strings are currently assigned!				

Screenshot is shown to add visual context for all listed source strings.

Assign source strings

Source string	Context	Location	Assigned screenshots	Actions
No new matching source strings found.				

Source string search Search Automatically recognize

Image

Source string

Hello, world!

One
Orangutan has %d banana.

Other
Orangutan has %d bananas.

Try Weblate at <http://demo.weblate.org/>!

Thank you for using Weblate.

Screenshot is shown to add visual context for all listed source strings.

Edit screenshot

Screenshot name

Automatic translation

Image

Currently: screenshots/screenshot.png
Change:
Choose File No file chosen
Upload JPEG or PNG images up to 2000x2000 pixels.

Save

Screenshot details

Created	now
Uploaded by	testuser

Delete screenshot

Deleting screenshot will remove it from all associated source strings.

Delete

Powered by Weblate 4.2 About Weblate Legal Contact Documentation Donate to Weblate

2.12 Проверки и исправления

2.12.1 Пользовательские автоматические исправления

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

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

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

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#

from django.utils.translation import gettext_lazy as _

from weblate.trans.autofixes.base import AutoFix


class ReplaceFooWithBar(AutoFix):
    """Replace foo with bar."""

    name = _("Foobar")

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

To install custom checks, provide a fully-qualified path to the Python class in the `AUTOFIX_LIST`, see *Custom quality checks, addons and auto-fixes*.

2.12.2 Настройка поведения

You can fine-tune the behavior of Weblate (mostly checks) for each source string (in source strings review, see *Additional info on source strings*) or in the *Конфигурация компонента* (*Флаги перевода*). Some file formats also allow to specify flags directly in the format (see *Поддерживаемые форматы файлов*).

The flags are comma-separated, the parameters are separated with colon. You can use quotes to include whitespace or special chars in the string. For example:

```
placeholders:"special:value":"other value", regex:.*
```

Вот список принимаемых в настоящее время флагов:

rst-text Treat a text as an RST document, affects *Перевод без изменения*.

md-text Treat text as a Markdown document.

dos-eol Uses DOS end-of-line markers instead of Unix ones (`\r\n` instead of `\n`).

url The string should consist of only a URL.

safe-html The string should be HTML safe, see *Небезопасный HTML-код*.

read-only The string is read-only and should not be edited in Weblate, see *Строки только для чтения*.

priority:N Priority of the string. Higher priority strings are presented first for translation. The default priority is 100, the higher priority a string has, the earlier it is offered for translation.

max-length:N Limit the maximal length for a string to N characters, see *Максимальная длина перевода*

xml-text Treat text as XML document, affects *Синтаксис XML* and *XML-разметка*.

font-family:NAME Define font-family for rendering checks, see *Управление шрифтами*.

font-weight:WEIGHT Define font-weight for rendering checks, see *Управление шрифтами*.

font-size:SIZE Define font-size for rendering checks, see *Управление шрифтами*.

font-spacing:SPACING Define font-spacing for rendering checks, see *Управление шрифтами*.

placeholders:NAME Placeholder strings expected in translation, see *Символы-заполнители*.

**replacements:FROM:TO:FROM2:TO2...`** Replacements to perform when checking resulting text parameters (for example in *Максимальный размер перевода* or *Максимальная длина перевода*). The typical use case for this is to expand placeables to ensure that the text fits even with long names, for example: `replacements:%s:"John Doe"`.

regex:REGEX Регулярное выражение для сопоставления с переводом, смотрите раздел *Регулярное выражение*.

python-format, c-format, php-format, python-brace-format, javascript-format, c-sharp-format, ja
Treats all strings like format strings, affects *Форматированные строки*, *Форматированные строки*, *Форматированные строки*, *Форматированные строки*, *Форматированные строки*, *Форматированные строки*, *Форматированные строки*, *Форматированные строки*, *Форматированные строки*, *Форматированные строки*, *Форматированные строки*, *Перевод без изменения*.

strict-same Make «Unchanged translation» avoid using built-in words blacklist, see *Перевод без изменения*.

ignore-bbcode Skip the «BBcode markup» quality check.

ignore-duplicate Пропускает проверку качества «Повторяющиеся слова».

ignore-double-space Пропускает проверку качества «Двойной пробел».

ignore-angularjs-format Пропускает проверку качества «Интерполяционная строка AngularJS».

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

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

ignore-es-format Пропускает проверку качества «Шаблонные литералы ECMAScript».

ignore-i18next-interpolation Skip the «i18next interpolation» quality check.

ignore-java-format Skip the «Java format» quality check.

ignore-java-messageformat Skip the «Java MessageFormat» quality check.

ignore-javascript-format Skip the «JavaScript format» quality check.

ignore-percent-placeholders Skip the «Percent placeholders» quality check.

ignore-perl-format Skip the «Perl format» quality check.

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

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

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

ignore-qt-format Skip the «Qt format» quality check.

ignore-qt-plural-format Skip the «Qt plural format» quality check.

ignore-ruby-format Skip the «Ruby format» quality check.

ignore-translated Пропускает проверку качества «Уже переведено».

ignore-inconsistent Пропускает проверку качества «Противоречия».

ignore-kashida Пропускает проверку качества «Используется кашида».

ignore-md-link Пропускает проверку качества «Ссылки Markdown».

ignore-md-reflink Пропускает проверку качества «Сноски Markdown».

ignore-md-syntax Пропускает проверку качества «Синтаксис Markdown».

ignore-max-length Пропускает проверку качества «Максимальная длина перевода».

ignore-max-size Пропускает проверку качества «Максимальный размер перевода».

ignore-escaped-newline Пропускает проверку качества «Несовпадающие n».

ignore-end-colon Пропускает проверку качества «Несовпадающие двоеточия».

ignore-end-ellipsis Пропускает проверку качества «Несовпадающие многоточия».

ignore-end-exclamation Пропускает проверку качества «Несовпадающие восклицательные знаки».

ignore-end-stop Пропускает проверку качества «Несовпадающие точки».

ignore-end-question Пропускает проверку качества «Несовпадающие вопросительные знаки».

ignore-end-semicolon Пропускает проверку качества «Несовпадающие точки с запятой».

ignore-newline-count Skip the «Mismatching line breaks» quality check.

ignore-plurals Пропускает проверку качества «Нет форм множественного числа».

ignore-placeholders Skip the «Placeholders» quality check.

ignore-punctuation-spacing Skip the «Punctuation spacing» quality check.

ignore-regex Skip the «Regular expression» quality check.

ignore-same-plurals Пропускает проверку качества «Одинаковые формы множественного числа».

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

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

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

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

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

ignore-safe-html Skip the «Unsafe HTML» quality check.

ignore-url Skip the «URL» quality check.

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

ignore-xml-invalid Skip the «XML syntax» quality check.

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

ignore-ellipsis Skip the «Ellipsis» quality check.

ignore-long-untranslated Skip the «Long untranslated» quality check.

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

ignore-unnamed-format Skip the «Multiple unnamed variables» quality check.

ignore-optional-plural Skip the «Unpluralised» 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 *Конфигурация компонента* settings, per source string settings and in the translation file itself (for example in GNU gettext).

2.12.3 Enforcing checks

Добавлено в версии 3.11.

You can configure a list of checks which can not be ignored by setting *Enforced checks* in *Конфигурация компонента*. Each listed check can not be ignored in the user interface and any string failing this check is marked as *Needs editing* (see *Состояния перевода*).

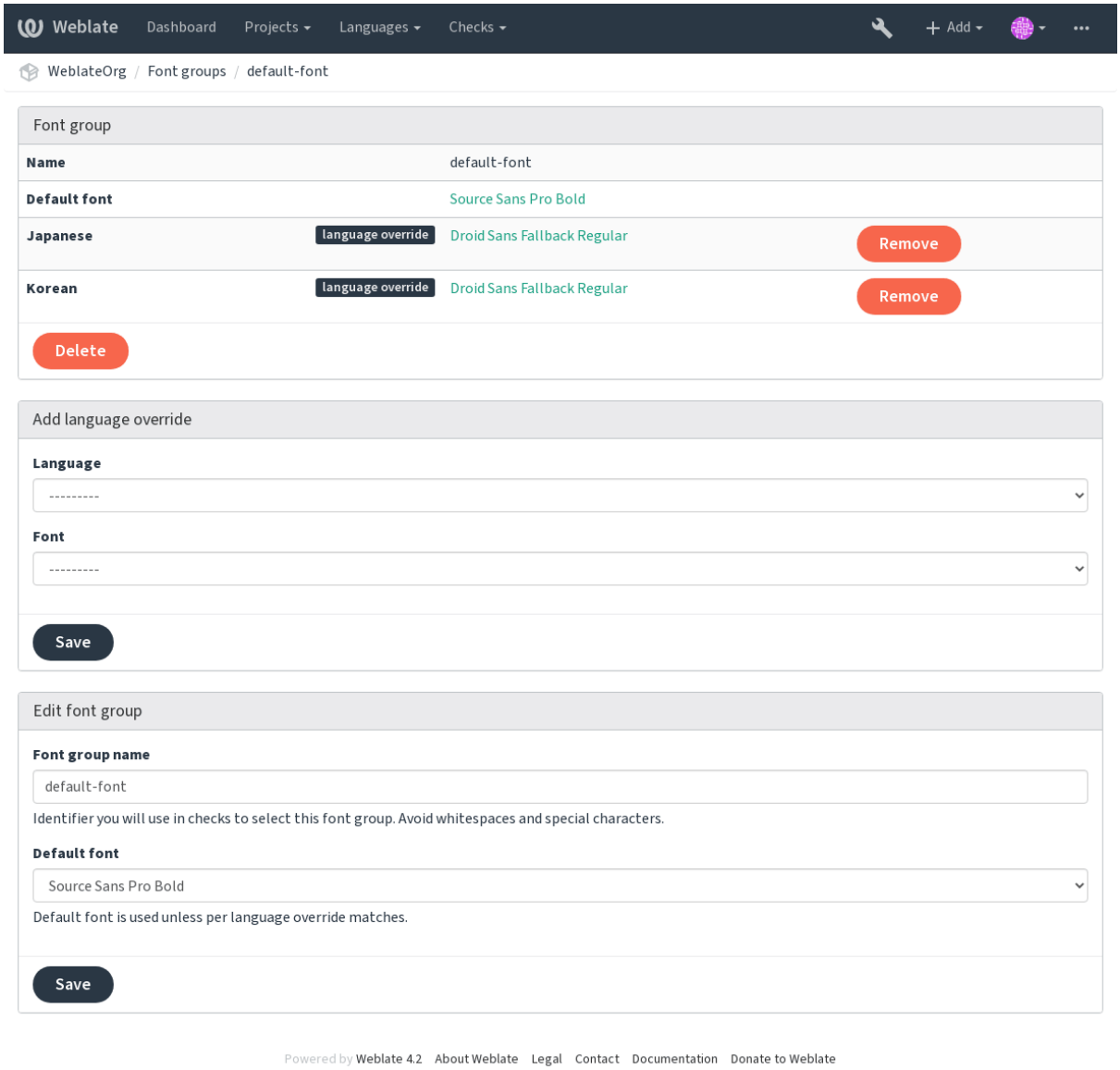
2.12.4 Управление шрифтами

Добавлено в версии 3.7.

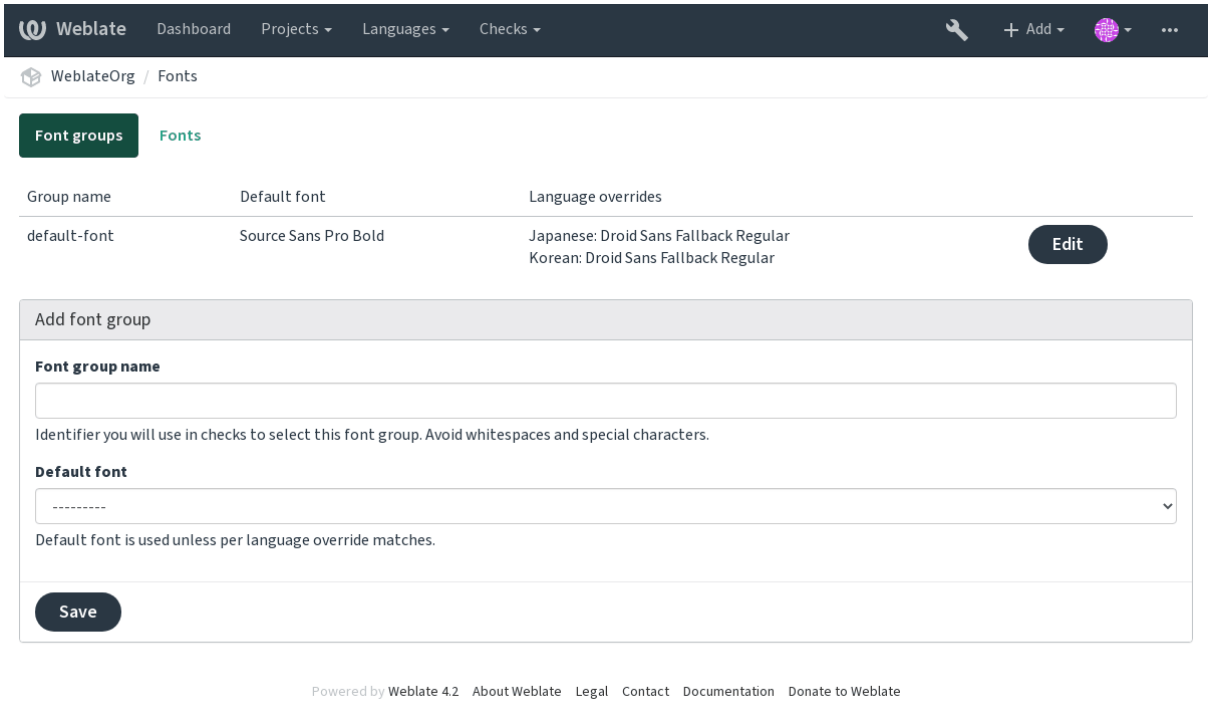
The *Максимальный размер перевода* check used to calculate dimensions of the rendered text needs font info to be selected, which can be done in the Weblate font management tool in *Fonts* under the *Manage* menu of your translation project.

TrueType or OpenType fonts can be uploaded, set up font-groups and use those in the check.

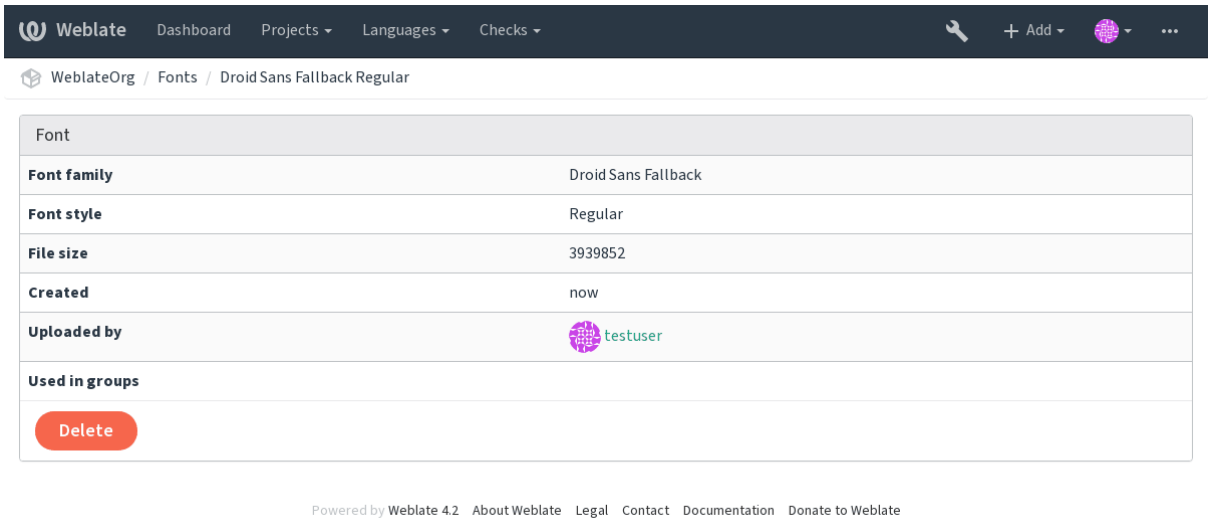
The font-groups allow you to define different fonts for different languages, which is typically needed for non-latin languages:



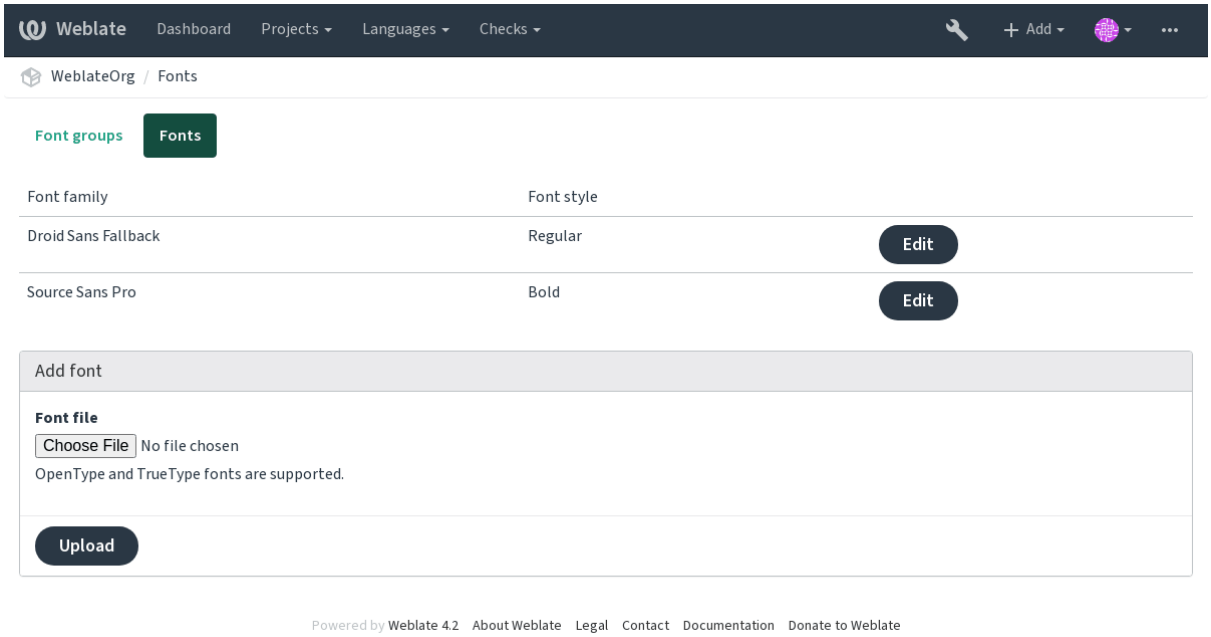
The font-groups are identified by name, which can not contain whitespace or special characters, so that it can be easily used in the check definition:



Font-family and style is automatically recognized after uploading them:



You can have a number of fonts loaded into Weblate:



To use the fonts for checking the string length, pass it the appropriate flags (see *Настройка поведения*). You will probably need the following ones:

`max-size:500` Defines maximal width.

`font-family:ubuntu` Defines font group to use by specifying its identifier.

`font-size:22` Defines font size.

2.12.5 Writing own checks

A wide range of quality checks are built-in, (see *Проверки качества*), though they might not cover everything you want to check. The list of performed checks can be adjusted using `CHECK_LIST`, and you can also add custom checks.

1. Subclass the *weblate.checks.Check*
2. Set a few attributes.
3. Implement either the `check` (if you want to deal with plurals in your code) or the `check_single` method (which does it for you).

Some examples:

To install custom checks, provide a fully-qualified path to the Python class in the `CHECK_LIST`, see *Custom quality checks, addons and auto-fixes*.

Checking translation text does not contain «foo»

This is a pretty simple check which just checks whether the translation is missing the string «foo».

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Simple quality check example."""

from django.utils.translation import gettext_lazy as _

from weblate.checks.base import TargetCheck


class FooCheck(TargetCheck):

    # Used as identifier for check, should be unique
    # Has to be shorter than 50 characters
    check_id = "foo"

    # Short name used to display failing check
    name = _("Foo check")

    # Description for failing check
    description = _("Your translation is foo")

    # Real check code
    def check_single(self, source, target, unit):
        return "foo" in target
```

Checking that Czech translation text plurals differ

Check using language info to verify the two plural forms in Czech language are not same.

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Quality check example for Czech plurals."""

from django.utils.translation import gettext_lazy as _

from weblate.checks.base import TargetCheck
```

(continues on next page)

(продолжение с предыдущей страницы)

```
class PluralCzechCheck(TargetCheck):

    # Used as identifier for check, should be unique
    # Has to be shorter than 50 characters
    check_id = "foo"

    # Short name used to display failing check
    name = _("Foo check")

    # Description for failing check
    description = _("Your translation is foo")

    # Real check code
    def check_target_unit(self, sources, targets, unit):
        if self.is_language(unit, ("cs",)):
            return targets[1] == targets[2]
        return False

    def check_single(self, source, target, unit):
        """We don't check target strings here."""
        return False
```

2.13 Машинный перевод

Built-in support for several machine translation services and can be turned on by the administrator using *MT_SERVICES* for each one. They come subject to their terms of use, so ensure you are allowed to use them how you want.

The source language can be configured at *Project configuration*.

2.13.1 amaGama

Special installation of *tmserver* run by the authors of Virtaal.

Turn on this service by adding `weblate.machinery.tmserver.AmagamaTranslation` to *MT_SERVICES*.

См.также:

Installing amaGama, Amagama, amaGama Translation Memory

2.13.2 Apertium

A libre software machine translation platform providing translations to a limited set of languages.

The recommended way to use Apertium is to run your own Apertium-APy server.

Turn on this service by adding `weblate.machinery.apertium.ApertiumAPYTranslation` to *MT_SERVICES* and set *MT_APERTIUM_APY*.

См.также:

MT_APERTIUM_APY, Apertium website, Apertium APy documentation

2.13.3 AWS

Добавлено в версии 3.1.

Amazon Translate is a neural machine translation service for translating text to and from English across a breadth of supported languages.

1. Turn on this service by adding `weblate.machinery.aws.AWSTranslation` to *MT_SERVICES*.
2. Install the *boto3* module.
3. Configure Weblate.

См.также:

MT_AWS_REGION, *MT_AWS_ACCESS_KEY_ID*, *MT_AWS_SECRET_ACCESS_KEY* Amazon Translate Documentation

2.13.4 Машинный перевод через API Baidu

Добавлено в версии 3.2.

Сервис машинного перевода, предоставляемый Baidu.

Этот сервис использует API, и чтобы его использовать, вам нужно получить от Baidu идентификатор и ключ API.

Turn on this service by adding `weblate.machinery.baidu.BaiduTranslation` to *MT_SERVICES* and set *MT_BAIDU_ID* and *MT_BAIDU_SECRET*.

См.также:

MT_BAIDU_ID, *MT_BAIDU_SECRET* Baidu Translate API

2.13.5 DeepL

Добавлено в версии 2.20.

DeepL is paid service providing good machine translation for a few languages. You need to purchase *DeepL API* subscription or you can use legacy *DeepL Pro (classic)* plan.

Turn on this service by adding `weblate.machinery.deepl.DeepLTranslation` to *MT_SERVICES* and set *MT_DEEPL_KEY*.

Подсказка: In case you have subscription for CAT tools, you are supposed to use «v1 API» instead of default «v2» used by Weblate (it is not really an API version in this case). You can toggle this by *MT_DEEPL_API_VERSION*.

См.также:

MT_DEEPL_KEY, *MT_DEEPL_API_VERSION*, DeepL website, DeepL pricing, DeepL API documentation

2.13.6 Glosbe

Free dictionary and translation memory for almost every living language.

The API is gratis to use, but subject to the used data source license. There is a limit of calls that may be done from one IP in a set period of time, to prevent abuse.

Turn on this service by adding `weblate.machinery.glosbe.GlosbeTranslation` to *MT_SERVICES*.

См.также:

Веб-сайт Glosbe

2.13.7 Google Translate

Machine translation service provided by Google.

This service uses the Google Translation API, and you need to obtain an API key and turn on billing in the Google API console.

To turn on this service, add `weblate.machinery.google.GoogleTranslation` to *MT_SERVICES* and set *MT_GOOGLE_KEY*.

См.также:

MT_GOOGLE_KEY, Google translate documentation

2.13.8 Google Translate API V3 (расширенный)

Machine translation service provided by Google Cloud services.

This service differs from the former one in how it authenticates. To enable service, add `weblate.machinery.googlev3.GoogleV3Translation` to *MT_SERVICES* and set

- *MT_GOOGLE_CREDENTIALS*
- *MT_GOOGLE_PROJECT*

If *location* fails, you may also need to specify *MT_GOOGLE_LOCATION*.

См.также:

MT_GOOGLE_CREDENTIALS, *MT_GOOGLE_PROJECT*, *MT_GOOGLE_LOCATION* Google translate documentation

2.13.9 Microsoft Cognitive Services Translator

Добавлено в версии 2.10.

Machine translation service provided by Microsoft in Azure portal as a one of Cognitive Services.

Weblate implements Translator API V3.

To enable this service, add `weblate.machinery.microsoft.MicrosoftCognitiveTranslation` to *MT_SERVICES* and set *MT_MICROSOFT_COGNITIVE_KEY*.

Translator Text API V2

The key you use with Translator API V2 can be used with API 3.

Translator Text API V3

You need to register at Azure portal and use the key you obtain there. With new Azure keys, you also need to set *MT_MICROSOFT_REGION* to locale of your service.

См.также:

MT_MICROSOFT_COGNITIVE_KEY, *MT_MICROSOFT_REGION*, Cognitive Services - Text Translation API, Microsoft Azure Portal

2.13.10 Microsoft Terminology Service (Терминологическая служба Майкрософт)

Добавлено в версии 2.19.

API терминологической службы Майкрософт позволяет программно получать доступ к терминологии, определениям и строкам пользовательского интерфейса (UI), доступным на языковом портале, через веб-сервис.

Turn this service on by adding `weblate.machinery.microsoftterminology.MicrosoftTerminologyService` to *MT_SERVICES*.

См.также:

API терминологической службы Майкрософт

2.13.11 ModernMT

Добавлено в версии 4.2.

Turn this service on by adding `weblate.machinery.modernmt.ModernMTTranslation` to *MT_SERVICES* and configure *MT_MODERNMT_KEY*.

См.также:

ModernMT API, *MT_MODERNMT_KEY*, *MT_MODERNMT_URL*

2.13.12 MyMemory

Huge translation memory with machine translation.

Free, anonymous usage is currently limited to 100 requests/day, or to 1000 requests/day when you provide a contact e-mail address in *MT_MYMEMORY_EMAIL*. You can also ask them for more.

Turn on this service by adding `weblate.machinery.mymemory.MyMemoryTranslation` to *MT_SERVICES* and set *MT_MYMEMORY_EMAIL*.

См.также:

MT_MYMEMORY_EMAIL, *MT_MYMEMORY_USER*, *MT_MYMEMORY_KEY*, MyMemory website

2.13.13 NetEase Sight API machine translation

Добавлено в версии 3.3.

Machine translation service provided by Netease.

This service uses an API, and you need to obtain key and secret from NetEase.

Turn on this service by adding `weblate.machinery.youdao.NeteaseSightTranslation` to *MT_SERVICES* and set *MT_NETEASE_KEY* and *MT_NETEASE_SECRET*.

См.также:

MT_NETEASE_KEY, *MT_NETEASE_SECRET* Netease Sight Translation Platform

2.13.14 tmserver

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

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

2. Turn on this service by adding `weblate.machinery.tmserver.TMServerTranslation` to *MT_SERVICES*.

```
build_tmdb -d /var/lib/tm/db -s en -t cs locale/cs/LC_MESSAGES/django.po
build_tmdb -d /var/lib/tm/db -s en -t de locale/de/LC_MESSAGES/django.po
build_tmdb -d /var/lib/tm/db -s en -t fr locale/fr/LC_MESSAGES/django.po
```

3. Start tmserver to listen to your requests:

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

4. Configure Weblate to talk to it:

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

См.также:

MT_TMSERVER, tmserver Installing amaGama, Amagama, Amagama Translation Memory

2.13.15 Yandex Translate

Machine translation service provided by Yandex.

This service uses a Translation API, and you need to obtain an API key from Yandex.

Turn on this service by adding `weblate.machinery.yandex.YandexTranslation` to *MT_SERVICES*, and set *MT_YANDEX_KEY*.

См.также:

MT_YANDEX_KEY, Yandex Translate API, Powered by Yandex.Translate

2.13.16 Youdao Zhiyun API machine translation

Добавлено в версии 3.2.

Machine translation service provided by Youdao.

This service uses an API, and you need to obtain an ID and an API key from Youdao.

Turn on this service by adding `weblate.machinery.youdao.YoudaoTranslation` to *MT_SERVICES* and set *MT_YOUDAO_ID* and *MT_YOUDAO_SECRET*.

См.также:

MT_YOUDAO_ID, *MT_YOUDAO_SECRET* Youdao Zhiyun Natural Language Translation Service

2.13.17 Weblate

Weblate can be the source of machine translations as well. It is based on the Woosh fulltext engine, and provides both exact and inexact matches.

Turn on these services by adding `weblate.machinery.weblatetm.WeblateTranslation` to *MT_SERVICES*.

2.13.18 Weblate Translation Memory

Добавлено в версии 2.20.

The *Память переводов* can be used as a source for machine translation suggestions as well.

Turn on these services by adding `weblate.memory.machine.WeblateMemory` to the *MT_SERVICES*. This service is turned on by default.

2.13.19 SAP Translation Hub

Machine translation service provided by SAP.

You need to have a SAP account (and enabled the SAP Translation Hub in the SAP Cloud Platform) to use this service.

Turn on this service by adding `weblate.machinery.saptranslationhub.SAPTranslationHub` to *MT_SERVICES* and set the appropriate access to either sandbox or the productive API.

Примечание: To access the Sandbox API, you need to set *MT_SAP_BASE_URL* and *MT_SAP_SANDBOX_APIKEY*.

To access the productive API, you need to set *MT_SAP_BASE_URL*, *MT_SAP_USERNAME* and *MT_SAP_PASSWORD*.

См.также:

MT_SAP_BASE_URL, *MT_SAP_SANDBOX_APIKEY*, *MT_SAP_USERNAME*, *MT_SAP_PASSWORD*, *MT_SAP_USE_MT* SAP Translation Hub API

2.13.20 Custom machine translation

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

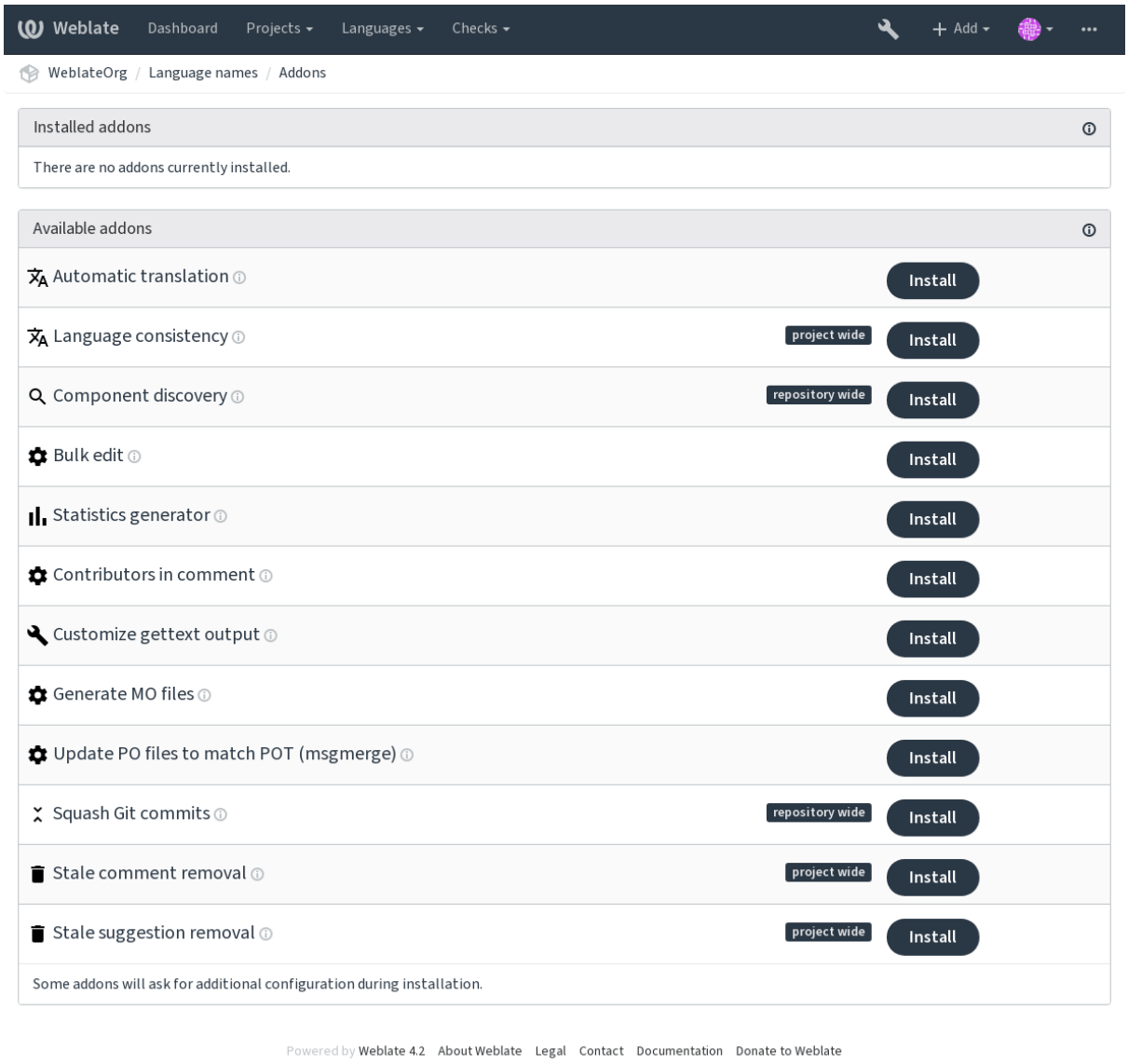
```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Machine translation example."""

import dictionary

from weblate.machinery.base import MachineTranslation


class SampleTranslation(MachineTranslation):
    """Sample machine translation interface."""

    name = "Sample"

    def download_languages(self):
        """Return list of languages your machine translation supports."""
        return {"cs"}

    def download_translations(self, source, language, text, unit, user, search):
        """Return tuple with translations."""
        for t in dictionary.translate(text):
            yield {"text": t, "quality": 100, "service": self.name, "source": text}
```

You can list own class in *MT_SERVICES* and Weblate will start using that.

2.14 Надстройки

Добавлено в версии 2.19.

Надстройки предоставляют способы настройки рабочего процесса перевода. Они могут быть установлены в виде компонента перевода и работать за кулисами. Управление надстройками доступно для администраторов из меню *Управлять* ↓ *Надстройки* соответствующего компонента перевода.



2.14.1 Встроенные надстройки

Автоматический перевод

Добавлено в версии 3.9.

Автоматически переводит строки при помощи машинного перевода или других компонентов.

Эта надстройка срабатывает автоматически при появлении в компоненте новых строк.

См.также:

Автоматический перевод, *Keeping translations same across components*

CDN локализации JavaScript'а

Добавляет CDN локализации для локализации JavaScript'а или HTML'я.

Его можно использовать для локализации статических HTML-страниц или для загрузки локализации в код JavaScript.

Upon installation the addon generates unique URL for your component which you can include in the HTML documents to get them localized. See *Translating HTML and JavaScript using Weblate CDN* for more details.

См.также:

Configuring Weblate CDN addon, *Translating HTML and JavaScript using Weblate CDN*, *String extraction for Weblate CDN*, *HTML localization using Weblate CDN*

Очистка файлов перевода

Обновить все файлы переводов до соответствия одноязычному базовому файлу. Для большинства форматов файлов это означает удаление устарелых ключей переводов, которых больше нет в базовом файле.

Языковая согласованность

Убедитесь, что все компоненты в рамках одного проекта переведены на все языки проекта.

Он создает пустые переводы на языках, у которых есть недобавленные компоненты.

Отсутствующие языки проверяются каждые 24 часа и при добавлении нового языка в Weblate.

В отличие от большинства других надстроек, эта влияет на весь проект.

Подсказка: Auto-translate the newly added strings with *Автоматический перевод*.

Обнаружение компонента

Автоматически добавляет или удаляет компоненты проекта, основываясь на изменениях файлов в системе контроля версий.

It is triggered on every VCS update, and otherwise similar to the `import_project` management command. This way you can track multiple translation components within one VCS.

Create one main component least likely to disappear in the future, and others will employ *Внутренние URL-адреса Weblate* to it as a VCS configuration, and configure it to find all components in it.

The matching is done using regular expressions, where power is a tradeoff for complexity in configuration. Some examples for common use cases can be found in the addon help section.

Once you hit *Save*, a preview of matching components will be presented, from where you can check whether the configuration actually matches your needs:

Weblate Dashboard Projects Languages Checks + Add

WeblateOrg / Language names / Addons / Component discovery

Configure addon

- Please review and confirm the matched components.

Component	Matched files
Following components would be created	
Djangojs	weblate/locale/hu/LC_MESSAGES/djangojs.po (hu) weblate/locale/cs/LC_MESSAGES/djangojs.po (cs) weblate/locale/he/LC_MESSAGES/djangojs.po (he)
Django	weblate/locale/cs/LC_MESSAGES/django.po (cs) weblate/locale/he/LC_MESSAGES/django.po (he) weblate/locale/hu/LC_MESSAGES/django.po (hu)

☐ **I confirm the above matches look correct**

Regular expression to match translation files against

weblate/locale/(?P<language>[^/]*)/LC_MESSAGES/(?P<component>[^/]*)\.po

File format

gettext PO file

Customize the component name

{{ component|title }}

Define the monolingual base filename

Leave empty for bilingual translation files.

Define the base file for new translations

weblate/locale/{{ component }}.pot

Filename of file used for creating new translations. For gettext choose .pot file.

Language filter

^(cs|he|hu)$

Regular expression to filter translation against when scanning for filemask.

☑ **Clone addons from the main component to the newly created ones**

☐ **Remove components for inexistant files**

The regular expression to match translation files has to contain two named groups to match component and language, some examples:

Regular expression	Example matched files	Description
(?P<language>[^/.]*)/(?P<component>[^/]*)\.po	cs/application.po cs/website.po de/application.po de/website.po	One folder per language containing translation files for components.
locale/(?P<language>[^/.]*)/LC_MESSAGES/(?P<component>[^/]*)\.po	locale/cs/LC_MESSAGES/application.po locale/cs/LC_MESSAGES/website.po locale/de/LC_MESSAGES/application.po locale/de/LC_MESSAGES/website.po	Usual structure for storing gettext PO files.
src/locale/(?P<component>[^/]*)\.(?P<language>[^/.]*)\.po	src/locale/application.cs.po src/locale/website.cs.po src/locale/application.de.po src/locale/website.de.po	Using both component and language name within filename.
locale/(?P<language>[^/.]*)/(?P<component>[^/]*)/(?P=language)\.po	locale/cs/application/cs.po locale/cs/website/cs.po locale/de/application/de.po locale/de/website/de.po	Using language in both path and filename.
res/values-(?P<language>[^/.]*)/strings-(?P<component>[^/]*)\.xml	res/values-cs/strings-about.xml res/values-cs/strings-help.xml res/values-de/strings-about.xml res/values-de/strings-help.xml	Android resource strings, split into several files.

You can use Django template markup in both component name and the monolingual base filename, for example:

`{{ component }}`
Component filename match
`{{ component|title }}`
Component filename with upper case first letter

Save

Powered by Weblate 4.2 About Weblate Legal Contact Documentation Donate to Weblate

См.также:

Template markup

Массовая правка

Добавлено в версии 3.11.

Позволяет массово править флаги, метки или состояния для строк.

Automating the labeling of new strings can be useful (start out with search query `NOT has:label` and add desired labels till all strings are properly labeled). You can also carry out any other automated operations for Weblate metadata.

См.также:

Массовая правка

Отмечать неизменённые переводы как «На правку»

Добавлено в версии 3.1.

При импорте из VCS подлежащей переводу строки и ее совпадении с исходной, она помечается в Weblate как «На правку». Это особенно полезно для файловых форматов, которые включают в себя все строки, даже если они не переведены.

Отмечать новые исходные строки как «На правку»

Когда новая исходная строка импортируется из VCS, она помечается как «На правку» в Weblate. Таким образом, вы можете легко фильтровать и править исходные строки, добавленные разработчиками.

Пометить новые переводы как «На правку»

При импорте из VCS подлежащей переводу строки она помечается в Weblate как «На правку». Таким образом, вы можете легко фильтровать и редактировать переводы, созданные разработчиками.

Генератор статистики

Генерирует файл, содержащий детальную информацию о переводе.

You can use Django template in both filename and content, see *Template markup* for a detailed markup description.

Например, создание сводного файла для каждого перевода:

Имя сгенерированного файла `locale/{{ language_code }}.json`

Содержимое

```
{
   "language": "{{ language_code }}",
   "strings": "{{ stats.all }}",
   "translated": "{{ stats.translated }}",
   "last_changed": "{{ stats.last_changed }}",
   "last_author": "{{ stats.last_author }}",
}
```

См.также:

Template markup

Участники в комментарии

Обновлять комментарий в заголовке PO-файла для включения имён участников и лет участия.

Заголовок файла PO будет содержать список внесших вклад и годы участия:

```
# Michal Čihař <michal@cihar.com>, 2012, 2018, 2019, 2020.
# Pavel Borecki <pavel@example.com>, 2018, 2019.
# Filip Hron <filip@example.com>, 2018, 2019.
# anonymous <noreply@weblate.org>, 2019.
```

Обновлять переменную ALL_LINGUAS в файле «configure»

Updates the ALL_LINGUAS variable in `configure`, `configure.in` or any `configure.ac` files, when a new translation is added.

Настроить вывод gettext

Позволяет настроить поведение вывода gettext, например, заворот строк.

Он предлагает следующие варианты:

- Переносить строки длиной 77 символов и у символов перевода строки
- Переносить строки только в конце строки
- Не переносить строки

Примечание: По умолчанию gettext переносит строки из 77 символов и символы новой строки. С параметром –no-wrap переносится только на символах новой строки.

Обновлять LINGUAS-файл

Обновляет LINGUAS-файл при добавлении нового перевода.

Генерировать MO-файлы

Автоматически генерирует MO-файл для каждого изменённого PO-файла.

Обновлять PO-файлы до соответствия POT (msgmerge)

Обновляет все PO-файлы, используя msgmerge, приводя их в соответствие с POT-файлом. Это происходит всякий раз, когда новые изменения извлекаются из вышестоящего репозитория.

Склеивать Git-коммиты

Склеивать Git-коммиты до отправки изменений.

Вы можете выбрать один из следующих режимов:

Добавлено в версии 3.4.

- Все коммиты в один
- По языку
- По файлу

Добавлено в версии 3.5.

- По автору

Original commit messages are kept, but authorship is lost unless «Per author» is selected, or the commit message is customized to include it.

Добавлено в версии 4.1.

The original commit messages can optionally be overridden with a custom commit message.

Trailers (commit lines like `Co-authored-by: ...`) can optionally be removed from the original commit messages and appended to the end of the squashed commit message. This also generates proper `Co-authored-by:` credit for every translator.

Настроить вывод JSON

Позволяет подправить поведение вывода JSON, например, отступы или сортировку ключей.

Форматирует файл свойств Java

Сортирует файл свойств Java.

Удаление устаревшего комментария

Добавлено в версии 3.7.

Устанавливает сроки удаления комментариев.

This can be useful to remove old comments which might have become outdated. Use with care as comment being old does not mean they have lost their importance.

Устаревшее предложение удалено

Добавлено в версии 3.7.

Устанавливает сроки удаления предложений.

This can be very useful in connection with suggestion voting (see *Коллегиальное рецензирование*) to remove suggestions which don't receive enough positive votes in a given timeframe.

Обновить файлы RESX

Добавлено в версии 3.9.

Обновлять все файлы перевода, приводя их в соответствие с одноязычным исходным базовым файлом. Неиспользуемые строки удаляются, а новые добавляются как копии исходных строк.

Подсказка: Use *Очистка файлов перевода* if you only want to remove stale translation keys.

Настроить вывод YAML

Добавлено в версии 3.10.2.

Позволяет подправить поведение вывода YAML, например, длину строк и символы перевода строк.

2.14.2 Customizing list of addons

The list of addons is configured by *WEBLATE_ADDONS*. To add another addon, simply include class absolute name in this setting.

2.14.3 Writing addon

You can write your own addons too, all you need to do is subclass `BaseAddon`, define the addon metadata and implement a callback which will do the processing.

Here is an example addon:

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#


from django.utils.translation import gettext_lazy as _

from weblate.addons.base import BaseAddon
from weblate.addons.events import EVENT_PRE_COMMIT


class ExampleAddon(BaseAddon):
    # Filter for compatible components, every key is
    # matched against property of component
    compat = {"file_format": {"po", "po-mono"}}
    # List of events addon should receive
    events = (EVENT_PRE_COMMIT,)
    # Addon unique identifier
    name = "weblate.example.example"
    # Verbose name shown in the user interface
    verbose = _("Example addon")
    # Detailed addon description
    description = _("This addon does nothing it is just an example.")

    # Callback to implement custom behavior
    def pre_commit(self, translation, author):
        return
```

2.14.4 Executing scripts from addon

Addons can also be used to execute external scripts. This used to be integrated in Weblate, but now you have to write some code to wrap your script with an addon.

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#
"""Example pre commit script."""


from django.utils.translation import gettext_lazy as _

from weblate.addons.events import EVENT_PRE_COMMIT
from weblate.addons.scripts import BaseScriptAddon


class ExamplePreAddon(BaseScriptAddon):
    # Event used to trigger the script
    events = (EVENT_PRE_COMMIT,)
    # Name of the addon, has to be unique
    name = "weblate.example.pre"
    # Verbose name and long descrption
    verbose = _("Execute script before commit")
    description = _("This addon executes a script.")

    # Script to execute
    script = "/bin/true"
    # File to add in commit (for pre commit event)
    # does not have to be set
    add_file = "po/{{ language_code }}.po"
```

For installation instructions see *Custom quality checks, addons and auto-fixes*.

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

Additionally, the following environment variables are available:

WL_VCS
Version control system used.

WL_REPO
Upstream repository URL.

WL_PATH
Absolute path to VCS repository.

WL_BRANCH
Добавлено в версии 2.11.

Repository branch configured in the current component.

WL_FILEMASK
Filemask for current component.

WL_TEMPLATE
Filename of template for monolingual translations (can be empty).

WL_NEW_BASE
Добавлено в версии 2.14.

Filename of the file used for creating new translations (can be empty).

WL_FILE_FORMAT
Fileformat used in current component.

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

WL_PREVIOUS_HEAD
Previous HEAD on update (available only when running post update hook).

WL_COMPONENT_SLUG
Добавлено в версии 3.9.

Плашка компонента, используемая для построения URL-адреса.

WL_PROJECT_SLUG
Добавлено в версии 3.9.

Плашка проекта, используемая для построения URL-адреса.

WL_COMPONENT_NAME
Добавлено в версии 3.9.

Имя компонента.

WL_PROJECT_NAME
Добавлено в версии 3.9.

Название проекта.

WL_COMPONENT_URL
Добавлено в версии 3.9.

Component URL.

WL_ENGAGE_URL
Добавлено в версии 3.9.

Project engage URL.

См.также:

Конфигурация компонента

Post update repository processing

Post update repository processing can be used to update translation files when the VCS upstream source changes. To achieve this, please remember that Weblate only sees files committed to the VCS, so you need to commit changes as a part of the script.

For example with Gulp you can do it using following code:

```
#! /bin/sh
gulp --gulpfile gulp-i18n-extract.js
git commit -m 'Update source strings' src/languages/en.lang.json
```

Pre commit processing of translations

Use the commit script to automatically make changes to the translation before it is committed to the repository.

It is passed as a single parameter consisting of the filename of a current translation.

2.15 Память переводов

Добавлено в версии 2.20.

Weblate comes with a built-in translation memory consisting of the following:

- Manually imported translation memory (see *User interface*).
- Automatically stored translations performed in Weblate (depending on *Translation memory scopes*).
- Automatically imported past translations.

Content in the translation memory can be applied one of two ways:

- Manually, *Машинный перевод* view while translating.
- Automatically, by translating strings using *Автоматический перевод*, or *Автоматический перевод* addon.

For installation tips, see *Weblate Translation Memory*, which is turned on by default.

2.15.1 Translation memory scopes

Добавлено в версии 3.2: In earlier versions translation memory could be only loaded from a file corresponding to the current imported translation memory scope.

The translation memory scopes are there to allow both privacy and sharing of translations, to suit the desired behavior.

Imported translation memory

Importing arbitrary translation memory data using the `import_memory` command makes memory content available to all users and projects.

Per user translation memory

Stores all user translations automatically in the personal translation memory of each respective user.

Per project translation memory

All translations within a project are automatically stored in a project translation memory only available for this project.

Совместная память переводов

All translation within projects with shared translation memory turned on are stored in a shared translation memory available to all projects.

Please consider carefully whether to turn this feature on for shared Weblate installations, as it can have severe implications:

- The translations can be used by anybody else.
- This might lead to disclosing secret information.

2.15.2 Managing translation memory

User interface

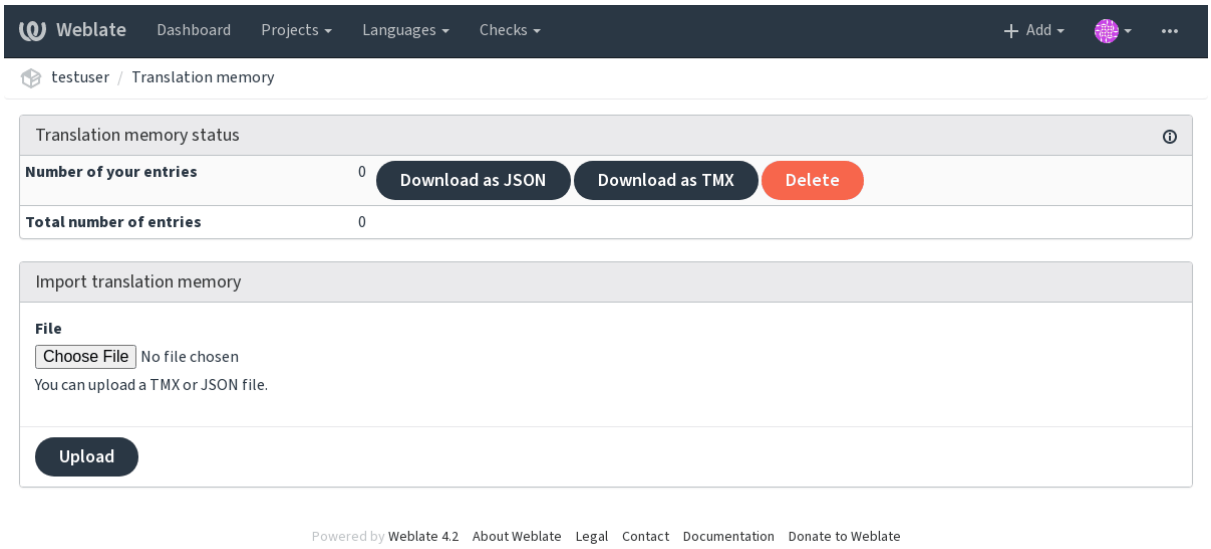
Добавлено в версии 3.2.

In the basic user interface you can manage per user and per project translation memories. It can be used to download, wipe or import translation memory.

Подсказка: Translation memory in JSON can be imported into Weblate, TMX is provided for interoperability with other tools.

См.также:

Weblate Translation Memory Schema



Интерфейс управления

Для управления содержимым памяти переводов существует несколько команд управления. Они работают с памятью переводов целиком, не применяя фильтр по областям (если только этот фильтр не запрошен параметрами):

`dump_memory` Exports the memory into JSON

`import_memory` Imports TMX or JSON files into the translation memory

2.16 Конфигурация

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

Примечание: After changing any of these settings, you need to restart Weblate - both WSGI and Celery processes.

In case it is run as mod_wsgi, you need to restart Apache to reload the configuration.

См.также:

Please also check Django's documentation for parameters configuring Django itself.

2.16.1 AKISMET_API_KEY

Weblate can use Akismet to check incoming anonymous suggestions for spam. Visit akismet.com to purchase an API key and associate it with a site.

2.16.2 ANONYMOUS_USER_NAME

Username of users that are not signed in.

См.также:

Управление доступом

2.16.3 AUDITLOG_EXPIRY

Добавлено в версии 3.6.

How many days Weblate should keep audit logs, which contain info about account activity.

По умолчанию установлен в 180 дней.

2.16.4 AUTH_LOCK_ATTEMPTS

Добавлено в версии 2.14.

Maximum number of failed authentication attempts before rate limiting is applied.

This is currently applied in the following locations:

- Logins. Deletes the account password, preventing the user from signing in without requesting a new password.
- Password resets. Prevents new e-mails from being sent, avoiding spamming users with too many password reset attempts.

По умолчанию установлен в 10.

См.также:

Ограничение частоты запросов,

2.16.5 AUTO_UPDATE

Добавлено в версии 3.2.

Изменено в версии 3.11: The original on/off option was changed to differentiate which strings are accepted.

Updates all repositories on a daily basis.

Подсказка: Useful if you are not using *Обработчики уведомлений* to update Weblate repositories automatically.

Примечание: On/off options exist in addition to string selection for backward compatibility.

Options are:

"none" No daily updates.

"remote" или False Only update remotes.

"full" или True Update remotes and merge working copy.

Примечание: This requires that *Фоновые задачи с использованием Celery* is working, and will take effect after it is restarted.

2.16.6 AVATAR_URL_PREFIX

Prefix for constructing avatar URLs as: `${AVATAR_URL_PREFIX}/avatar/${MAIL_HASH}?${PARAMS}`. The following services are known to work:

Gravatar (default), as per https://gravatar.com/ `AVATAR_URL_PREFIX = 'https://www.gravatar.com/'`

Libravatar, as per https://www.libravatar.org/ `AVATAR_URL_PREFIX = 'https://www.libravatar.org/'`

См.также:

Кеширование аватара, `ENABLE_AVATARS`, *Аватары*

2.16.7 AUTH_TOKEN_VALID

Добавлено в версии 2.14.

How long the authentication token and temporary password from password reset e-mails is valid for. Set in number of seconds, defaulting to 172800 (2 days).

2.16.8 AUTH_PASSWORD_DAYS

Добавлено в версии 2.15.

How many days using the same password should be allowed.

Примечание: Password changes made prior to Weblate 2.15 will not be accounted for in this policy.

По умолчанию установлен в 180 дней.

2.16.9 AUTOFIX_LIST

List of automatic fixes to apply when saving a string.

Примечание: Provide a fully-qualified path to the Python class that implementing the autofixer interface.

Available fixes:

weblate.trans.autofixes.whitespace.SameBookendingWhitespace Matches whitespace at the start and end of the string to the source.

weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis Replaces trailing dots (…) if the source string has ellipsis (…).

weblate.trans.autofixes.chars.RemoveZeroSpace Removes zero-width space characters if the source does not contain any.

weblate.trans.autofixes.chars.RemoveControlChars Removes control characters if the source does not contain any.

weblate.trans.autofixes.html.BleachHTML Removes unsafe HTML markup from strings flagged as `safe-html` (see *Небезопасный HTML-код*).

You can select which ones to use:

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

См.также:

Автоматические исправления, *Пользовательские автоматические исправления*

2.16.10 BASE_DIR

Base directory where Weblate sources are located. Used to derive several other paths by default:

- *DATA_DIR*

Default value: Top level directory of Weblate sources.

2.16.11 CSP_SCRIPT_SRC, CSP_IMG_SRC, CSP_CONNECT_SRC, CSP_STYLE_SRC, CSP_FONT_SRC

Customize `Content-Security-Policy` header for Weblate. The header is automatically generated based on enabled integrations with third-party services (Matomo, Google Analytics, Sentry, …).

All these default to empty list.

** Пример:: **

```
# Enable Cloudflare Javascript optimizations
CSP_SCRIPT_SRC = ["ajax.cloudflare.com"]
```

См.также:

Политика безопасности содержимого, Content Security Policy (CSP)

2.16.12 CHECK_LIST

List of quality checks to perform on a translation.

Примечание: Provide a fully-qualified path to the Python class implementing the check interface.

Adjust the list of checks to include ones relevant to you.

All built-in *Проверки качества* are turned on by default, from where you can change these settings. By default they are commented out in *Пример конфигурации* so that default values are used. New checks then carried out for each new Weblate version.

You can turn off all checks:

```
CHECK_LIST = ()
```

You can turn on only a few:

```
CHECK_LIST = (
    'weblate.checks.chars.BeginNewlineCheck',
    'weblate.checks.chars.EndNewlineCheck',
    'weblate.checks.chars.MaxLengthCheck',
)
```

Примечание: Changing this setting only affects newly changed translations, existing checks will still be stored in the database. To also apply changes to the stored translations, run `updatechecks`.

См.также:

Проверки качества, *Настройка поведения*

2.16.13 COMMENT_CLEANUP_DAYS

Добавлено в версии 3.6.

Delete comments after a given number of days. Defaults to `None`, meaning no deletion at all.

2.16.14 COMMIT_PENDING_HOURS

Добавлено в версии 2.10.

Number of hours between committing pending changes by way of the background task.

См.также:

Конфигурация компонента, *Возраст изменений для коммита*, *Выполнение задач технического обслуживания*, `commit_pending`

2.16.15 DATA_DIR

The folder Weblate stores all data in. It contains links to VCS repositories, a fulltext index and various configuration files for external tools.

The following subdirectories usually exist:

home Home directory used for invoking scripts.

ssh SSH keys and configuration.

static Default location for static Django files, specified by `STATIC_ROOT`.

media Default location for Django media files, specified by `MEDIA_ROOT`.

vcs Version control repositories.

backups Daily backup data, please check *Дампы данных для резервного копирования* for details.

Примечание: This directory has to be writable by Weblate. Running it as uWSGI means the `www-data` user should have write access to it.

The easiest way to achieve this is to make the user the owner of the directory:

```
sudo chown www-data:www-data -R $DATA_DIR
```

По умолчанию установлен в `$BASE_DIR/data`.

См.также:

BASE_DIR, *Резервное копирование и перенос Weblate*

2.16.16 DATABASE_BACKUP

Добавлено в версии 3.1.

Whether the database backups should be stored as plain text, compressed or skipped. The authorized values are:

- `"plain"`
- `"compressed"`
- `"none"`

См.также:

Резервное копирование и перенос Weblate

2.16.17 DEFAULT_ACCESS_CONTROL

Добавлено в версии 3.3.

The default access control setting for new projects:

0 *Публичный*

1 *Защищённый*

100 *Частный*

200 *Настраиваемый*

Use *Custom* if you are managing ACL manually, which means not relying on the internal Weblate management.

См.также:

По-проектный контроль доступа, *Управление доступом*, *Управление доступом*

2.16.18 DEFAULT_RESTRICTED_COMPONENT

Добавлено в версии 4.1.

The default value for component restriction.

См.также:

По-проектный контроль доступа, *Restricted access*, *Управление доступом*

2.16.19 DEFAULT_ADD_MESSAGE, DEFAULT_ADDON_MESSAGE, DEFAULT_COMMIT_MESSAGE, DEFAULT_DELETE_MESSAGE, DEFAULT_MERGE_MESSAGE

Default commit messages for different operations, please check *Конфигурация компонента* for details.

См.также:

Template markup, *Конфигурация компонента*, *Commit, add, delete, merge and addon messages*

2.16.20 DEFAULT_ADDONS

Default addons to install on every created component.

Примечание: This setting affects only newly created components.

Пример:

```
DEFAULT_ADDONS = {
     # Addon with no parameters
     "weblate.flags.target_edit": {},

     # Addon with parameters
     "weblate.autotranslate.autotranslate": {
         "mode": "suggest",
         "filter_type": "todo",
         "auto_source": "mt",
         "component": "",
         "engines": ["weblate-translation-memory"],
         "threshold": "80",
     }
}
```

См.также:

`install_addon`

2.16.21 DEFAULT_COMMITER_EMAIL

Добавлено в версии 2.4.

Committer e-mail address for created translation components defaulting to `noreply@weblate.org`.

См.также:

`DEFAULT_COMMITER_NAME`, *Конфигурация компонента*, *E-mail делающего коммит*

2.16.22 DEFAULT_COMMITER_NAME

Добавлено в версии 2.4.

Committer name for created translation components defaulting to `Weblate`.

См.также:

`DEFAULT_COMMITER_EMAIL`, *Конфигурация компонента*, *Имя произведшего коммит*

2.16.23 DEFAULT_MERGE_STYLE

Добавлено в версии 3.4.

Merge style for any new components.

- *rebase* - default
- *merge*

См.также:

Конфигурация компонента, *Стиль слияния*

2.16.24 DEFAULT_TRANSLATION_PROPAGATION

Добавлено в версии 2.5.

Значение по умолчанию для настройки распространения перевода, по умолчанию установлена в `True`.

См.также:

Конфигурация компонента, *Разрешить распространение перевода*

2.16.25 DEFAULT_PULL_MESSAGE

Title for new pull requests, defaulting to `'Update from Weblate'`.

2.16.26 ENABLE_AVATARS

Whether to turn on Gravatar-based avatars for users. By default this is on.

Avatars are fetched and cached on the server, lowering the risk of leaking private info, speeding up the user experience.

См.также:

Кеширование аватара, *`AVATAR_URL_PREFIX`*, *Аватары*

2.16.27 ENABLE_HOOKS

Whether to enable anonymous remote hooks.

См.также:

Обработчики уведомлений

2.16.28 ENABLE_HTTPS

Whether to send links to Weblate as HTTPS or HTTP. This setting affects sent e-mails and generated absolute URLs.

Подсказка: In the default configuration this is also used for several Django settings related to HTTPS.

См.также:

`SESSION_COOKIE_SECURE`, `CSRF_COOKIE_SECURE`, `SECURE_SSL_REDIRECT`, *Установка правильного домена сайта*

2.16.29 ENABLE_SHARING

Turn on/off the *Share* menu so users can share translation progress on social networks.

2.16.30 GITLAB_USERNAME

GitLab username used to send merge requests for translation updates.

См.также:

GitLab, *Установка Lab*

2.16.31 GITHUB_USERNAME

GitHub username used to send pull requests for translation updates.

См.также:

GitHub, *Установка hub*

2.16.32 GOOGLE_ANALYTICS_ID

Google Analytics ID to turn on monitoring of Weblate using Google Analytics.

2.16.33 HIDE_REPO_CREDENTIALS

Hide repository credentials from appearing in the web interface. In case you have repository URL with user and password, Weblate will hide it when related info is shown to users.

For example instead of `https://user:password@git.example.com/repo.git` it will show just `https://git.example.com/repo.git`. It tries to clean up VCS error messages too in a similar manner.

Примечание: This is turned on by default.

2.16.34 IP_BEHIND_REVERSE_PROXY

Добавлено в версии 2.14.

Indicates whether Weblate is running behind a reverse proxy.

If set to `True`, Weblate gets IP address from a header defined by *IP_PROXY_HEADER*.

Предупреждение: Ensure you are actually using a reverse proxy and that it sets this header, otherwise users will be able to fake the IP address.

Примечание: This is not on by default.

См.также:

Работа за обратным прокси, *Ограничение частоты запросов*, *IP_PROXY_HEADER*, *IP_PROXY_OFFSET*

2.16.35 IP_PROXY_HEADER

Добавлено в версии 2.14.

Indicates which header Weblate should obtain the IP address from when *IP_BEHIND_REVERSE_PROXY* is turned on.

По умолчанию установлен в `HTTP_X_FORWARDED_FOR`.

См.также:

Работа за обратным прокси, *Ограничение частоты запросов*, `SECURE_PROXY_SSL_HEADER`, *IP_BEHIND_REVERSE_PROXY*, *IP_PROXY_OFFSET*

2.16.36 IP_PROXY_OFFSET

Добавлено в версии 2.14.

Indicates which part of *IP_PROXY_HEADER* is used as client IP address.

Depending on your setup, this header might consist of several IP addresses, (for example `X-Forwarded-For: a, b, client-ip`) and you can configure which address from the header is used as client IP address here.

Предупреждение: Setting this affects the security of your installation, you should only configure it to use trusted proxies for determining IP address.

По умолчанию установлен в 0.

См.также:

Работа за обратным прокси, *Ограничение частоты запросов*, `SECURE_PROXY_SSL_HEADER`, *IP_BEHIND_REVERSE_PROXY*, *IP_PROXY_HEADER*

2.16.37 LEGAL_URL

Добавлено в версии 3.5.

URL where your Weblate instance shows its legal documents.

Подсказка: Useful if you host your legal documents outside Weblate for embedding them inside Weblate, please check *Право* for details.

Пример:

```
LEGAL_URL = "https://weblate.org/terms/"
```

2.16.38 LICENSE_EXTRA

Additional licenses to include in the license choices.

Примечание: Each license definition should be tuple of its short name, a long name and an URL.

Например:

```
LICENSE_EXTRA = [
    (
        "AGPL-3.0",
        "GNU Affero General Public License v3.0",
        "https://www.gnu.org/licenses/agpl-3.0-standalone.html",
    ),
]
```

2.16.39 LICENSE_FILTER

Optional addition of licenses to show.

Примечание: This filter uses the short license names.

Например:

```
LICENSE_FILTER = {"AGPL-3.0", "GPL-3.0-or-later"}
```

2.16.40 LICENSE_REQUIRED

Defines whether the license attribute in *Конфигурация компонента* is required.

Примечание: This is off by default.

2.16.41 LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH

Whether the length of a given translation should be limited. The restriction is the length of the source string * 10 characters.

Подсказка: Set this to `False` to allow longer translations (up to 10.000 characters) irrespective of source string length.

Примечание: По умолчанию установлен в `True`.

2.16.42 LOCALIZE_CDN_URL and LOCALIZE_CDN_PATH

These settings configure the *CDN локализации JavaScript'а* addon. `LOCALIZE_CDN_URL` defines root URL where the localization CDN is available and `LOCALIZE_CDN_PATH` defines path where Weblate should store generated files which will be served at the `LOCALIZE_CDN_URL`.

Подсказка: On Hosted Weblate, this uses `https://weblate-cdn.com/`.

См.также:

CDN локализации JavaScript'а

2.16.43 LOGIN_REQUIRED_URLS

A list of URLs you want to require logging into. (Besides the standard rules built into Weblate).

Подсказка: This allows you to password protect a whole installation using:

```
LOGIN_REQUIRED_URLS = (
    r'/(.*)$',
)
REST_FRAMEWORK["DEFAULT_PERMISSION_CLASSES"] = [
    "rest_framework.permissions.IsAuthenticated"
]
```

Подсказка: It is desirable to lock down API access as well, as shown in the above example.

2.16.44 LOGIN_REQUIRED_URLS_EXCEPTIONS

List of exceptions for `LOGIN_REQUIRED_URLS`. If not specified, users are allowed to access the login page.

Some of exceptions you might want to include:

```
LOGIN_REQUIRED_URLS_EXCEPTIONS = (
    r'/accounts/(.*)$', # Required for login
    r'/static/(.*)$',   # Required for development mode
    r'/widgets/(.*)$',  # Allowing public access to widgets
    r'/data/(.*)$',     # Allowing public access to data exports
    r'/hooks/(.*)$',    # Allowing public access to notification hooks
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    r'/api/(.*)$',       # Allowing access to API
    r'/js/i18n/$',       # JavaScript localization
)
```

2.16.45 MATOMO_SITE_ID

ID of a site in Matomo (formerly Piwik) you want to track.

Примечание: This integration does not support the Matomo Tag Manager.

См.также:

MATOMO_URL

2.16.46 MATOMO_URL

Full URL (including trailing slash) of a Matomo (formerly Piwik) installation you want to use to track Weblate use. Please check <https://matomo.org/> for more details.

Подсказка: This integration does not support the Matomo Tag Manager.

Например:

```
MATOMO_SITE_ID = 1
MATOMO_URL = "https://example.matomo.cloud/"
```

См.также:

MATOMO_SITE_ID

2.16.47 MT_SERVICES

Изменено в версии 3.0: The setting was renamed from `MACHINE_TRANSLATION_SERVICES` to `MT_SERVICES` to be consistent with other machine translation settings.

List of enabled machine translation services to use.

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

```
MT_SERVICES = (
    'weblate.machinery.apertium.ApertiumAPYTranslation',
    'weblate.machinery.deepl.DeepLTranslation',
    'weblate.machinery.glosbe.GlosbeTranslation',
    'weblate.machinery.google.GoogleTranslation',
    'weblate.machinery.microsoft.MicrosoftCognitiveTranslation',
    'weblate.machinery.microsoftterminology.MicrosoftTerminologyService',
    'weblate.machinery.mymemory.MyMemoryTranslation',
    'weblate.machinery.tmserver.AmagamaTranslation',
    'weblate.machinery.tmserver.TMServerTranslation',
    'weblate.machinery.yandex.YandexTranslation',
    'weblate.machinery.weblatetm.WeblateTranslation',
    'weblate.machinery.saptranslationhub.SAPTranslationHub',
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    'weblate.memory.machine.WeblateMemory',
)
```

См.также:

Машинный перевод, *Машинный перевод*

2.16.48 MT_APERTIUM_APY

URL of the Apertium-APy server, https://wiki.apertium.org/wiki/Apertium-apy

См.также:

Apertium, *Машинный перевод*, *Машинный перевод*

2.16.49 MT_AWS_ACCESS_KEY_ID

Access key ID for Amazon Translate.

См.также:

AWS, *Машинный перевод*, *Машинный перевод*

2.16.50 MT_AWS_SECRET_ACCESS_KEY

API secret key for Amazon Translate.

См.также:

AWS, *Машинный перевод*, *Машинный перевод*

2.16.51 MT_AWS_REGION

Region name to use for Amazon Translate.

См.также:

AWS, *Машинный перевод*, *Машинный перевод*

2.16.52 MT_BAIDU_ID

Client ID for the Baidu Zhiyun API, you can register at https://api.fanyi.baidu.com/api/trans/product/index

См.также:

Машинный перевод через API Baidu, *Машинный перевод*, *Машинный перевод*

2.16.53 MT_BAIDU_SECRET

Client secret for the Baidu Zhiyun API, you can register at https://api.fanyi.baidu.com/api/trans/product/index

См.также:

Машинный перевод через API Baidu, *Машинный перевод*, *Машинный перевод*

2.16.54 MT_DEEPL_API_VERSION

Добавлено в версии 4.1.1.

API version to use with DeepL service. The version limits scope of usage:

v1 Is meant for CAT tools and is usable with user-based subscription.

v2 Is meant for API usage and the subscription is usage based.

Previously Weblate was classified as a CAT tool by DeepL, so it was supposed to use the v1 API, but now is supposed to use the v2 API. Therefore it defaults to v2, and you can change it to v1 in case you have an existing CAT subscription and want Weblate to use that.

См.также:

DeepL, *Машинный перевод*, *Машинный перевод*

2.16.55 MT_DEEPL_KEY

API key for the DeepL API, you can register at https://www.deepl.com/pro.html

См.также:

DeepL, *Машинный перевод*, *Машинный перевод*

2.16.56 MT_GOOGLE_KEY

API key for Google Translate API v2, you can register at https://cloud.google.com/translate/docs

См.также:

Google Translate, *Машинный перевод*, *Машинный перевод*

2.16.57 MT_GOOGLE_CREDENTIALS

API v3 JSON credentials file obtained in the Google cloud console. Please provide a full OS path. Credentials are per service-account affiliated with certain project. Please check https://cloud.google.com/docs/authentication/getting-started for more details.

2.16.58 MT_GOOGLE_PROJECT

API v3 Google cloud *project id* with activated translation service and billing activated. Please check https://cloud.google.com/appengine/docs/standard/nodejs/building-app/creating-project for more details

2.16.59 MT_GOOGLE_LOCATION

API v3 Google Cloud Application Engine may be specific to a location. Change accordingly if the default `global` fallback does not work for you.

Please check https://cloud.google.com/appengine/docs/locations for more details

См.также:

Google Translate API V3 (расширенный)

2.16.60 MT_MICROSOFT_BASE_URL

Region base URL domain as defined in the «Base URLs» section.

Defaults to `api.cognitive.microsofttranslator.com` for Azure Global.

For Azure China, please use `api.translator.azure.cn`.

2.16.61 MT_MICROSOFT_COGNITIVE_KEY

Client key for the Microsoft Cognitive Services Translator API.

См.также:

Microsoft Cognitive Services Translator, *Машинный перевод*, *Машинный перевод*, Cognitive Services - Text Translation API, Microsoft Azure Portal

2.16.62 MT_MICROSOFT_REGION

Region prefix as defined in «Multi service subscription».

2.16.63 MT_MICROSOFT_ENDPOINT_URL

Region endpoint URL domain for access token as defined in the «Authenticating with an access token» section.

Defaults to `api.cognitive.microsoft.com` for Azure Global.

For Azure China, please use your endpoint from the Azure Portal.

2.16.64 MT_MODERNMT_KEY

Ключ API для движка машинного перевода ModernMT.

См.также:

ModernMT `MT_MODERNMT_URL`

2.16.65 MT_MODERNMT_URL

URL of ModernMT. It defaults to `https://api.modernmt.com/` for the cloud service.

См.также:

ModernMT `MT_MODERNMT_KEY`

2.16.66 MT_MYMEMORY_EMAIL

MyMemory identification e-mail address. It permits 1000 requests per day.

См.также:

MyMemory, *Машинный перевод*, *Машинный перевод*, MyMemory: API technical specifications

2.16.67 MT_MYMEMORY_KEY

MyMemory access key for private translation memory, use it with `MT_MYMEMORY_USER`.

См.также:

MyMemory, *Машинный перевод*, *Машинный перевод*, MyMemory: API key generator

2.16.68 MT_MYMEMORY_USER

MyMemory user ID for private translation memory, use it with `MT_MYMEMORY_KEY`.

См.также:

MyMemory, *Машинный перевод*, *Машинный перевод*, MyMemory: API key generator

2.16.69 MT_NETEASE_KEY

App key for NetEase Sight API, you can register at https://sight.netease.com/

См.также:

NetEase Sight API machine translation, *Машинный перевод*, *Машинный перевод*

2.16.70 MT_NETEASE_SECRET

App secret for the NetEase Sight API, you can register at https://sight.netease.com/

См.также:

NetEase Sight API machine translation, *Машинный перевод*, *Машинный перевод*

2.16.71 MT_TMSERVER

URL where tmserver is running.

См.также:

tmserver, *Машинный перевод*, *Машинный перевод*, tmserver

2.16.72 MT_YANDEX_KEY

API key for the Yandex Translate API, you can register at https://tech.yandex.com/translate/

См.также:

Yandex Translate, *Машинный перевод*, *Машинный перевод*

2.16.73 MT_YOUDAO_ID

Client ID for the Youdao Zhiyun API, you can register at https://ai.youdao.com/product-fanyi-text.s.

См.также:

Youdao Zhiyun API machine translation, *Машинный перевод*, *Машинный перевод*

2.16.74 MT_YOUDAO_SECRET

Client secret for the Youdao Zhiyun API, you can register at https://ai.youdao.com/product-fanyi-text.s.

См.также:

Youdao Zhiyun API machine translation, *Машинный перевод*, *Машинный перевод*

2.16.75 MT_SAP_BASE_URL

API URL to the SAP Translation Hub service.

См.также:

SAP Translation Hub, *Машинный перевод*, *Машинный перевод*

2.16.76 MT_SAP_SANDBOX_APIKEY

API key for sandbox API usage

См.также:

SAP Translation Hub, *Машинный перевод*, *Машинный перевод*

2.16.77 MT_SAP_USERNAME

Your SAP username

См.также:

SAP Translation Hub, *Машинный перевод*, *Машинный перевод*

2.16.78 MT_SAP_PASSWORD

Your SAP password

См.также:

SAP Translation Hub, *Машинный перевод*, *Машинный перевод*

2.16.79 MT_SAP_USE_MT

Whether to also use machine translation services, in addition to the term database. Possible values: `True` or `False`

См.также:

SAP Translation Hub, *Машинный перевод*, *Машинный перевод*

2.16.80 NEARBY_MESSAGES

How many strings to show around the currently translated string. This is just a default value, users can adjust this in *Профиль пользователя*.

2.16.81 RATELIMIT_ATTEMPTS

Добавлено в версии 3.2.

Maximum number of authentication attempts before rate limiting is applied.

По умолчанию установлен в 5.

См.также:

Ограничение частоты запросов, `RATELIMIT_WINDOW`, `RATELIMIT_LOCKOUT`

2.16.82 RATELIMIT_WINDOW

Добавлено в версии 3.2.

How long authentication is accepted after rate limiting applies.

An amount of seconds defaulting to 300 (5 minutes).

См.также:

Ограничение частоты запросов, `RATELIMIT_ATTEMPTS`, `RATELIMIT_LOCKOUT`

2.16.83 RATELIMIT_LOCKOUT

Добавлено в версии 3.2.

How long authentication is locked after rate limiting applies.

An amount of seconds defaulting to 600 (10 minutes).

См.также:

Ограничение частоты запросов, `RATELIMIT_ATTEMPTS`, `RATELIMIT_WINDOW`

2.16.84 REGISTRATION_ALLOW_BACKENDS

Добавлено в версии 4.1.

List of authentication backends to allow registration from in case it is otherwise disabled by `REGISTRATION_OPEN`.

Пример:

```
REGISTRATION_ALLOW_BACKENDS = ["azuread-oauth2", "azuread-tenant-oauth2"]
```

Подсказка: The backend names match names used in URL for authentication.

См.также:

REGISTRATION_OPEN

2.16.85 REGISTRATION_CAPTCHA

A value of either `True` or `False` indicating whether registration of new accounts is protected by CAPTCHA. This setting is optional, and a default of `True` will be assumed if it is not supplied.

If turned on, a CAPTCHA is added to all pages where a users enters their e-mail address:

- New account registration.
- Password recovery.
- Adding e-mail to an account.
- Contact form for users that are not signed in.

2.16.86 REGISTRATION_EMAIL_MATCH

Добавлено в версии 2.17.

Allows you to filter which e-mail addresses can register.

Defaults to `.*`, which allows any e-mail address to be registered.

You can use it to restrict registration to a single e-mail domain:

```
REGISTRATION_EMAIL_MATCH = r'^.*@weblate\.org$'
```

2.16.87 REGISTRATION_OPEN

Whether registration of new accounts is currently permitted. This optional setting can remain the default `True`, or changed to `False`.

This setting affects built-in authentication by e-mail address or through the Python Social Auth (you can whitelist certain back-ends using *REGISTRATION_ALLOW_BACKENDS*).

Примечание: If using third-party authentication methods such as *Аутентификация через LDAP*, it just hides the registration form, but new users might still be able to sign in and create accounts.

См.также:

REGISTRATION_ALLOW_BACKENDS, *REGISTRATION_EMAIL_MATCH*

2.16.88 REPOSITORY_ALERT_THRESHOLD

Добавлено в версии 4.0.2.

Threshold for triggering an alert for outdated repositories, or ones that contain too many changes. Defaults to 25.

См.также:

Предупреждения компонента перевода

2.16.89 SENTRY_DSN

Добавлено в версии 3.9.

Sentry DSN to use for *Сбор отчётов об ошибках*.

См.также:

Django integration for Sentry

2.16.90 SIMPLIFY_LANGUAGES

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

Turn this off if you want to different translations for each variant.

2.16.91 SITE_DOMAIN

Configures site domain. This is necessary to produce correct absolute links in many scopes (for example activation e-mails, notifications or RSS feeds).

In case Weblate is running on non-standard port, include it here as well.

Примеры:

```
# Production site with domain name
SITE_DOMAIN = "weblate.example.com"

# Local development with IP address and port
SITE_DOMAIN = "127.0.0.1:8000"
```

Примечание: This setting should only contain the domain name. For configuring protocol, (enabling and enforcing HTTPS) use `ENABLE_HTTPS` and for changing URL, use `URL_PREFIX`.

Подсказка: On a Docker container, the site domain is configured through `WEBLATE_ALLOWED_HOSTS`.

См.также:

Установка правильного домена сайта, *Настройка разрешённых хостов*, *Правильная настройка HTTPS* `WEBLATE_SITE_DOMAIN`, `ENABLE_HTTPS`

2.16.92 SITE_TITLE

Site title to be used for the website and sent e-mails.

2.16.93 SPECIAL_CHARS

Additional characters to include in the visual keyboard, *Визуальная клавиатура*.

The default value is:

```
SPECIAL_CHARS = ('\t', '\n', '…')
```

2.16.94 SINGLE_PROJECT

Добавлено в версии 3.8.

Redirects users directly to a project or component instead of showing the dashboard. You can either set it to `True` and in this case it only works in case there is actually only single project in Weblate. Alternatively set the project slug, and it will redirect unconditionally to this project.

Изменено в версии 3.11: The setting now also accepts a project slug, to force displaying that single project.

Пример:

```
SINGLE_PROJECT = "test"
```

2.16.95 STATUS_URL

The URL where your Weblate instance reports its status.

2.16.96 SUGGESTION_CLEANUP_DAYS

Добавлено в версии 3.2.1.

Automatically deletes suggestions after a given number of days. Defaults to `None`, meaning no deletions.

2.16.97 URL_PREFIX

This setting allows you to run Weblate under some path (otherwise it relies on being run from the webserver root).

Примечание: 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`.

Подсказка: The prefix should start with a `/`.

Пример:

```
URL_PREFIX = '/translations'
```

Примечание: This setting does not work with Django's built-in server, you would have to adjust `urls.py` to contain this prefix.

2.16.98 VCS_BACKENDS

Configuration of available VCS backends.

Примечание: Weblate tries to use all supported back-ends you have the tools for.

Подсказка: You can limit choices or add custom VCS back-ends by using this.

```
VCS_BACKENDS = (
    'weblate.vcs.git.GitRepository',
)
```

См.также:

Интеграция с системой контроля версий

2.16.99 VCS_CLONE_DEPTH

Добавлено в версии 3.10.2.

Configures how deep cloning of repositories Weblate should do.

Примечание: Currently this is only supported in *Git*. By default Weblate does shallow clones of the repositories to make cloning faster and save disk space. Depending on your usage (for example when using custom *Надстройки*), you might want to increase the depth or turn off shallow clones completely by setting this to 0.

Подсказка: In case you get `fatal: protocol error: expected old/new/ref, got 'shallow <commit hash>'` error when pushing from Weblate, turn off shallow clones completely by setting:

```
VCS_CLONE_DEPTH = 0
```

2.16.100 WEBLATE_ADDONS

List of addons available for use. To use them, they have to be enabled for a given translation component. By default this includes all built-in addons, when extending the list you will probably want to keep existing ones enabled, for example:

```
WEBLATE_ADDONS = (
    # Built-in addons
    "weblate.addons.gettext.GenerateMoAddon",
    "weblate.addons.gettext.UpdateLinguasAddon",
    "weblate.addons.gettext.UpdateConfigureAddon",
    "weblate.addons.gettext.MsgmergeAddon",
    "weblate.addons.gettext.GettextCustomizeAddon",
    "weblate.addons.gettext.GettextAuthorComments",
    "weblate.addons.cleanup.CleanupAddon",
    "weblate.addons.consistency.LangaugeConsistencyAddon",
    "weblate.addons.discovery.DiscoveryAddon",
    "weblate.addons.flags.SourceEditAddon",
    "weblate.addons.flags.TargetEditAddon",
    "weblate.addons.flags.SameEditAddon",
    "weblate.addons.flags.BulkEditAddon",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "weblate.addons.generate.GenerateFileAddon",
    "weblate.addons.json.JSONCustomizeAddon",
    "weblate.addons.properties.PropertiesSortAddon",
    "weblate.addons.git.GitSquashAddon",
    "weblate.addons.removal.RemoveComments",
    "weblate.addons.removal.RemoveSuggestions",
    "weblate.addons.resx.ResxUpdateAddon",
    "weblate.addons.autotranslate.AutoTranslateAddon",
    "weblate.addons.yaml.YAMLCustomizeAddon",
    "weblate.addons.cdn.CDNJSAddon",

    # Addon you want to include
    "weblate.addons.example.ExampleAddon",
)
```

См.также:

Надстройки

2.16.101 WEBLATE_EXPORTERS

Добавлено в версии 4.2.

List of a available exporters offering downloading translations or glossaries in various file formats.

См.также:

Поддерживаемые форматы файлов

2.16.102 WEBLATE_FORMATS

Добавлено в версии 3.0.

List of file formats available for use.

Примечание: The default list already has the common formats.

См.также:

Поддерживаемые форматы файлов

2.16.103 WEBLATE_GPG_IDENTITY

Добавлено в версии 3.1.

Identity used by Weblate to sign Git commits, for example:

```
WEBLATE_GPG_IDENTITY = 'Weblate <weblate@example.com>'
```

The Weblate GPG keyring is searched for a matching key (`home/.gnupg` under *DATA_DIR*). If not found, a key is generated, please check *Signing Git commits with GnuPG* for more details.

См.также:

Signing Git commits with GnuPG

2.17 Пример конфигурации

The following example is shipped as `weblate/settings_example.py` with Weblate:

```
#
# Copyright © 2012 - 2020 Michal Čihař <michal@cihar.com>
#
# This file is part of Weblate <https://weblate.org/>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.
#


import os
import platform
from logging.handlers import SysLogHandler

#
# Django settings for Weblate project.
#

DEBUG = True

ADMINS = (
    # ("Your Name", "your_email@example.com"),
)

MANAGERS = ADMINS

DATABASES = {
    "default": {
        # Use "postgresql" or "mysql".
        "ENGINE": "django.db.backends.postgresql",
        # Database name.
        "NAME": "weblate",
        # Database user.
        "USER": "weblate",
        # Database password.
        "PASSWORD": "",
        # Set to empty string for localhost.
        "HOST": "127.0.0.1",
        # Set to empty string for default.
        "PORT": "",
        # Customizations for databases.
        "OPTIONS": {
            # In case of using an older MySQL server,
            # which has MyISAM as a default storage
            # "init_command": "SET storage_engine=INNODB",
            # Uncomment for MySQL older than 5.7:
            # "init_command": "SET sql_mode='STRICT_TRANS_TABLES'",
            # Set emoji capable charset for MySQL:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            # "charset": "utf8mb4",
            # Change connection timeout in case you get MySQL gone away error:
            # "connect_timeout": 28800,
        },
    }
}

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Data directory
DATA_DIR = os.path.join(BASE_DIR, "data")

# Local time zone for this installation. Choices can be found here:
# http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
# although not all choices may be available on all operating systems.
# In a Windows environment this must be set to your system time zone.
TIME_ZONE = "UTC"

# Language code for this installation. All choices can be found here:
# http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = "en-us"

LANGUAGES = (
    ("ar", "ةيبرعلا"),
    ("az", "Azərbaycan"),
    ("be", "Беларуская"),
    ("be@latin", "Biełaruskaja"),
    ("bg", "Български"),
    ("br", "Brezhoneg"),
    ("ca", "Català"),
    ("cs", "Čeština"),
    ("da", "Dansk"),
    ("de", "Deutsch"),
    ("en", "English"),
    ("el", "Ελληνικά"),
    ("en-gb", "English (United Kingdom)"),
    ("es", "Español"),
    ("fi", "Suomi"),
    ("fr", "Français"),
    ("gl", "Galego"),
    ("he", "עברית"),
    ("hu", "Magyar"),
    ("hr", "Hrvatski"),
    ("id", "Indonesia"),
    ("is", "Íslenska"),
    ("it", "Italiano"),
    ("ja", "□□□"),
    ("kab", "Taqbaylit"),
    ("kk", "Қазақ тілі"),
    ("ko", "□□□"),
    ("nb", "Norsk bokmål"),
    ("nl", "Nederlands"),
    ("pl", "Polski"),
    ("pt", "Português"),
    ("pt-br", "Português brasileiro"),
    ("ru", "Русский"),
    ("sk", "Slovenčina"),
    ("sl", "Slovenščina"),
    ("sq", "Shqip"),
    ("sr", "Српски"),
    ("sv", "Svenska"),
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    ("tr", "Türkçe"),
    ("uk", "Українська"),
    ("zh-hans", "□□□"),
    ("zh-hant", "□□□"),
)

SITE_ID = 1

# If you set this to False, Django will make some optimizations so as not
# to load the internationalization machinery.
USE_I18N = True

# If you set this to False, Django will not format dates, numbers and
# calendars according to the current locale.
USE_L10N = True

# If you set this to False, Django will not use timezone-aware datetimes.
USE_TZ = True

# URL prefix to use, please see documentation for more details
URL_PREFIX = ""

# Absolute filesystem path to the directory that will hold user-uploaded files.
MEDIA_ROOT = os.path.join(DATA_DIR, "media")

# URL that handles the media served from MEDIA_ROOT. Make sure to use a
# trailing slash.
MEDIA_URL = f"{URL_PREFIX}/media/"

# Absolute path to the directory static files should be collected to.
# Don't put anything in this directory yourself; store your static files
# in apps' "static/" subdirectories and in STATICFILES_DIRS.
STATIC_ROOT = os.path.join(DATA_DIR, "static")

# URL prefix for static files.
STATIC_URL = f"{URL_PREFIX}/static/"

# Additional locations of static files
STATICFILES_DIRS = (
    # Put strings here, like "/home/html/static" or "C:/www/django/static".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not relative paths.
)

# List of finder classes that know how to find static files in
# various locations.
STATICFILES_FINDERS = (
    "django.contrib.staticfiles.finders.FileSystemFinder",
    "django.contrib.staticfiles.finders.AppDirectoriesFinder",
    "compressor.finders.CompressorFinder",
)

# Make this unique, and don't share it with anybody.
# You can generate it using weblate/examples/generate-secret-key
SECRET_KEY = ""

_TEMPLATE_LOADERS = [
    "django.template.loaders.filesystem.Loader",
    "django.template.loaders.app_directories.Loader",
]
if not DEBUG:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    _TEMPLATE_LOADERS = [("django.template.loaders.cached.Loader", _TEMPLATE_
↪LOADERS)]
TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "OPTIONS": {
            "context_processors": [
                "django.contrib.auth.context_processors.auth",
                "django.template.context_processors.debug",
                "django.template.context_processors.i18n",
                "django.template.context_processors.request",
                "django.template.context_processors.csrf",
                "django.contrib.messages.context_processors.messages",
                "weblate.trans.context_processors.weblate_context",
            ],
            "loaders": _TEMPLATE_LOADERS,
        },
    }
]


# GitHub username for sending pull requests.
# Please see the documentation for more details.
GITHUB_USERNAME = None

# GitLab username for sending merge requests.
# Please see the documentation for more details.
GITLAB_USERNAME = None

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.email.EmailAuth",
    # "social_core.backends.google.GoogleOAuth2",
    # "social_core.backends.github.GithubOAuth2",
    # "social_core.backends.bitbucket.BitbucketOAuth",
    # "social_core.backends.suse.OpenSUSEOpenId",
    # "social_core.backends.ubuntu.UbuntuOpenId",
    # "social_core.backends.fedora.FedoraOpenId",
    # "social_core.backends.facebook.FacebookOAuth2",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Custom user model
AUTH_USER_MODEL = "weblate_auth.User"

# Social auth backends setup
SOCIAL_AUTH_GITHUB_KEY = ""
SOCIAL_AUTH_GITHUB_SECRET = ""
SOCIAL_AUTH_GITHUB_SCOPE = ["user:email"]

SOCIAL_AUTH_BITBUCKET_KEY = ""
SOCIAL_AUTH_BITBUCKET_SECRET = ""
SOCIAL_AUTH_BITBUCKET_VERIFIED_EMAILS_ONLY = True

SOCIAL_AUTH_FACEBOOK_KEY = ""
SOCIAL_AUTH_FACEBOOK_SECRET = ""
SOCIAL_AUTH_FACEBOOK_SCOPE = ["email", "public_profile"]
SOCIAL_AUTH_FACEBOOK_PROFILE_EXTRA_PARAMS = {"fields": "id,name,email"}
SOCIAL_AUTH_FACEBOOK_API_VERSION = "3.1"

SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = ""
```

(continues on next page)

(продолжение с предыдущей страницы)

```
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = ""

# Social auth settings
SOCIAL_AUTH_PIPELINE = (
    "social_core.pipeline.social_auth.social_details",
    "social_core.pipeline.social_auth.social_uid",
    "social_core.pipeline.social_auth.auth_allowed",
    "social_core.pipeline.social_auth.social_user",
    "weblate.accounts.pipeline.store_params",
    "weblate.accounts.pipeline.verify_open",
    "social_core.pipeline.user.get_username",
    "weblate.accounts.pipeline.require_email",
    "social_core.pipeline.mail.mail_validation",
    "weblate.accounts.pipeline.revoke_mail_code",
    "weblate.accounts.pipeline.ensure_valid",
    "weblate.accounts.pipeline.remove_account",
    "social_core.pipeline.social_auth.associate_by_email",
    "weblate.accounts.pipeline.reauthenticate",
    "weblate.accounts.pipeline.verify_username",
    "social_core.pipeline.user.create_user",
    "social_core.pipeline.social_auth.associate_user",
    "social_core.pipeline.social_auth.load_extra_data",
    "weblate.accounts.pipeline.cleanup_next",
    "weblate.accounts.pipeline.user_full_name",
    "weblate.accounts.pipeline.store_email",
    "weblate.accounts.pipeline.notify_connect",
    "weblate.accounts.pipeline.password_reset",
)
SOCIAL_AUTH_DISCONNECT_PIPELINE = (
    "social_core.pipeline.disconnect.allowed_to_disconnect",
    "social_core.pipeline.disconnect.get_entries",
    "social_core.pipeline.disconnect.revoke_tokens",
    "weblate.accounts.pipeline.cycle_session",
    "weblate.accounts.pipeline.adjust_primary_mail",
    "weblate.accounts.pipeline.notify_disconnect",
    "social_core.pipeline.disconnect.disconnect",
    "weblate.accounts.pipeline.cleanup_next",
)

# Custom authentication strategy
SOCIAL_AUTH_STRATEGY = "weblate.accounts.strategy.WeblateStrategy"

# Raise exceptions so that we can handle them later
SOCIAL_AUTH_RAISE_EXCEPTIONS = True

SOCIAL_AUTH_EMAIL_VALIDATION_FUNCTION = "weblate.accounts.pipeline.send_validation"
SOCIAL_AUTH_EMAIL_VALIDATION_URL = "{0}/accounts/email-sent/".format(URL_PREFIX)
SOCIAL_AUTH_LOGIN_ERROR_URL = "{0}/accounts/login/".format(URL_PREFIX)
SOCIAL_AUTH_EMAIL_FORM_URL = "{0}/accounts/email/".format(URL_PREFIX)
SOCIAL_AUTH_NEW_ASSOCIATION_REDIRECT_URL = "{0}/accounts/profile/#account".format(
    URL_PREFIX
)
SOCIAL_AUTH_PROTECTED_USER_FIELDS = ("email",)
SOCIAL_AUTH_SLUGIFY_USERNAMES = True
SOCIAL_AUTH_SLUGIFY_FUNCTION = "weblate.accounts.pipeline.slugify_username"

# Password validation configuration
AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME": "django.contrib.auth.password_validation.
↪UserAttributeSimilarityValidator"  # noqa: E501, pylint: disable=line-too-long
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    },
    {
        "NAME": "django.contrib.auth.password_validation.MinimumLengthValidator",
        "OPTIONS": {"min_length": 10},
    },
    {"NAME": "django.contrib.auth.password_validation.CommonPasswordValidator"},
    {"NAME": "django.contrib.auth.password_validation.NumericPasswordValidator"},
    {"NAME": "weblate.accounts.password_validation.CharsPasswordValidator"},
    {"NAME": "weblate.accounts.password_validation.PastPasswordsValidator"},
    # Optional password strength validation by django-zxcvbn-password
    # {
    #     "NAME": "zxcvbn_password.ZXCVBNValidator",
    #     "OPTIONS": {
    #         "min_score": 3,
    #         "user_attributes": ("username", "email", "full_name")
    #     }
    # },
]

# Allow new user registrations
REGISTRATION_OPEN = True

# Shortcut for login required setting
REQUIRE_LOGIN = False

# Middleware
MIDDLEWARE = [
    "weblate.middleware.RedirectMiddleware",
    "weblate.middleware.ProxyMiddleware",
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "weblate.accounts.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
    "social_django.middleware.SocialAuthExceptionMiddleware",
    "weblate.accounts.middleware.RequireLoginMiddleware",
    "weblate.api.middleware.ThrottlingMiddleware",
    "weblate.middleware.SecurityMiddleware",
]

ROOT_URLCONF = "weblate.urls"

# Django and Weblate apps
INSTALLED_APPS = [
    # Weblate apps on top to override Django locales and templates
    "weblate.addons",
    "weblate.auth",
    "weblate.checks",
    "weblate.formats",
    "weblate.glossary",
    "weblate.machinery",
    "weblate.trans",
    "weblate.lang",
    "weblate.langdata",
    "weblate.memory",
    "weblate.screenshots",
    "weblate.fonts",
    "weblate.accounts",
    "weblate.utils",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "weblate.vcs",
    "weblate.wladmin",
    "weblate",
    # Optional: Git exporter
    "weblate.gitexport",
    # Standard Django modules
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "django.contrib.admin.apps.SimpleAdminConfig",
    "django.contrib.admindocs",
    "django.contrib.sitemaps",
    "django.contrib.humanize",
    # Third party Django modules
    "social_django",
    "crispy_forms",
    "compressor",
    "rest_framework",
    "rest_framework.authtoken",
    "django_filters",
]

# Custom exception reporter to include some details
DEFAULT_EXCEPTION_REPORTER_FILTER = "weblate.trans.debug.
↪WeblateExceptionReporterFilter"

# Default logging of Weblate messages
# - to syslog in production (if available)
# - otherwise to console
# - you can also choose "logfile" to log into separate file
#   after configuring it below

# Detect if we can connect to syslog
HAVE_SYSLOG = False
if platform.system() != "Windows":
    try:
        handler = SysLogHandler(address="/dev/log", facility=SysLogHandler.LOG_
↪LOCAL2)
        handler.close()
        HAVE_SYSLOG = True
    except IOError:
        HAVE_SYSLOG = False

if DEBUG or not HAVE_SYSLOG:
    DEFAULT_LOG = "console"
else:
    DEFAULT_LOG = "syslog"
DEFAULT_LOGLEVEL = "DEBUG" if DEBUG else "INFO"

# A sample logging configuration. The only tangible logging
# performed by this configuration is to send an email to
# the site admins on every HTTP 500 error when DEBUG=False.
# See http://docs.djangoproject.com/en/stable/topics/logging for
# more details on how to customize your logging configuration.
LOGGING = {
    "version": 1,
    "disable_existing_loggers": True,
    "filters": {"require_debug_false": {"()": "django.utils.log.RequireDebugFalse"}
↪},
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "formatters": {
        "syslog": {"format": "weblate[%(process)d]: %(levelname)s %(message)s"},
        "simple": {"format": "%(levelname)s %(message)s"},
        "logfile": {"format": "%(asctime)s %(levelname)s %(message)s"},
        "django.server": {
            "()": "django.utils.log.ServerFormatter",
            "format": "[%(server_time)s] %(message)s",
        },
    },
    "handlers": {
        "mail_admins": {
            "level": "ERROR",
            "filters": ["require_debug_false"],
            "class": "django.utils.log.AdminEmailHandler",
            "include_html": True,
        },
        "console": {
            "level": "DEBUG",
            "class": "logging.StreamHandler",
            "formatter": "simple",
        },
        "django.server": {
            "level": "INFO",
            "class": "logging.StreamHandler",
            "formatter": "django.server",
        },
        "syslog": {
            "level": "DEBUG",
            "class": "logging.handlers.SysLogHandler",
            "formatter": "syslog",
            "address": "/dev/log",
            "facility": SysLogHandler.LOG_LOCAL2,
        },
        # Logging to a file
        # "logfile": {
        #     "level":"DEBUG",
        #     "class":"logging.handlers.RotatingFileHandler",
        #     "filename": "/var/log/weblate/weblate.log",
        #     "maxBytes": 100000,
        #     "backupCount": 3,
        #     "formatter": "logfile",
        # },
    },
    "loggers": {
        "django.request": {
            "handlers": ["mail_admins", DEFAULT_LOG],
            "level": "ERROR",
            "propagate": True,
        },
        "django.server": {
            "handlers": ["django.server"],
            "level": "INFO",
            "propagate": False,
        },
        # Logging database queries
        # "django.db.backends": {
        #     "handlers": [DEFAULT_LOG],
        #     "level": "DEBUG",
        # },
        "weblate": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
        # Logging VCS operations
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        "weblate.vcs": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
        # Python Social Auth
        "social": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
        # Django Authentication Using LDAP
        "django_auth_ldap": {"handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL},
    },
}

# Remove syslog setup if it's not present
if not HAVE_SYSLOG:
    del LOGGING["handlers"]["syslog"]

# List of machine translations
MT_SERVICES = (
    #     "weblate.machinery.apertium.ApertiumAPYTranslation",
    #     "weblate.machinery.baidu.BaiduTranslation",
    #     "weblate.machinery.deepl.DeepLTranslation",
    #     "weblate.machinery.glosbe.GlosbeTranslation",
    #     "weblate.machinery.google.GoogleTranslation",
    #     "weblate.machinery.googlev3.GoogleV3Translation",
    #     "weblate.machinery.microsoft.MicrosoftCognitiveTranslation",
    #     "weblate.machinery.microsoftterminology.MicrosoftTerminologyService",
    #     "weblate.machinery.modernmt.ModernMTTranslation",
    #     "weblate.machinery.mymemory.MyMemoryTranslation",
    #     "weblate.machinery.netease.NeteaseSightTranslation",
    #     "weblate.machinery.tmserver.AmagamaTranslation",
    #     "weblate.machinery.tmserver.TMServerTranslation",
    #     "weblate.machinery.yandex.YandexTranslation",
    #     "weblate.machinery.saptranslationhub.SAPTranslationHub",
    #     "weblate.machinery.youdao.YoudaoTranslation",
    "weblate.machinery.weblatetm.WeblateTranslation",
    "weblate.memory.machine.WeblateMemory",
)

# Machine translation API keys

# URL of the Apertium APy server
MT_APERTIUM_APY = None

# DeepL API key
MT_DEEPL_KEY = None

# Microsoft Cognitive Services Translator API, register at
# https://portal.azure.com/
MT_MICROSOFT_COGNITIVE_KEY = None
MT_MICROSOFT_REGION = None

# ModernMT
MT_MODERNMT_KEY = None

# MyMemory identification email, see
# https://mymemory.translated.net/doc/spec.php
MT_MYMEMORY_EMAIL = None

# Optional MyMemory credentials to access private translation memory
MT_MYMEMORY_USER = None
MT_MYMEMORY_KEY = None

# Google API key for Google Translate API v2
MT_GOOGLE_KEY = None
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Google Translate API3 credentials and project id
MT_GOOGLE_CREDENTIALS = None
MT_GOOGLE_PROJECT = None

# Baidu app key and secret
MT_BAIDU_ID = None
MT_BAIDU_SECRET = None

# Youdao Zhiyun app key and secret
MT_YOUDAO_ID = None
MT_YOUDAO_SECRET = None

# Netease Sight (Jianwai) app key and secret
MT_NETEASE_KEY = None
MT_NETEASE_SECRET = None

# API key for Yandex Translate API
MT_YANDEX_KEY = None

# tmserver URL
MT_TMSERVER = None

# SAP Translation Hub
MT_SAP_BASE_URL = None
MT_SAP_SANDBOX_APIKEY = None
MT_SAP_USERNAME = None
MT_SAP_PASSWORD = None
MT_SAP_USE_MT = True

# Title of site to use
SITE_TITLE = "Weblate"

# Site domain
SITE_DOMAIN = ""

# Whether site uses https
ENABLE_HTTPS = False

# Use HTTPS when creating redirect URLs for social authentication, see
# documentation for more details:
# https://python-social-auth-docs.readthedocs.io/en/latest/configuration/settings.
↪html#processing-redirects-and-urlopen
SOCIAL_AUTH_REDIRECT_IS_HTTPS = ENABLE_HTTPS

# Make CSRF cookie HttpOnly, see documentation for more details:
# https://docs.djangoproject.com/en/1.11/ref/settings/#csrf-cookie-httponly
CSRF_COOKIE_HTTPONLY = True
CSRF_COOKIE_SECURE = ENABLE_HTTPS
# Store CSRF token in session
CSRF_USE_SESSIONS = True
# Customize CSRF failure view
CSRF_FAILURE_VIEW = "weblate.trans.views.error.csrf_failure"
SESSION_COOKIE_SECURE = ENABLE_HTTPS
SESSION_COOKIE_HTTPONLY = True
# SSL redirect
SECURE_SSL_REDIRECT = ENABLE_HTTPS
# Sent referrrer only for same origin links
SECURE_REFERRER_POLICY = "same-origin"
# SSL redirect URL exemption list
SECURE_REDIRECT_EXEMPT = (r"healthz/$",)  # Allowing HTTP access to health check
# Session cookie age (in seconds)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
SESSION_COOKIE_AGE = 1209600
# Increase allowed upload size
DATA_UPLOAD_MAX_MEMORY_SIZE = 50000000

# Apply session coookie settings to language cookie as ewll
LANGUAGE_COOKIE_SECURE = SESSION_COOKIE_SECURE
LANGUAGE_COOKIE_HTTPONLY = SESSION_COOKIE_HTTPONLY
LANGUAGE_COOKIE_AGE = SESSION_COOKIE_AGE * 10

# Some security headers
SECURE_BROWSER_XSS_FILTER = True
X_FRAME_OPTIONS = "DENY"
SECURE_CONTENT_TYPE_NOSNIFF = True

# Optionally enable HSTS
SECURE_HSTS_SECONDS = 31536000 if ENABLE_HTTPS else 0
SECURE_HSTS_PRELOAD = ENABLE_HTTPS
SECURE_HSTS_INCLUDE_SUBDOMAINS = ENABLE_HTTPS

# HTTPS detection behind reverse proxy
SECURE_PROXY_SSL_HEADER = None

# URL of login
LOGIN_URL = "{0}/accounts/login/".format(URL_PREFIX)

# URL of logout
LOGOUT_URL = "{0}/accounts/logout/".format(URL_PREFIX)

# Default location for login
LOGIN_REDIRECT_URL = "{0}/".format(URL_PREFIX)

# Anonymous user name
ANONYMOUS_USER_NAME = "anonymous"

# Reverse proxy settings
IP_PROXY_HEADER = "HTTP_X_FORWARDED_FOR"
IP_BEHIND_REVERSE_PROXY = False
IP_PROXY_OFFSET = 0

# Sending HTML in mails
EMAIL_SEND_HTML = True

# Subject of emails includes site title
EMAIL_SUBJECT_PREFIX = "[{0}] ".format(SITE_TITLE)

# Enable remote hooks
ENABLE_HOOKS = True

# By default the length of a given translation is limited to the length of
# the source string * 10 characters. Set this option to False to allow longer
# translations (up to 10.000 characters)
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH = True

# Use simple language codes for default language/country combinations
SIMPLIFY_LANGUAGES = True

# Render forms using bootstrap
CRISPY_TEMPLATE_PACK = "bootstrap3"

# List of quality checks
# CHECK_LIST = (
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#     "weblate.checks.same.SameCheck",
#     "weblate.checks.chars.BeginNewlineCheck",
#     "weblate.checks.chars.EndNewlineCheck",
#     "weblate.checks.chars.BeginSpaceCheck",
#     "weblate.checks.chars.EndSpaceCheck",
#     "weblate.checks.chars.DoubleSpaceCheck",
#     "weblate.checks.chars.EndStopCheck",
#     "weblate.checks.chars.EndColonCheck",
#     "weblate.checks.chars.EndQuestionCheck",
#     "weblate.checks.chars.EndExclamationCheck",
#     "weblate.checks.chars.EndEllipsisCheck",
#     "weblate.checks.chars.EndSemicolonCheck",
#     "weblate.checks.chars.MaxLengthCheck",
#     "weblate.checks.chars.KashidaCheck",
#     "weblate.checks.chars.PuctuationSpacingCheck",
#     "weblate.checks.format.PythonFormatCheck",
#     "weblate.checks.format.PythonBraceFormatCheck",
#     "weblate.checks.format.PHPFormatCheck",
#     "weblate.checks.format.CFormatCheck",
#     "weblate.checks.format.PerlFormatCheck",
#     "weblate.checks.format.JavaScriptFormatCheck",
#     "weblate.checks.format.CSharpFormatCheck",
#     "weblate.checks.format.JavaFormatCheck",
#     "weblate.checks.format.JavaMessageFormatCheck",
#     "weblate.checks.format.PercentPlaceholdersCheck",
#     "weblate.checks.format.I18NextInterpolationCheck",
#     "weblate.checks.format.ESTemplateLiteralsCheck",
#     "weblate.checks.angularjs.AngularJSInterpolationCheck",
#     "weblate.checks.qt.QtFormatCheck",
#     "weblate.checks.qt.QtPluralCheck",
#     "weblate.checks.ruby.RubyFormatCheck",
#     "weblate.checks.consistency.PluralsCheck",
#     "weblate.checks.consistency.SamePluralsCheck",
#     "weblate.checks.consistency.ConsistencyCheck",
#     "weblate.checks.consistency.TranslatedCheck",
#     "weblate.checks.chars.EscapedNewlineCountingCheck",
#     "weblate.checks.chars.NewLineCountCheck",
#     "weblate.checks.markup.BBCodeCheck",
#     "weblate.checks.chars.ZeroWidthSpaceCheck",
#     "weblate.checks.render.MaxSizeCheck",
#     "weblate.checks.markup.XMLValidityCheck",
#     "weblate.checks.markup.XMLTagsCheck",
#     "weblate.checks.markup.MarkdownRefLinkCheck",
#     "weblate.checks.markup.MarkdownLinkCheck",
#     "weblate.checks.markup.MarkdownSyntaxCheck",
#     "weblate.checks.markup.URLCheck",
#     "weblate.checks.markup.SafeHTMLCheck",
#     "weblate.checks.placeholders.PlaceholderCheck",
#     "weblate.checks.placeholders.RegexCheck",
#     "weblate.checks.duplicate.DuplicateCheck",
#     "weblate.checks.source.OptionalPluralCheck",
#     "weblate.checks.source.EllipsisCheck",
#     "weblate.checks.source.MultipleFailingCheck",
#     "weblate.checks.source.LongUntranslatedCheck",
#     "weblate.checks.format.MultipleUnnamedFormatsCheck",
# )

# List of automatic fixups
# AUTOFIX_LIST = (
#     "weblate.trans.autofixes.whitespace.SameBookendingWhitespace",
#     "weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#     "weblate.trans.autofixes.chars.RemoveZeroSpace",
#     "weblate.trans.autofixes.chars.RemoveControlChars",
# )

# List of enabled addons
# WEBLATE_ADDONS = (
#     "weblate.addons.gettext.GenerateMoAddon",
#     "weblate.addons.gettext.UpdateLinguasAddon",
#     "weblate.addons.gettext.UpdateConfigureAddon",
#     "weblate.addons.gettext.MsgmergeAddon",
#     "weblate.addons.gettext.GettextCustomizeAddon",
#     "weblate.addons.gettext.GettextAuthorComments",
#     "weblate.addons.cleanup.CleanupAddon",
#     "weblate.addons.consistency.LangaugeConsistencyAddon",
#     "weblate.addons.discovery.DiscoveryAddon",
#     "weblate.addons.flags.SourceEditAddon",
#     "weblate.addons.flags.TargetEditAddon",
#     "weblate.addons.flags.SameEditAddon",
#     "weblate.addons.flags.BulkEditAddon",
#     "weblate.addons.generate.GenerateFileAddon",
#     "weblate.addons.json.JSONCustomizeAddon",
#     "weblate.addons.properties.PropertiesSortAddon",
#     "weblate.addons.git.GitSquashAddon",
#     "weblate.addons.removal.RemoveComments",
#     "weblate.addons.removal.RemoveSuggestions",
#     "weblate.addons.resx.ResxUpdateAddon",
#     "weblate.addons.yaml.YAMLCustomizeAddon",
#     "weblate.addons.cdn.CDNJSAddon",
#     "weblate.addons.autotranslate.AutoTranslateAddon",
# )

# E-mail address that error messages come from.
SERVER_EMAIL = "noreply@example.com"

# Default email address to use for various automated correspondence from
# the site managers. Used for registration emails.
DEFAULT_FROM_EMAIL = "noreply@example.com"

# List of URLs your site is supposed to serve
ALLOWED_HOSTS = [SITE_DOMAIN]

# Configuration for caching
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/1",
        # If redis is running on same host as Weblate, you might
        # want to use unix sockets instead:
        # "LOCATION": "unix:///var/run/redis/redis.sock?db=1",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
            "PARSER_CLASS": "redis.connection.HiredisParser",
            "PASSWORD": None,
            "CONNECTION_POOL_KWARGS": {},
        },
        "KEY_PREFIX": "weblate",
    },
    "avatar": {
        "BACKEND": "django.core.cache.backends.filebased.FileBasedCache",
        "LOCATION": os.path.join(DATA_DIR, "avatar-cache"),
        "TIMEOUT": 86400,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        "OPTIONS": {"MAX_ENTRIES": 1000},
    },
}

# Store sessions in cache
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
# Store messages in session
MESSAGE_STORAGE = "django.contrib.messages.storage.session.SessionStorage"

# REST framework settings for API
REST_FRAMEWORK = {
    # Use Django's standard `django.contrib.auth` permissions,
    # or allow read-only access for unauthenticated users.
    "DEFAULT_PERMISSION_CLASSES": [
        # Require authentication for login required sites
        "rest_framework.permissions.IsAuthenticated"
        if REQUIRE_LOGIN
        else "rest_framework.permissions.IsAuthenticatedOrReadOnly"
    ],
    "DEFAULT_AUTHENTICATION_CLASSES": (
        "rest_framework.authentication.TokenAuthentication",
        "weblate.api.authentication.BearerAuthentication",
        "rest_framework.authentication.SessionAuthentication",
    ),
    "DEFAULT_THROTTLE_CLASSES": (
        "weblate.api.throttling.UserRateThrottle",
        "weblate.api.throttling.AnonRateThrottle",
    ),
    "DEFAULT_THROTTLE_RATES": {"anon": "100/day", "user": "5000/hour"},
    "DEFAULT_PAGINATION_CLASS": ("rest_framework.pagination.PageNumberPagination"),
    "PAGE_SIZE": 20,
    "VIEW_DESCRIPTION_FUNCTION": "weblate.api.views.get_view_description",
    "UNAUTHENTICATED_USER": "weblate.auth.models.get_anonymous",
}

# Fonts CDN URL
FONTS_CDN_URL = None

# Django compressor offline mode
COMPRESS_OFFLINE = False
COMPRESS_OFFLINE_CONTEXT = [
    {"fonts_cdn_url": FONTS_CDN_URL, "STATIC_URL": STATIC_URL, "LANGUAGE_BIDI":␣
↪True},
    {"fonts_cdn_url": FONTS_CDN_URL, "STATIC_URL": STATIC_URL, "LANGUAGE_BIDI":␣
↪False},
]

# Require login for all URLs
if REQUIRE_LOGIN:
    LOGIN_REQUIRED_URLS = (r"/(.*)$",)

# In such case you will want to include some of the exceptions
# LOGIN_REQUIRED_URLS_EXCEPTIONS = (
#     rf"{URL_PREFIX}/accounts/(.*)$",  # Required for login
#     rf"{URL_PREFIX}/admin/login/(.*)$",  # Required for admin login
#     rf"{URL_PREFIX}/static/(.*)$",  # Required for development mode
#     rf"{URL_PREFIX}/widgets/(.*)$",  # Allowing public access to widgets
#     rf"{URL_PREFIX}/data/(.*)$",  # Allowing public access to data exports
#     rf"{URL_PREFIX}/hooks/(.*)$",  # Allowing public access to notification hooks
#     rf"{URL_PREFIX}/healthz/$",  # Allowing public access to health check
#     rf"{URL_PREFIX}/api/(.*)$",  # Allowing access to API
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#    rf"{URL_PREFIX}/js/i18n/$",  # JavaScript localization
#    rf"{URL_PREFIX}/contact/$",  # Optional for contact form
#    rf"{URL_PREFIX}/legal/(.*)$",  # Optional for legal app
# )

# Silence some of the Django system checks
SILENCED_SYSTEM_CHECKS = [
    # We have modified django.contrib.auth.middleware.AuthenticationMiddleware
    # as weblate.accounts.middleware.AuthenticationMiddleware
    "admin.E408"
]

# Celery worker configuration for testing
# CELERY_TASK_ALWAYS_EAGER = True
# CELERY_BROKER_URL = "memory://"
# CELERY_TASK_EAGER_PROPAGATES = True
# Celery worker configuration for production
CELERY_TASK_ALWAYS_EAGER = False
CELERY_BROKER_URL = "redis://localhost:6379"
CELERY_RESULT_BACKEND = CELERY_BROKER_URL

# Celery settings, it is not recommended to change these
CELERY_WORKER_MAX_MEMORY_PER_CHILD = 200000
CELERY_BEAT_SCHEDULE_FILENAME = os.path.join(DATA_DIR, "celery", "beat-schedule")
CELERY_TASK_ROUTES = {
    "weblate.trans.tasks.auto_translate": {"queue": "translate"},
    "weblate.accounts.tasks.notify_*": {"queue": "notify"},
    "weblate.accounts.tasks.send_mails": {"queue": "notify"},
    "weblate.utils.tasks.settings_backup": {"queue": "backup"},
    "weblate.utils.tasks.database_backup": {"queue": "backup"},
    "weblate.wladmin.tasks.backup": {"queue": "backup"},
    "weblate.wladmin.tasks.backup_service": {"queue": "backup"},
}

# Enable plain database backups
DATABASE_BACKUP = "plain"

# Enable auto updating
AUTO_UPDATE = False

# PGP commits signing
WEBLATE_GPG_IDENTITY = None

# Third party services integration
MATOMO_SITE_ID = None
MATOMO_URL = None
GOOGLE_ANALYTICS_ID = None
SENTRY_DSN = None
AKISMET_API_KEY = None
```

2.18 Команды управления

Примечание: Запуск команд управления под пользователем, отличным от того, под которым работает ваш веб-сервер, может привести к тому, что файлы получат неправильные разрешения, для получения более подробной информации обратитесь к разделу *Разрешения файловой системы*.

Основные команды управления доступны из скрипта `./manage.py` в исходниках Django, или их расширенный набор из скрипта **weblate**, устанавливаемом поверх Weblate.

2.18.1 Вызов команд управления

As mentioned before, invocation depends on how you installed Weblate.

If using virtualenv for Weblate, you can either specify the full path to **weblate**, or activate the virtualenv prior to invoking it:

```
# Direct invocation
~/weblate-env/bin/weblate

# Activating virtualenv adds it to search path
. ~/weblate-env/bin/activate
weblate
```

If you are using source code directly (either from a tarball or Git checkout), the management script is `./manage.py` available in the Weblate sources. To run it:

```
python ./manage.py list_versions
```

If you've installed Weblate using the pip or pip3 installer, or by using the `./setup.py` script, the **weblate** is installed to your path (or virtualenv path), from where you can use it to control Weblate:

```
weblate list_versions
```

For the Docker image, the script is installed like above, and you can run it using **docker exec**:

```
docker exec --user weblate <container> weblate list_versions
```

For **docker-compose** the process is similar, you just have to use **docker-compose exec**:

```
docker-compose exec --user weblate weblate weblate list_versions
```

In case you need to pass it a file, you can temporary add a volume:

```
docker-compose exec --user weblate /tmp:/tmp weblate weblate importusers /tmp/
↪users.json
```

См.также:

Установка с помощью Docker'а, *Установка на Debian и Ubuntu*, *Установка на SUSE и openSUSE*, *Установка на RedHat, Fedora и CentOS*

- *Установка из исходников*, рекомендуется для разработки.

2.18.2 add_suggestions

weblate add_suggestions <project> <component> <language> <file>

Добавлено в версии 2.5.

Imports a translation from the file to use as a suggestion for the given translation. It skips duplicated translations; only different ones are added.

--author USER@EXAMPLE.COM
E-mail of author for the suggestions. This user has to exist prior to importing (you can create one in the admin interface if needed).

Пример:

```
weblate --author michal@cihar.com add_suggestions weblate application cs /tmp/
↪suggestions-cs.po
```

2.18.3 auto_translate

weblate auto_translate <project> <component> <language>

Добавлено в версии 2.5.

Performs automatic translation based on other component translations.

--source PROJECT/COMPONENT
Specifies the component to use as source available for translation. If not specified all components in the project are used.

--user USERNAME
Specify username listed as author of the translations. «Anonymous user» is used if not specified.

--overwrite
Whether to overwrite existing translations.

--inconsistent
Whether to overwrite existing translations that are inconsistent (see *Противоречия*).

--add
Automatically add language if a given translation does not exist.

--mt MT
Use machine translation instead of other components as machine translations.

--threshold THRESHOLD
Similarity threshold for machine translation, defaults to 80.

Пример:

```
weblate auto_translate --user nijel --inconsistent --source weblate/application␣
↪weblate website cs
```

См.также:

Автоматический перевод

2.18.4 celery_queues

weblate celery_queues

Добавлено в версии 3.7.

Displays length of Celery task queues.

2.18.5 checkgit

weblate checkgit <project|project/component>

Prints current state of the back-end Git repository.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

2.18.6 commitgit

weblate commitgit <project|project/component>

Commits any possible pending changes to the back-end Git repository.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

2.18.7 commit_pending

weblate commit_pending <project|project/component>

Commits pending changes older than a given age.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

--age `HOURS`
: Age in hours for committing. If not specified the value configured in *Конфигурация компонента* is used.

Примечание: This is automatically performed in the background by Weblate, so there no real need to invoke this manually, besides forcing an earlier commit than specified by *Конфигурация компонента*.

См.также:

Выполнение задач технического обслуживания, `COMMIT_PENDING_HOURS`

2.18.8 cleanuptrans

weblate cleanuptrans

Cleans up orphaned checks and translation suggestions. There is normally no need to run this manually, as the cleanups happen automatically in the background.

См.также:

Выполнение задач технического обслуживания

2.18.9 createadmin

weblate createadmin

Создаёт учётную запись `admin` со случайным паролем, если он не указан явно.

--password `PASSWORD`
Provides a password on the command-line, to not generate a random one.

--no-password
Do not set password, this can be useful with –update.

--username `USERNAME`
Use the given name instead of `admin`.

--email `USER@EXAMPLE.COM`
Specify the admin e-mail address.

--name
Specify the admin name (visible).

--update
Update the existing user (you can use this to change passwords).

Изменено в версии 2.9: Added parameters `--username`, `--email`, `--name` and `--update`.

2.18.10 dump_memory

weblate dump_memory

Добавлено в версии 2.20.

Export a JSON file containing Weblate Translation Memory content.

См.также:

Память переводов, *Weblate Translation Memory Schema*

2.18.11 dumpuserdata

weblate dumpuserdata <file.json>

Dumps userdata to a file for later use by *importuserdata*

Подсказка: This comes in handy when migrating or merging Weblate instances.

2.18.12 import_demo

weblate import_demo

Добавлено в версии 4.1.

Создаёт демонстрационный проект с компонентами на основе проекта <https://github.com/WeblateOrg/demo>.

This can be useful when developing Weblate.

2.18.13 import_json

weblate import_json <json-file>

Добавлено в версии 2.7.

Batch import of components based on JSON data.

The imported JSON file structure pretty much corresponds to the component object (see *GET /api/components/(string:project)/(string:component)/*). You have to include the `name` and `filemask` fields.

--project `PROJECT`
Specifies where the components will be imported from.

--main-component `COMPONENT`
Use the given VCS repository from this component for all of them.

--ignore
Skip (already) imported components.

--update
Update (already) imported components.

Изменено в версии 2.9: The parameters `--ignore` and `--update` are there to deal with already imported components.

Example of JSON file:

```
[
  {
    "slug": "po",
    "name": "Gettext PO",
    "file_format": "po",
    "filemask": "po/*.po",
    "new_lang": "none"
  },
  {
    "name": "Android",
    "filemask": "android/values-*/strings.xml",
    "template": "android/values/strings.xml",
    "repo": "weblate://test/test",
    "file_format": "aresource"
  }
]
```

См.также:

import_memory

2.18.14 import_memory

weblate import_memory <file>

Добавлено в версии 2.20.

Imports a TMX or JSON file into the Weblate translation memory.

--language-map `LANGMAP`
Allows mapping languages in the TMX to the Weblate translation memory. The language codes are mapped after normalization usually done by Weblate.

`--language-map en_US:en` will for example import all `en_US` strings as `en` ones.

This can be useful in case your TMX file locales happen not to match what you use in Weblate.

См.также:

Память переводов, *Weblate Translation Memory Schema*

2.18.15 import_project

weblate import_project <project> <gitrepo> <branch> <filemask>

Изменено в версии 3.0: The import_project command is now based on the *Обнаружение компонента* addon, leading to some changes in behavior and what parameters are accepted.

Batch imports components into project based on filemask.

<project> names an existing project, into which the components are to be imported.

The *<gitrepo>* defines the Git repository URL to use, and *<branch>* signifies the Git branch. To import additional translation components from an existing Weblate component, use a *weblate://<project>/<component>* URL for the *<gitrepo>*.

The *<filemask>* defines file discovery for the repository. It can be either be made simple using wildcards, or it can use the full power of regular expressions.

The simple matching uses `**` for component name and `*` for language, for example: `**/*.po`

The regular expression has to contain groups named *component* and *language*. For example: `(?P<language>[^/]*)/(?P<component>[^-/]*)\.po`

The import matches existing components based on files and adds the ones that do not exist. It does not change already existing ones.

--name-template TEMPLATE
Customize the name of a component using Django template syntax.

For example: `Documentation: {{ component }}`

--base-file-template TEMPLATE
Customize the base file for monolingual translations.

For example: `{{ component }}/res/values/string.xml`

--new-base-template TEMPLATE
Customize the base file for addition of new translations.

For example: `{{ component }}/ts/en.ts`

--file-format FORMAT
You can also specify the file format to use (see *Поддерживаемые форматы файлов*), the default is auto-detection.

--language-regex REGEX
You can specify language filtering (see *Конфигурация компонента*) with this parameter. It has to be a valid regular expression.

--main-component
You can specify which component will be chosen as the main one—the one actually containing the VCS repository.

--license NAME
Specify the overall, project or component translation license.

--license-url URL
Specify the URL where the translation license is to be found.

--vcs NAME
In case you need to specify which version control system to use, you can do it here. The default version control is Git.

To give you some examples, let's try importing two projects.

First The Debian Handbook translations, where each language has separate a folder with the translations of each chapter:

```
weblate import_project \
    debian-handbook \
    git://anonscm.debian.org/debian-handbook/debian-handbook.git \
    squeeze/master \
    '*/**.po'
```

Then the Tanaguru tool, where the file format needs be specified, along with the base file template, and how all components and translations are located in single folder:

```
weblate import_project \
    --file-format=properties \
    --base-file-template=web-app/tgol-web-app/src/main/resources/i18n/%s-I18N.
↪properties \
    tanaguru \
    https://github.com/Tanaguru/Tanaguru \
    master \
    web-app/tgol-web-app/src/main/resources/i18n/**-I18N_*.properties
```

More complex example of parsing of filenames to get the correct component and language out of a filename like `src/security/Numerous_security_holes_in_0.10.1.de.po`:

```
weblate import_project \
    tails \
    git://git.tails.boum.org/tails master \
    'wiki/src/security/(?P<component>.*)\.(?P<language>[^.]*)\.po$'
```

Filtering only translations in a chosen language:

```
./manage import_project \
    --language-regex '^(cs|sk)$' \
    weblate \
    https://github.com/WeblateOrg/weblate.git \
    'weblate/locale/*/LC_MESSAGES/**.po'
```

Importing Sphinx documentation split to multiple files:

```
$ weblate import_project --name-template 'Documentation: %s' \
    --file-format po \
    project https://github.com/project/docs.git master \
    'docs/locale/*/LC_MESSAGES/**.po'
```

Importing Sphinx documentation split to multiple files and directories:

```
$ weblate import_project --name-template 'Directory 1: %s' \
    --file-format po \
    project https://github.com/project/docs.git master \
    'docs/locale/*/LC_MESSAGES/dir1/**.po'
$ weblate import_project --name-template 'Directory 2: %s' \
    --file-format po \
    project https://github.com/project/docs.git master \
    'docs/locale/*/LC_MESSAGES/dir2/**.po'
```

См.также:

More detailed examples can be found in the *Знакомство с интернационализацией* chapter, alternatively you might want to use `import_json`.

2.18.16 importuserdata

weblate importuserdata <file.json>

Imports user data from a file created by *dumpuserdata*

2.18.17 importusers

weblate importusers --check <file.json>

Imports users from JSON dump of the Django auth_users database.

--check
With this option it will just check whether a given file can be imported and report possible conflicts arising from usernames or e-mails.

You can dump users from the existing Django installation using:

```
weblate dumpdata auth.User > users.json
```

2.18.18 install_addon

Добавлено в версии 3.2.

weblate install_addon --addon ADDON <project|project/component>

Installs an addon to a set of components.

--addon ADDON
Name of the addon to install. For example `weblate.gettext.customize`.

--configuration CONFIG
JSON encoded configuration of an addon.

--update
Update the existing addon configuration.

You can either define which project or component to install the addon in (for example `weblate/application`), or use `--all` to include all existing components.

To install *Настроить вывод gettext* for all components:

```
weblate install_addon --addon weblate.gettext.customize --config '{"width": -1}' --
↪update --all
```

См.также:

Надстройки

2.18.19 list_languages

weblate list_languages <locale>

Lists supported languages in MediaWiki markup - language codes, English names and localized names.

This is used to generate <https://wiki.l10n.cz/Jazyky>.

2.18.20 list_translators

weblate list_translators <project|project/component>

Lists translators by contributed language for the given project:

```
[French]
Jean Dupont <jean.dupont@example.com>
[English]
John Doe <jd@example.com>
```

--language-code
List names by language code instead of language name.

You can either define which project or component to use (for example `weblate/application`), or use `--all` to list translators from all existing components.

2.18.21 list_versions

weblate list_versions

Lists all Weblate dependencies and their versions.

2.18.22 loadpo

weblate loadpo <project|project/component>

Reloads translations from disk (for example in case you have done some updates in the VCS repository).

--force
Force update, even if the files should be up-to-date.

--lang `LANGUAGE`
Limit processing to a single language.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Примечание: You seldom need to invoke this, Weblate will automatically load changed files for every VCS update. This is needed in case you manually changed an underlying Weblate VCS repository or in some special cases following an upgrade.

2.18.23 lock_translation

weblate lock_translation <project|project/component>

Prevents further translation of a component.

Подсказка: Useful in case you want to do some maintenance on the underlying repository.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

См.также:

unlock_translation

2.18.24 move_language

weblate move_language source target

Добавлено в версии 3.0.

Allows you to merge language content. This is useful when updating to a new version which contains aliases for previously unknown languages that have been created with the *(generated)* suffix. It moves all content from the *source* language to the *target* one.

Пример:

```
weblate move_language cze cs
```

After moving the content, you should check whether there is anything left (this is subject to race conditions when somebody updates the repository meanwhile) and remove the *(generated)* language.

2.18.25 pushgit

weblate pushgit <project|project/component>

Pushes committed changes to the upstream VCS repository.

--force-commit
Force commits any pending changes, prior to pushing.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Примечание: Weblate pushes changes automatically if *Push on commit* in *Конфигурация компонента* is turned on, which is the default.

2.18.26 unlock_translation

weblate unlock_translation <project|project/component>

Unlocks a given component, making it available for translation.

Подсказка: Useful in case you want to do some maintenance on the underlying repository.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

См.также:

lock_translation

2.18.27 setupgroups

weblate setupgroups

Configures default groups and optionally assigns all users to that default group.

--no-privs-update
Turns off automatic updating of existing groups (only adds new ones).

--no-projects-update
Prevents automatic updates of groups for existing projects. This allows adding newly added groups to existing projects, see *По-проектный контроль доступа*.

См.также:

Управление доступом

2.18.28 setuplang

weblate setuplang

Updates list of defined languages in Weblate.

--no-update
Turns off automatic updates of existing languages (only adds new ones).

2.18.29 updatechecks

weblate updatechecks <project|project/component>

Updates all checks for all strings.

Подсказка: Useful for upgrades which do major changes to checks.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

2.18.30 updategit

weblate updategit <project|project/component>

Fetches remote VCS repositories and updates the internal cache.

You can either define which project or component to update (for example `weblate/application`), or use `--all` to update all existing components.

Примечание: Usually it is better to configure hooks in the repository to trigger *Обработчики уведомлений*, instead of regular polling by `updategit`.

2.19 Объявления

Изменено в версии 4.0: In prior releases this feature was called whiteboard messages.

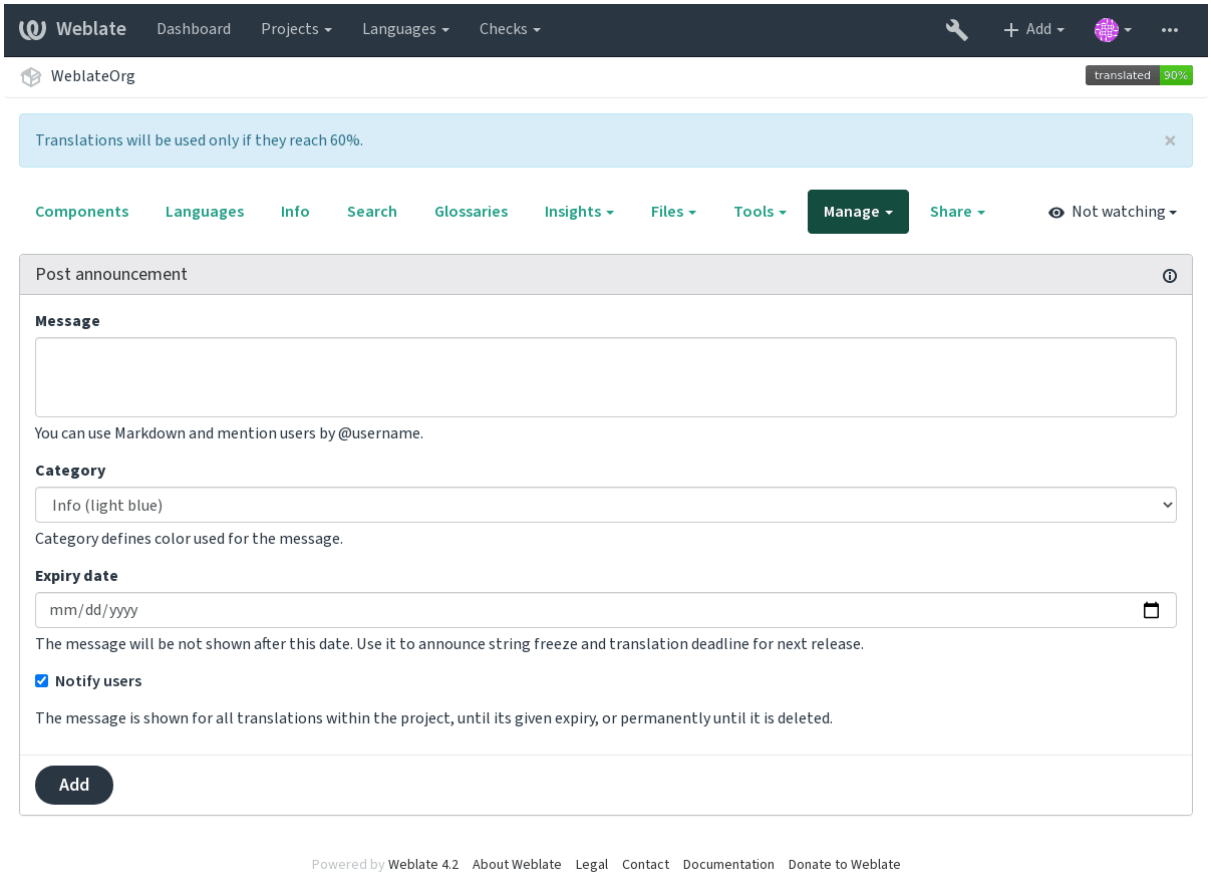
Provide info to your translators by posting announcements, site-wide, per project, component, or language.

Announce the purpose, deadlines, status, or specify targets for translation.

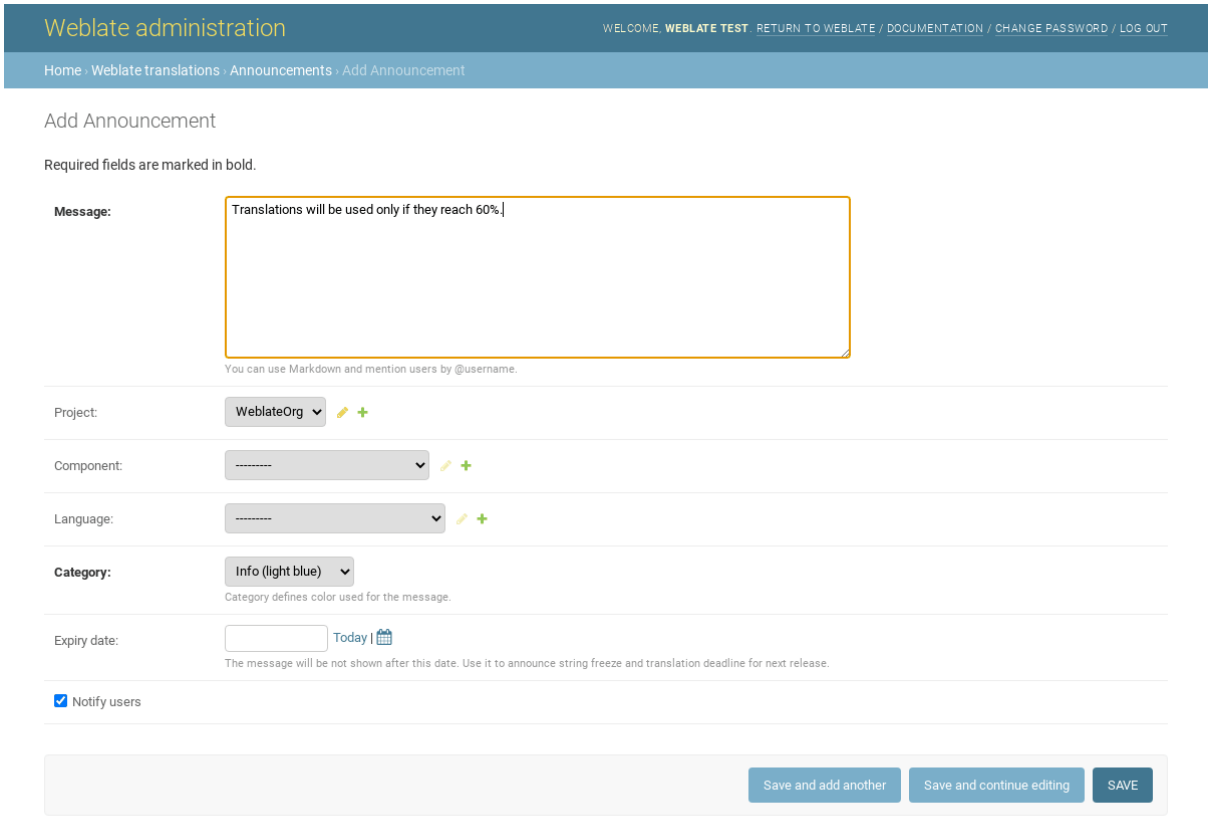
The users will receive notification on the announcements for watched projects (unless they opt out).

This can be useful for various things from announcing the purpose of the website to specifying targets for translations.

The announcements can posted on each level in the *Manage* menu, using *Post announcement*:



It can be also added using the admin interface:



The announcements are then shown based on their specified context:

No context specified

Shown on dashboard (landing page).

Project specified

Shown within the project, including all its components and translations.

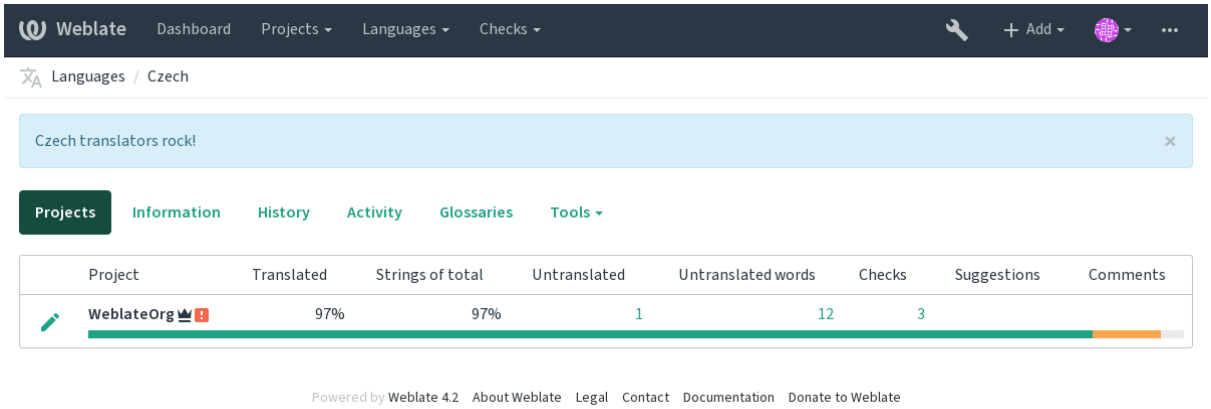
Component specified

Shown for a given component and all its translations.

Language specified

Shown on the language overview and all translations in that language.

This is how it looks on the language overview page:



2.20 Списки компонентов

Укажите несколько списков компонентов, чтобы они появились в качестве вариантов выбора на панели управления пользователя, из которых пользователи смогут выбрать один из них, как своё умолчательное представление панели. Для получения более подробной информации смотрите раздел *Панель управления*.

Изменено в версии 2.20: Для каждого списка компонентов, присутствующего в панели управления, будет представлен свой статус.

Названия и содержимое списков компонентов можно указать в интерфейсе администратора в разделе *Списки компонентов*. Каждый список компонентов должен иметь название, отображаемое пользователю, и плашку, представляющую его в URL-адресе.

Изменено в версии 2.13: Для анонимных пользователей настройки панели управления изменяются из интерфейса администратора, в котором определяется, какая панель управления будет показана неаутентифицированным пользователям.

2.20.1 Автоматические списки компонентов

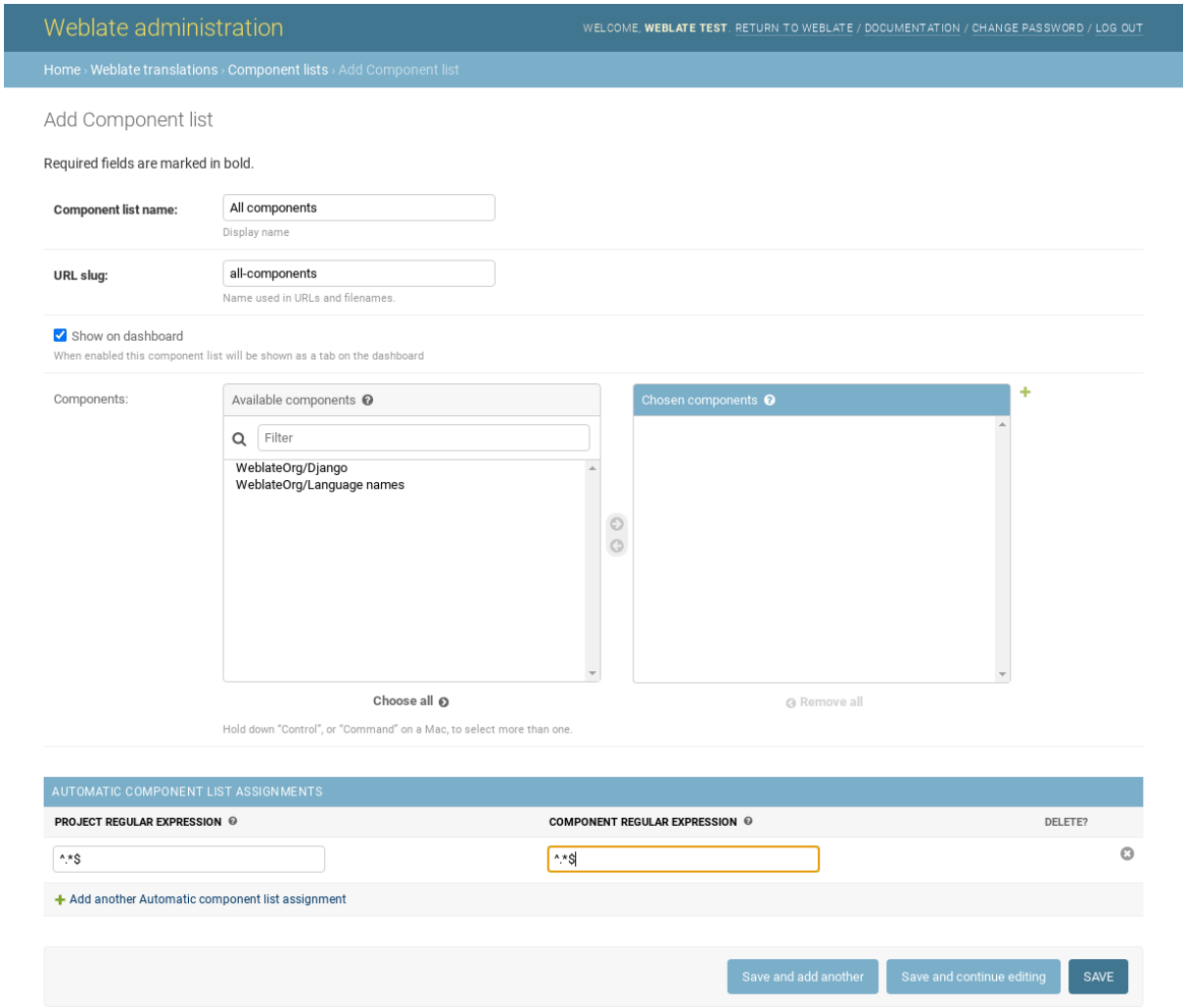
Добавлено в версии 2.13.

Автоматическое добавление компонентов в список на основе их плашки настраивается созданием правил *Автоматического прикрепления списка компонентов*.

- Useful for maintaining component lists for large installations, or in case you want to have one component list with all components on your Weblate installation.

Подсказка: Создайте список компонентов, содержащий все компоненты вашей установки Weblate.

1. Define *Automatic component list assignment* with `^.*$` as regular expression in both the project and the component fields, as shown on this image:



2.21 Optional Weblate modules

Several optional modules are available for your setup.

2.21.1 Git exporter

Добавлено в версии 2.10.

Provides you read-only access to the underlying Git repository using HTTP(S).

Установка

1. Add `weblate.gitexport` to installed apps in `settings.py`:

```
INSTALLED_APPS += (
    'weblate.gitexport',
)
```

2. Export existing repositories by migrating your database after installation:

```
weblate migrate
```

Использование

The module automatically hooks into Weblate and sets the exported repository URL in the *Конфигурация компонента*. The repositories are accessible under the `/git/` part of the Weblate URL, for example `https://example.org/git/weblate/master/`:

```
git clone 'https://example.org/git/weblate/master/'
```

Repositories are available anonymously unless *По-проектный контроль доступа* is turned on. This requires authenticate using your API token (it can be obtained in your *Профиль пользователя*):

```
git clone 'https://user:KEY@example.org/git/weblate/master/'
```

2.21.2 Биллинг

Добавлено в версии 2.4.

This is used on Hosted Weblate to define billing plans, track invoices and usage limits.

Установка

1. Add `weblate.billing` to installed apps in `settings.py`:

```
INSTALLED_APPS += (
    'weblate.billing',
)
```

2. Run the database migration to optionally install additional database structures for the module:

```
weblate migrate
```

Использование

After installation you can control billing in the admin interface. Users with billing enabled will get new *Billing* tab in their *Профиль пользователя*.

The billing module additionally allows project admins to create new projects and components without being superusers (see *Adding translation projects and components*). This is possible when following conditions are met:

- The billing is in its configured limits (any overusage results in blocking of project/component creation) and paid (if its price is non zero)
- The user is admin of existing project with billing or user is owner of billing (the latter is necessary when creating new billing for users to be able to import new projects).

Upon project creation user is able to choose which billing should be charged for the project in case he has access to more of them.

2.21.3 Право

Добавлено в версии 2.15.

This is used on Hosted Weblate to provide required legal documents. It comes provided with blank documents, and you are expected to fill out the following templates in the documents:

legal/documents/tos.html Terms of service document

legal/documents/privacy.html Privacy policy document

legal/documents/summary.html Short overview of the terms of service and privacy policy

---

Примечание: Legal documents for the Hosted Weblate service is available in this Git repository <https://github.com/WeblateOrg/hosted/tree/master/wlhosted/legal/templates/legal/documents>.

Most likely these will not be directly usable to you, but might come in handy as a starting point if adjusted to meet your needs.

Установка

1. Add `weblate.legal` to installed apps in `settings.py`:

```
INSTALLED_APPS += (
    'weblate.legal',
)

# Optional:

# Social auth pipeline to confirm TOS upon registration/subsequent login
SOCIAL_AUTH_PIPELINE += (
    'weblate.legal.pipeline.tos_confirm',
)

# Middleware to enforce TOS confirmation of signed in users
MIDDLEWARE += [
    'weblate.legal.middleware.RequireTOSMiddleware',
]
```

2. Run the database migration to optionally install additional database structures for the module:

```
weblate migrate
```

3. Edit the legal documents in the `weblate/legal/templates/legal/` folder to match your service.

Использование

After installation and editing, the legal documents are shown in the Weblate UI.

2.21.4 Аватары

Avatars are downloaded and cached server-side to reduce information leaks to the sites serving them by default. The built-in support for fetching avatars from e-mails addresses configured for it can be turned off using `ENABLE_AVATARS`.

Weblate currently supports:

- Gravatar

См.также:

Кеширование аватара, `AVATAR_URL_PREFIX`, `ENABLE_AVATARS`

2.21.5 Spam protection

You can protect against suggestion spamming by unauthenticated users by using the akismet.com service.

1. Install the *akismet* Python module
2. Configure the Akismet API key.

Примечание: This (among other things) relies on IP address of the client, please see *Работа за обратным прокси* for properly configuring that.

См.также:

Работа за обратным прокси, `AKISMET_API_KEY`

2.21.6 Signing Git commits with GnuPG

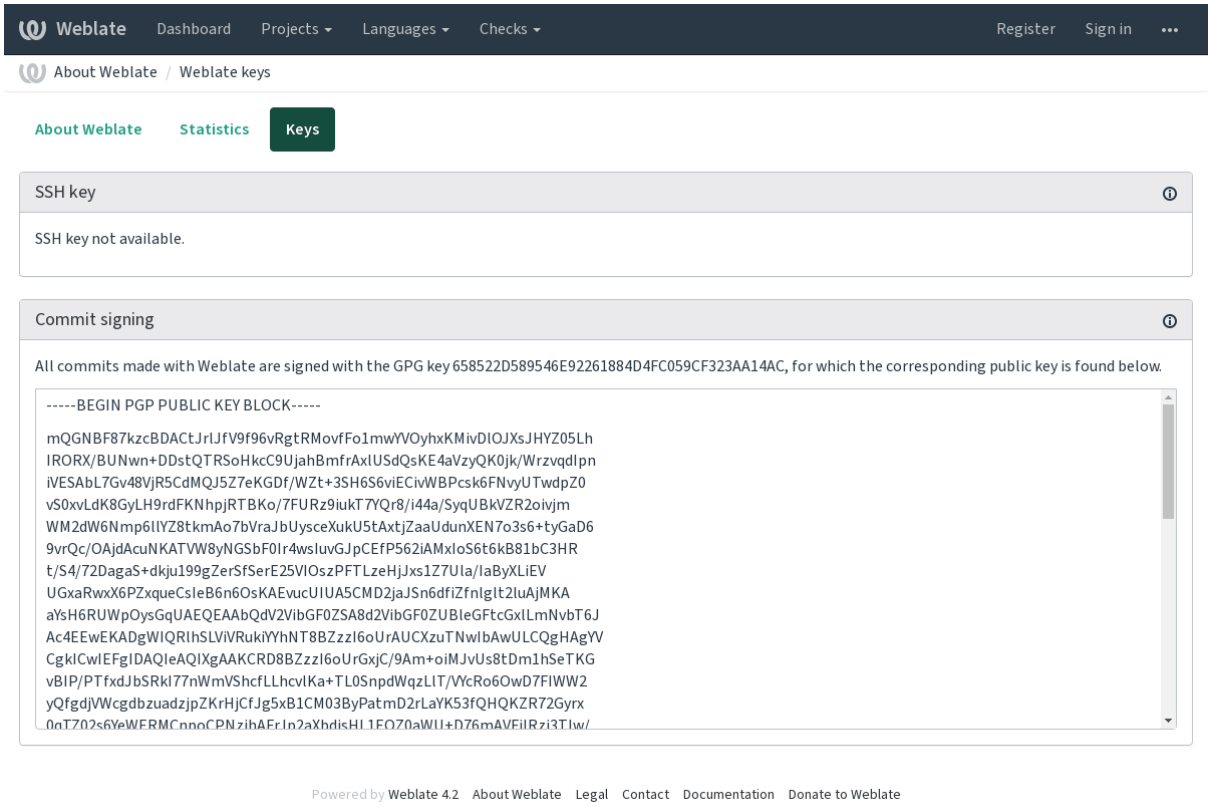
Добавлено в версии 3.1.

All commits can be signed by the GnuPG key of the Weblate instance.

1. Turn on `WEBLATE_GPG_IDENTITY`. (Weblate will generate a GnuPG key when needed and will use it to sign all translation commits.)

This feature needs GnuPG 2.1 or newer installed.

You can find the key in the `DATA_DIR` and the public key is shown on the «About» page:



2. Alternatively you can also import existing keys into Weblate, just set `HOME=$DATA_DIR/home` when invoking gpg.

См.также:

WEBLATE_GPG_IDENTITY

2.21.7 Ограничение частоты запросов

Изменено в версии 3.2: The rate limiting now accepts more fine-grained configuration.

Several operations in Weblate are rate limited. At most *RATELIMIT_ATTEMPTS* attempts are allowed within *RATELIMIT_WINDOW* seconds. The user is then blocked for *RATELIMIT_LOCKOUT*. There are also settings specific to scopes, for example `RATELIMIT_CONTACT_ATTEMPTS` or `RATELIMIT_TRANSLATE_ATTEMPTS`. The table below is a full list of available scopes.

The following operations are subject to rate limiting:

Имя	Область	Allowed attempts	Ratelimit window	Lockout period
Регистрация	`REGISTRATION`	5	300	600
Sending message to admins	`MESSAGE`	5	300	600
Password authentication on login	`LOGIN`	5	300	600
Sitewide search	`SEARCH`	6	60	60
Перевод	`TRANSLATE`	30	60	600
Adding to glossary	`GLOSSARY`	30	60	600

If a user fails to log in *AUTH_LOCK_ATTEMPTS* times, password authentication will be turned off on the account until having gone through the process of having its password reset.

См.также:

Ограничение частоты запросов, *Работа за обратным прокси*

2.22 Customizing Weblate

Extend and customize using Django and Python. Contribute your changes upstream so that everybody can benefit. This reduces your maintenance costs; code in Weblate is taken care of when changing internal interfaces or refactoring the code.

Предупреждение: Neither internal interfaces nor templates are considered a stable API. Please review your own customizations for every upgrade, the interfaces or their semantics might change without notice.

См.также:

Внесение вклада в Weblate

2.22.1 Creating a Python module

If you are not familiar with Python, you might want to look into Python For Beginners, explaining the basics and pointing to further tutorials.

To write some custom Python code (called a module), a place to store it is needed, either in the system path (usually something like `/usr/lib/python3.7/site-packages/`) or in the Weblate directory, which is also added to the interpreter search path.

Better yet, turn your customization into a proper Python package:

1. Create a folder for your package (we will use *weblate_customization*).
2. within it, create a `setup.py` file to describe the package:

```
from setuptools import setup

setup(
    name = "weblate_customization",
    version = "0.0.1",
    author = "Your name",
    author_email = "yourname@example.com",
    description = "Sample Custom check for Weblate.",
    license = "GPLv3+",
    keywords = "Weblate check example",
    packages=['weblate_customization'],
)
```

3. Create a folder for the Python module (also called `weblate_customization`) for the customization code.
4. Within it, create a `__init__.py` file to make sure Python can import the module.
5. This package can now be installed using **`pip install -e`**. More info to be found in "Editable" Installs.
6. Once installed, the module can be used in the Weblate configuration (for example `weblate_customization.checks.FooCheck`).

Your module structure should look like this:

```
weblate_customization
├── setup.py
└── weblate_customization
    ├── __init__.py
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    ├── addons.py
    └── checks.py
```

You can find an example of customizing Weblate at <https://github.com/WeblateOrg/customize-example>, it covers all the topics described below.

2.22.2 Changing the logo

1. Create a simple Django app containing the static files you want to overwrite (see *Creating a Python module*).
2. Добавьте его в `INSTALLED_APPS`:

```
INSTALLED_APPS = (
   # Add your customization as first
   'weblate_customization',

   # Weblate apps are here…
)
```

Branding appears in the following files:

`icons/weblate.svg` Logo shown in the navigation bar.

`logo-*.png` Web icons depending on screen resolution and web-browser.

`favicon.ico` Web icon used by legacy browsers.

`weblate-*.png` Avatars for bots or anonymous users. Some web-browsers use these as shortcut icons.

`email-logo.png` Used in notifications e-mails.

3. Run `weblate collectstatic --noinput`, to collect static files served to clients.

См.также:

Управление статическими файлами (изображениями, JavaScript, CSS и прочими подобными), *Обслуживание статических файлов*

2.22.3 Custom quality checks, addons and auto-fixes

To install your code for *Пользовательские автоматические исправления*, *Writing own checks* or *Writing addon* and in Weblate:

1. Place the files in your Python module containing the Weblate customization (see *Creating a Python module*).
2. Add its fully-qualified path to the Python class in the dedicated settings (*`WEBLATE_ADDONS`*, *`CHECK_LIST`* or *`AUTOFIX_LIST`*):

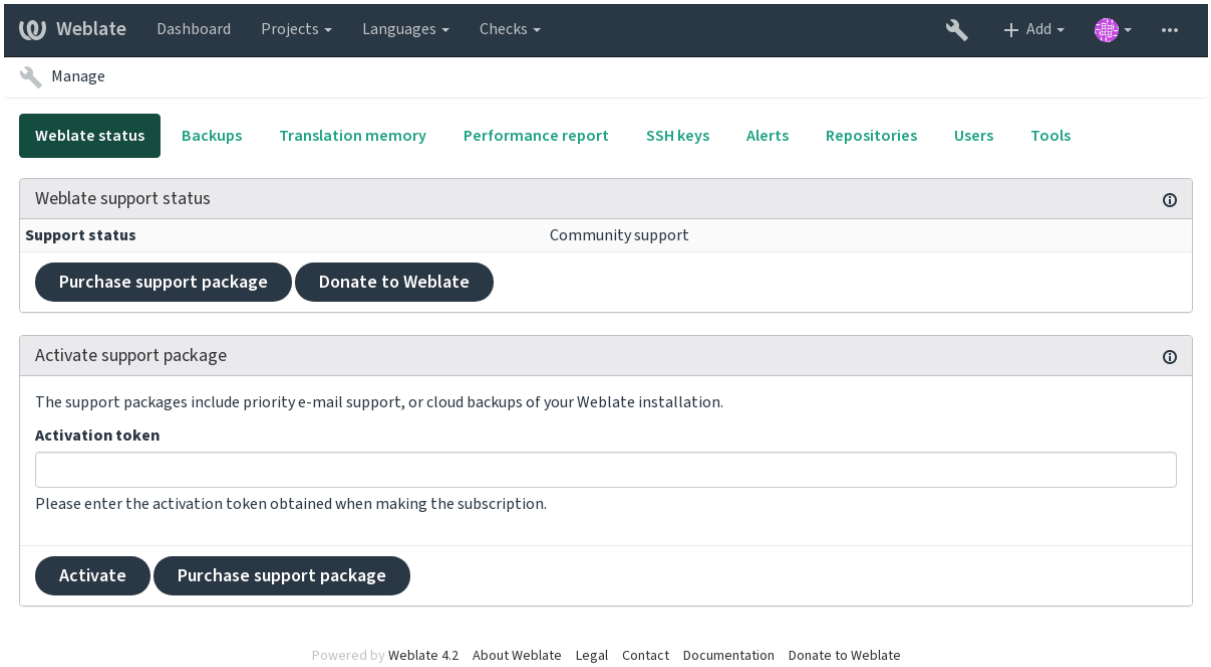
```
# Checks
CHECK_LIST += (
    'weblate_customization.checks.FooCheck',
)

# Autofixes
AUTOFIX_LIST += (
  'weblate_customization.autofix.FooFixer',
)

# Addons
WEBLATE_ADDONS += (
  'weblate_customization.addons.ExamplePreAddon',
)
```

См.также:

Пользовательские автоматические исправления:, *Writing own checks*, *Writing addon*, *Executing scripts from addon*

2.23 Интерфейс управления

The management interface offer administration settings under the `/management/` URL. It is available for users signed in with admin privileges, accessible by using the wrench icon top right:



2.23.1 The Django admin interface

> **Предупреждение:** Will be removed in the future, as its use is discouraged—most features can be managed directly in Weblate.

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

Weblate administration

WELCOME, **WEBLATE TEST**. RETURN TO WEBLATE / DOCUMENTATION / CHANGE PASSWORD / LOG OUT

Site administration

REPORTS
Weblate support status
Status of repositories
SSH keys
Performance report
Translation memory

ACCOUNTS		
Audit logs	+ Add	Change
Profiles	+ Add	Change
Verified emails	+ Add	Change

AUTH TOKEN		
Tokens	+ Add	Change

AUTHENTICATION		
Groups	+ Add	Change
Roles	+ Add	Change
Users	+ Add	Change

BILLING		
Billings	+ Add	Change
Invoices	+ Add	Change
Plans	+ Add	Change

FONTS		
Font groups	+ Add	Change
Fonts	+ Add	Change

GLOSSARIES		
Glossaries	+ Add	Change

LEGAL		
Agreements	+ Add	Change

PYTHON SOCIAL AUTH		
Associations	+ Add	Change
Nonces	+ Add	Change
User social auths	+ Add	Change

SCREENSHOTS		
Screenshots	+ Add	Change

TRANSLATION MEMORY		
Memorys	+ Add	Change

WEBLATE LANGUAGES		
Languages	+ Add	Change

WEBLATE TRANSLATIONS		
Announcements	+ Add	Change
Component lists	+ Add	Change
Components	+ Add	Change
Contributor agreements	+ Add	Change
Projects	+ Add	Change

Recent actions

My actions

None available

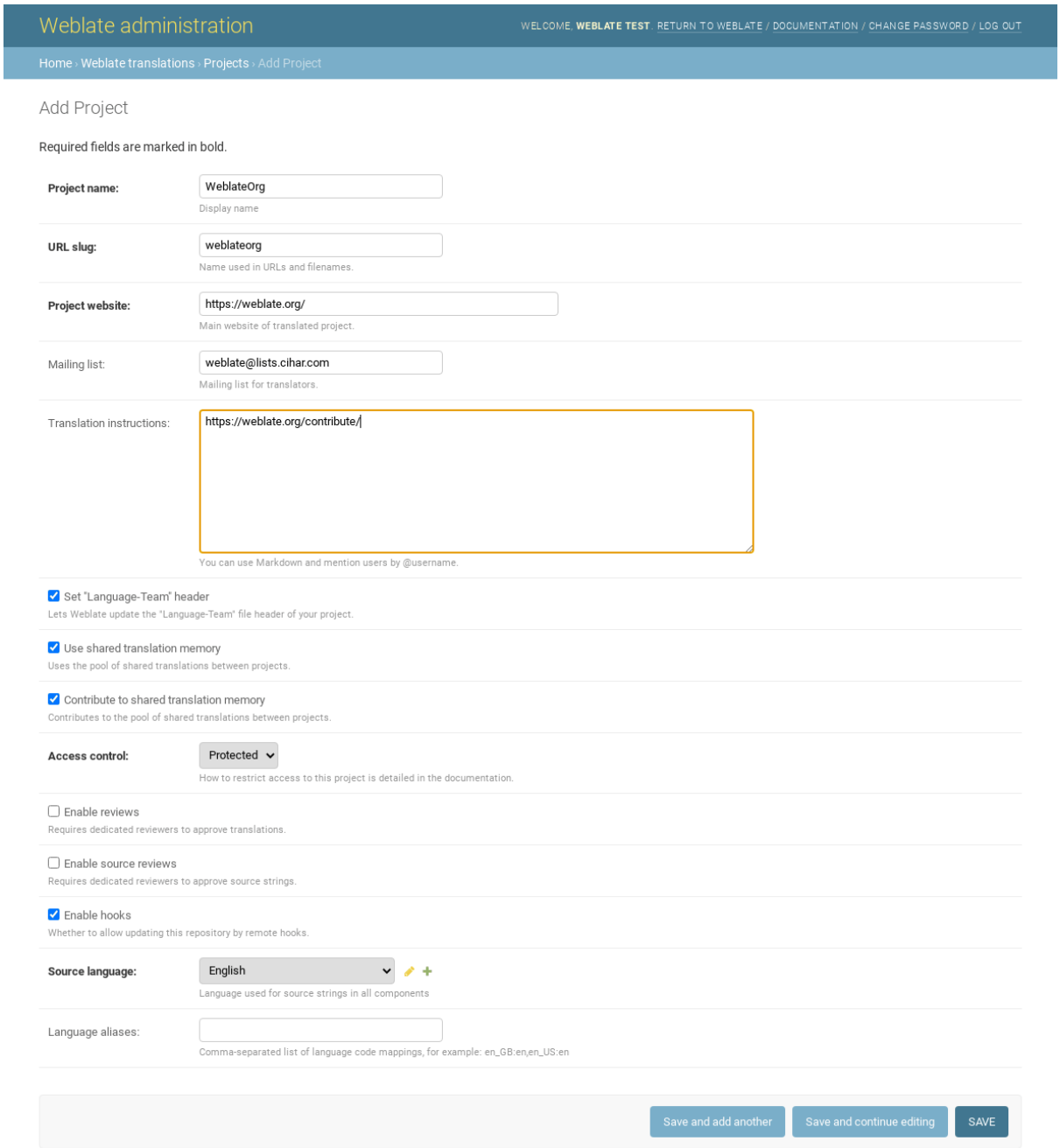
In the *Reports* section, you can check the status of your site, tweak it for *Рабочая среда*, or manage SSH keys used to access *Доступ к репозиториям*.

Manage database objects under any of the sections. The most interesting one is probably *Weblate translations*, where you can manage translatable projects, see *Project configuration* and *Конфигурация компонента*.

Weblate languages holds language definitions, explained further in *Определения языка*.

Adding a project

Adding a project serves as container for all components. Usually you create one project for one piece of software, or book (See *Project configuration* for info on individual parameters):



См.также:

Project configuration

Двуязычные компоненты

Once you have added a project, translation components can be added to it. (See *Конфигурация компонента* for info regarding individual parameters):

Weblate administration

WELCOME, WEBLATE TEST. RETURN TO WEBLATE / DOCUMENTATION / CHANGE PASSWORD / LOG OUT

Home › Weblate translations › Components › Add Component

Add Component

IMPORT SPEED DOCUMENTATION

Required fields are marked in bold.

Component name: Language names
Display name

URL slug: language-names
Name used in URLs and filenames.

Project: WeblateOrg

Version control system: Git
Version control system to use to access your repository containing translations. You can also choose additional integration with third-party providers to submit merge requests.

Source code repository: https://github.com/WeblateOrg/demo.git
URL of a repository, use weblate://project/component to share it with other component.

Repository push URL:
URL of a push repository, pushing is turned off if empty.

Repository browser: https://github.com/WeblateOrg/demo/blob/{{branch}}/{{filename}}#L{
Link to repository browser, use {{branch}} for branch, {{filename}} and {{line}} as filename and line placeholders.

Exported repository URL:
URL of repository where users can fetch changes from Weblate.

Source string bug reporting address:
E-mail address for reports on errors in source strings. Leave empty for no e-mails.

Repository branch:
Repository branch to translate

Push branch:
Branch for pushing changes, leave empty to use repository branch.

Filemask: weblate/langdata/locale/*/LC_MESSAGES/djs
Path of files to translate relative to repository root, use * instead of language code, for example: po/* po or locale/*/LC_MESSAGES/django.po.

Monolingual base language file:
Filename of translation base file, containing all strings and their source; it is recommended for monolingual translation formats.

☑ Edit base file
Whether users will be able to edit the base file for monolingual translations.

Intermediate language file:
Filename of intermediate translation file. In most cases this is a translation file provided by developers and is used when creating actual source strings.

Template for new translations: weblate/langdata/locale/django.pot
Filename of file used for creating new translations. For gettext choose .pot file.

File format: gettext PO file

☐ Locked
Locked component will not get any translation updates.

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

☑ Turn on suggestions
Whether to allow translation suggestions at all.

☐ Suggestion voting
Whether users can vote for suggestions.

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

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

Enforced checks: []
List of checks which can not be ignored.

Translation license: GNU General Public License v3.0 or later

Contributor agreement:
User agreement which needs to be approved before a user can translate this component.

Adding new translation: Create new language file
How to handle requests for creating new translations.

Language code style: Default based on the file format
Customize language code used to generate the filename for translations created by Weblate.

Merge style: Rebase
Define whether Weblate should merge the upstream repository or rebase changes onto it.

Commit message when translating:
Translated using Weblate ({{ language_name }})

Currently translated at {{ stats.translated_percent }}% ({{ stats.translated }} of {{ stats.all }} strings)

Translation: {{ project_name }}/{{ component_name }}
Translate-URL: {{ url }}

You can use template language for various info, please consult the documentation for more details.

Commit message when adding translation:
Added translation using Weblate ({{ language_name }})

You can use template language for various info, please consult the documentation for more details.

Commit message when removing translation:
Deleted translation using Weblate ({{ language_name }})

You can use template language for various info, please consult the documentation for more details.

Commit message when merging translation:
Merge branch '{{ component_remote_branch }}' into Weblate.

You can use template language for various info, please consult the documentation for more details.

Commit message when addon makes a change:
Update translation files

Updated by "{{ addon_name }}" hook in Weblate.

Translation: {{ project_name }}/{{ component_name }}
Translate-URL: {{ url }}

You can use template language for various info, please consult the documentation for more details.

Committer name: Weblate

Committer e-mail: noreply@weblate.org

☑ Push on commit
Whether the repository should be pushed upstream on every commit.

Age of changes to commit: 24
Time in hours after which any pending changes will be committed to the VCS.

☑ Lock on error
Whether the component should be locked on repository errors.

Language filter: ^(cs|he|hu)$
Regular expression used to filter translation when scanning for filemask.

Variants regular expression:
Regular expression used to determine variants of a string.

Priority: Medium
Components with higher priority are offered first to translators.

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

Save and add another | Save and continue editing | SAVE

См.также:

Конфигурация компонента, *Двуязычные и одноязычные форматы*

Одноязычные компоненты

For easier translation of these, provide a template file containing the mapping of message IDs to its respective source language (usually English). (See *Конфигурация компонента* for info regarding individual parameters):

Weblate administration

WELCOME, WEBLATE TEST. RETURN TO WEBLATE / DOCUMENTATION / CHANGE PASSWORD / LOG OUT

Home › Weblate translations › Components › Add Component

Add Component

IMPORT SPEED DOCUMENTATION

Required fields are marked in bold.

Component name: Android
Display name

URL slug: android
Name used in URLs and filenames.

Project: WeblateOrg

Version control system: Git
Version control system to use to access your repository containing translations. You can also choose additional integration with third-party providers to submit merge requests.

Source code repository: weblate://weblateorg/language-names
URL of a repository, use weblate://project/component to share it with other component.

Repository push URL:
URL of a push repository, pushing is turned off if empty.

Repository browser:
Link to repository browser, use {{branch}} for branch, {{filename}} and {{line}} as filename and line placeholders.

Exported repository URL:
URL of repository where users can fetch changes from Weblate.

Source string bug reporting address:
E-mail address for reports on errors in source strings. Leave empty for no e-mails.

Repository branch:
Repository branch to translate

Push branch:
Branch for pushing changes, leave empty to use repository branch.

Filemask: app/src/main/res/values-*/strings.xml
Path of files to translate relative to repository root, use * instead of language code, for example: po/*.po or locale/*/LC_MESSAGES/djangojs.po.

Monolingual base language file: app/src/main/res/values/strings.xml
Filename of translation base file, containing all strings and their source, it is recommended for monolingual translation formats.

☑ Edit base file
Whether users will be able to edit the base file for monolingual translations.

Intermediate language file:
Filename of intermediate translation file. In most cases this is a translation file provided by developers and is used when creating actual source strings.

Template for new translations:
Filename of file used for creating new translations. For gettext choose .pot file.

File format: Android String Resource

☐ Locked
Locked component will not get any translation updates.

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

☑ Turn on suggestions
Whether to allow translation suggestions at all.

☐ Suggestion voting
Whether users can vote for suggestions.

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

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

Enforced checks: []
List of checks which can not be ignored.

Translation license: MIT License

Contributor agreement:
User agreement which needs to be approved before a user can translate this component.

Adding new translation: Create new language file
How to handle requests for creating new translations.

Language code style: Default based on the file format
Customize language code used to generate the filename for translations created by Weblate.

Merge style: Rebase
Define whether Weblate should merge the upstream repository or rebase changes onto it.

Commit message when translating:
```
Translated using Weblate ({{ language_name }})

Currently translated at {{ stats.translated_percent }}% ({{ stats.translated }} of {{ stats.all }} strings)

Translation: {{ project_name }}/{{ component_name }}
Translate-URL: {{ url }}
```
You can use template language for various info, please consult the documentation for more details.

Commit message when adding translation:
```
Added translation using Weblate ({{ language_name }})
```
You can use template language for various info, please consult the documentation for more details.

Commit message when removing translation:
```
Deleted translation using Weblate ({{ language_name }})
```
You can use template language for various info, please consult the documentation for more details.

Commit message when merging translation:
```
Merge branch '{{ component_remote_branch }}' into Weblate.
```
You can use template language for various info, please consult the documentation for more details.

Commit message when addon makes a change:
```
Update translation files

Updated by "{{ addon_name }}" hook in Weblate.

Translation: {{ project_name }}/{{ component_name }}
Translate-URL: {{ url }}
```
You can use template language for various info, please consult the documentation for more details.

Committer name: Weblate

Committer e-mail: noreply@weblate.org

☑ Push on commit
Whether the repository should be pushed upstream on every commit.

Age of changes to commit: 24
Time in hours after which any pending changes will be committed to the VCS.

☑ Lock on error
Whether the component should be locked on repository errors.

Language filter: ^[^.]+$
Regular expression used to filter translation when scanning for filemask.

Variants regular expression:
Regular expression used to determine variants of a string.

Priority: Medium
Components with higher priority are offered first to translators.

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

Save and add another | Save and continue editing | SAVE

См.также:

Конфигурация компонента, *Двуязычные и одноязычные форматы*

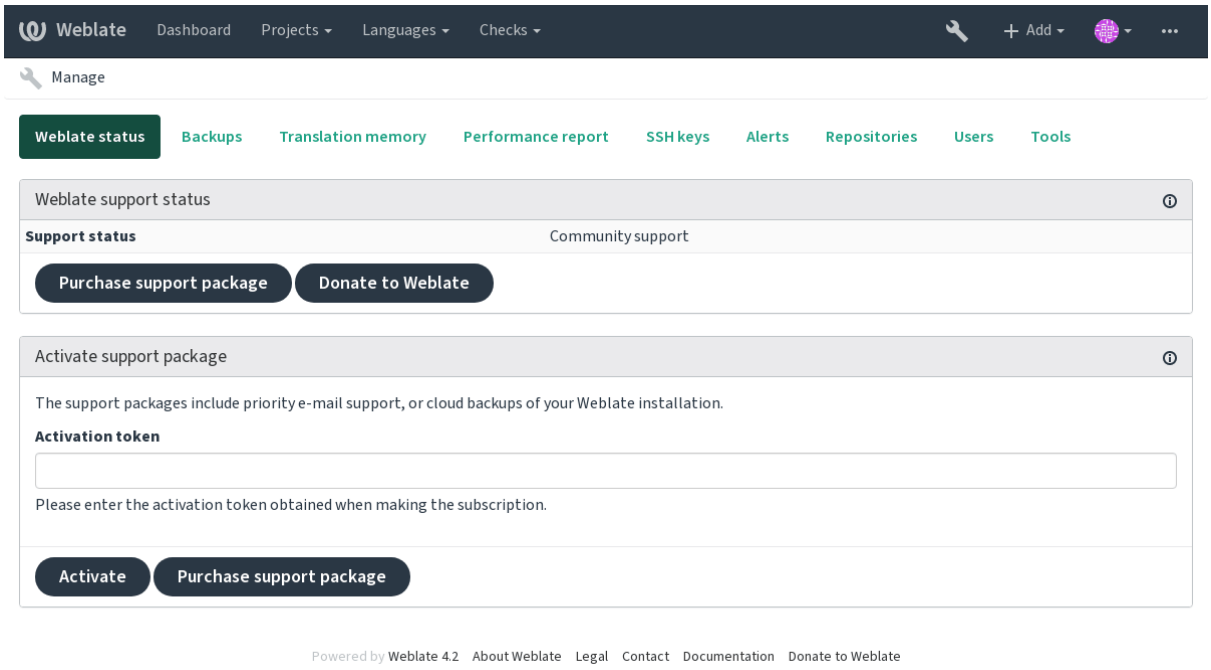
2.24 Getting support for Weblate

Weblate — это свободное программное обеспечение с поддержкой сообщества, выпускаемое на условиях авторского лева. Подписчики получают приоритетную поддержку без дополнительной оплаты. Предоплаченные пакеты помощи доступны для всех. Более подробную информацию о текущих предложениях поддержки вы можете найти по адресу <https://weblate.org/support/>.

2.24.1 Integrating support

Добавлено в версии 3.8.

Purchased support packages can optionally be integrated into your Weblate subscription management interface, from where you will find a link to it. Basic instance details about your installation are also reported back to Weblate this way.



2.24.2 Data submitted to the Weblate

- URL where your Weblate instance is configured
- Your site title
- The Weblate version you are running
- Tallies of some objects in your Weblate database (projects, components, languages, source strings and users)
- The public SSH key of your instance

No other data is submitted.

2.24.3 Integration services

- See if your support package is still valid
- *Использование хранилища резервных копий, предоставляемого Weblate*

Подсказка: Purchased support packages are already activated upon purchase, and can be used without integrating them.

2.25 Правовые документы

Примечание: Herein you will find various legal information you might need to operate Weblate in certain legal jurisdictions. It is provided as a means of guidance, without any warranty of accuracy or correctness. It is ultimately your responsibility to ensure that your use of Weblate complies with all applicable laws and regulations.

2.25.1 ITAR and other export controls

Weblate can be run within your own datacenter or virtual private cloud. As such, it can be used to store ITAR or other export-controlled information, however, end users are responsible for ensuring such compliance.

The Hosted Weblate service has not been audited for compliance with ITAR or other export controls, and does not currently offer the ability to restrict translations access by country.

2.25.2 US encryption controls

Weblate does not contain any cryptographic code, but might be subject export controls as it uses third party components utilizing cryptography for authentication, data-integrity and -confidentiality.

Most likely Weblate would be classified as ECCN 5D002 or 5D992 and, as publicly available libre software, it should not be subject to EAR (see Encryption items NOT Subject to the EAR).

Software components used by Weblate (listing only components related to cryptographic function):

Python Смотрите https://wiki.python.org/moin/PythonSoftwareFoundationLicenseFaq#Is_Python_subject_to_export_laws.3F

GnuPG Optionally used by Weblate

Git Optionally used by Weblate

curl Used by Git

OpenSSL Used by Python and cURL

The strength of encryption keys depend on the configuration of Weblate and the third party components it interacts with, but in any decent setup it will include all export restricted cryptographic functions:

- In excess of 56 bits for a symmetric algorithm
- Factorisation of integers in excess of 512 bits for an asymmetric algorithm
- Computation of discrete logarithms in a multiplicative group of a finite field of size greater than 512 bits for an asymmetric algorithm
- Discrete logarithms in a group different than above in excess of 112 bits for an asymmetric algorithm

Weblate doesn't have any cryptographic activation feature, but it can be configured in a way where no cryptography code would be involved. The cryptographic features include:

- Accessing remote servers using secure protocols (HTTPS)
- Generating signatures for code commits (PGP)

См.также:

Export Controls (EAR) on Open Source Software

ГЛАВА 3

Документация для участников

3.1 Внесение вклада в Weblate

Существуют десятки способов внесения вашего вклада в Weblate. Приветствуется любая помощь, будь то написание кода, проектирование внешнего вида, написание документации или спонсорство:

- *Reporting issues in Weblate*
- *Starting contributing code to Weblate*
- *Перевод Weblate*
- *Финансирование разработки Weblate*

3.1.1 Перевод Weblate

Weblate переводится с помощью самого себя, не стесняйтесь принимать участие в работе по обеспечению доступности Weblate на максимально возможном количестве человеческих языков.

3.1.2 Финансирование разработки Weblate

Вы можете профинансировать дальнейшую разработку Weblate на странице приёма пожертвований. Собранные там средства используются для финансирования безвозмездного хостинга для свободных программных проектов, а также для дальнейшей разработки Weblate. Для получения более подробной информации, например, о целях финансирования и о вознаграждении, которое вы можете получить за то, что являетесь спонсором, пожалуйста, посетите *страницу приёма пожертвований*.

Backers who have funded Weblate

List of Weblate supporters:

- Yashiro Ccs
- Cheng-Chia Tseng
- Timon Reinhard
- Cassidy James
- Loic Dachary
- Marozed
- https://freedombox.org/
- GNU Solidario (GNU Health)

Do you want to be in the list? Please see options on the Donate to Weblate.

3.2 Starting contributing code to Weblate

To understand Weblate source code, please first look into *Weblate source code*, *Weblate frontend* and *Weblate internals*.

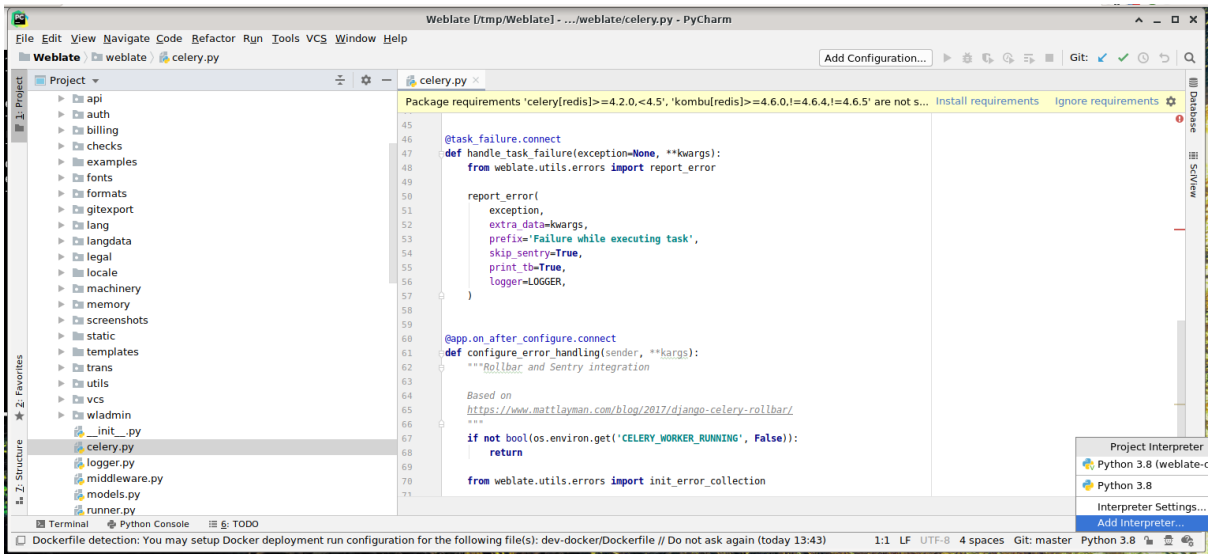
3.2.1 Starting with our codebase

If looking for some bugs to familiarize yourself with the Weblate codebase, look for ones labelled good first issue.

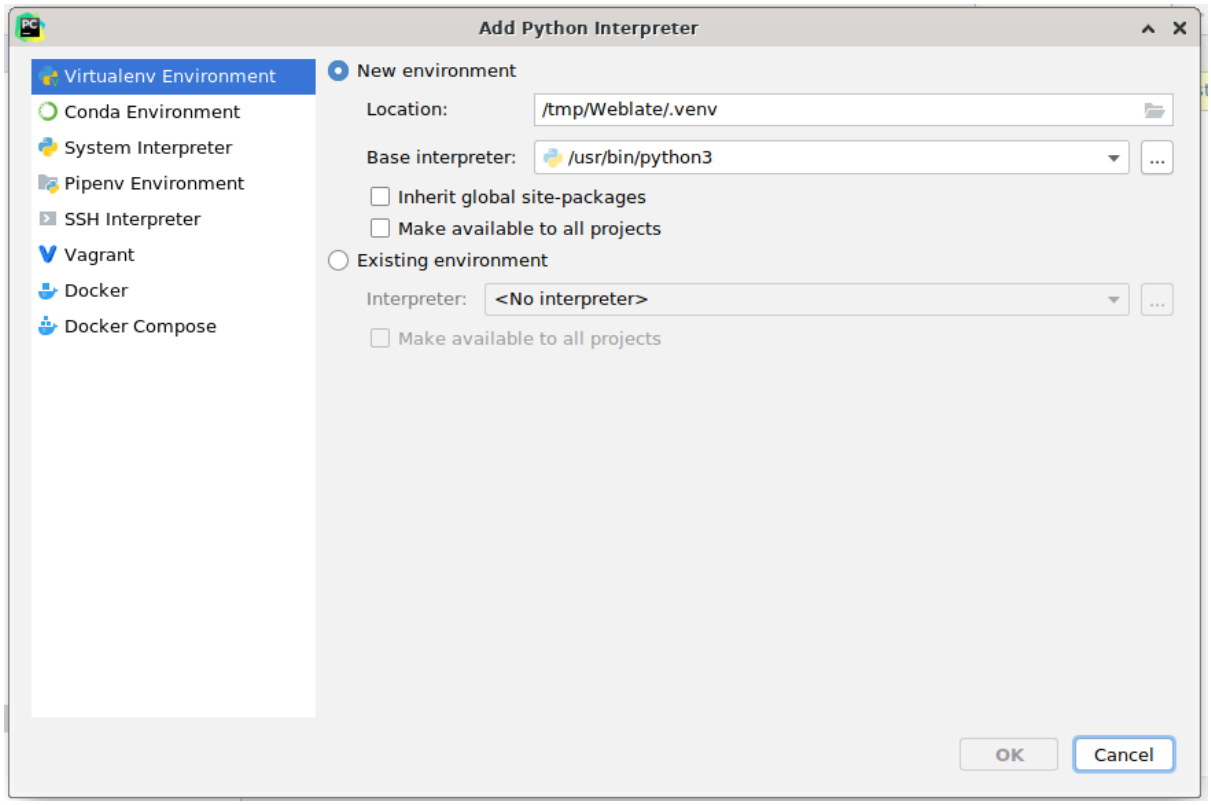
3.2.2 Coding Weblate with PyCharm

PyCharm is a known IDE for Python, here's some guidelines to help you setup Weblate project in it.

Considering you have just cloned the Github repository, just open the folder in which you cloned it in PyCharm. Once the IDE is open, the first step is to specify the interpreter you want:

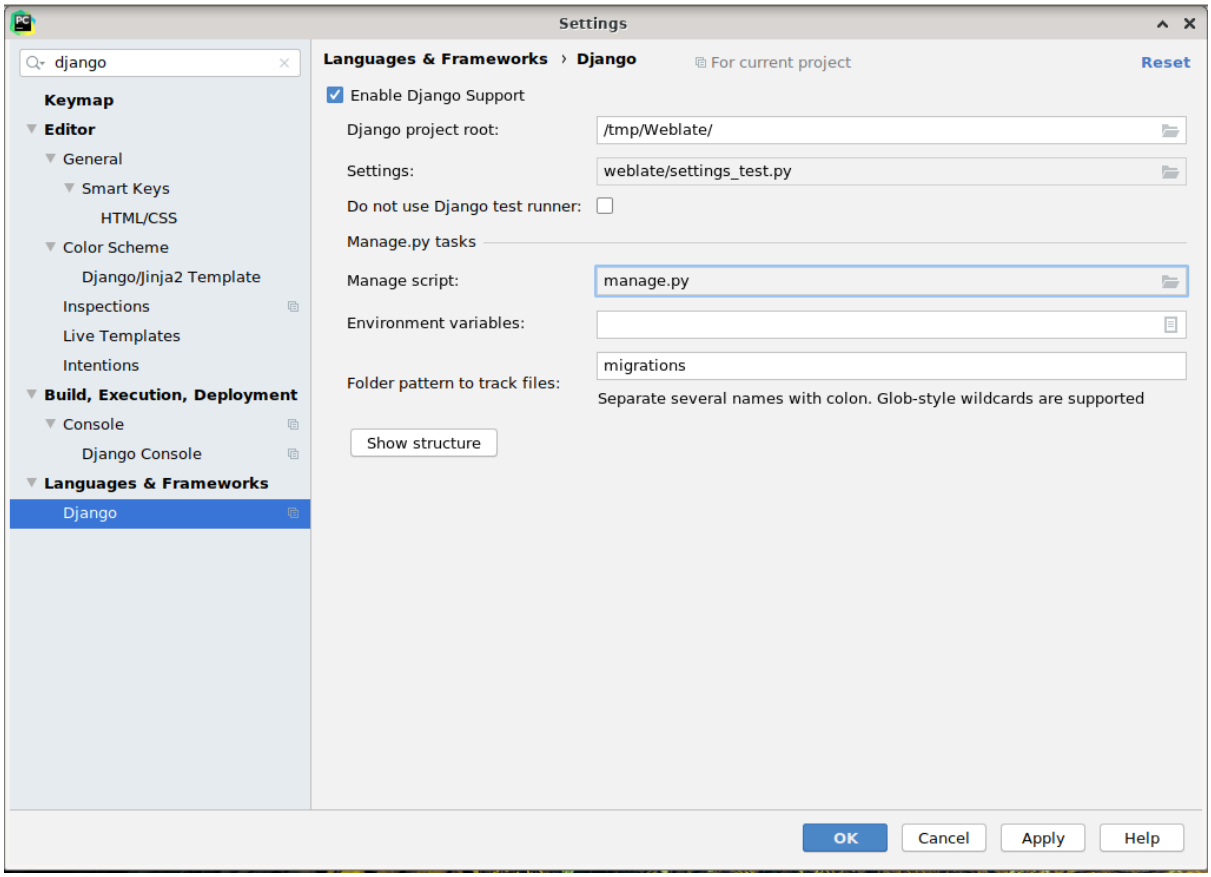


You can either chose to let PyCharm create the virtualenv for you, or select an already existing one:



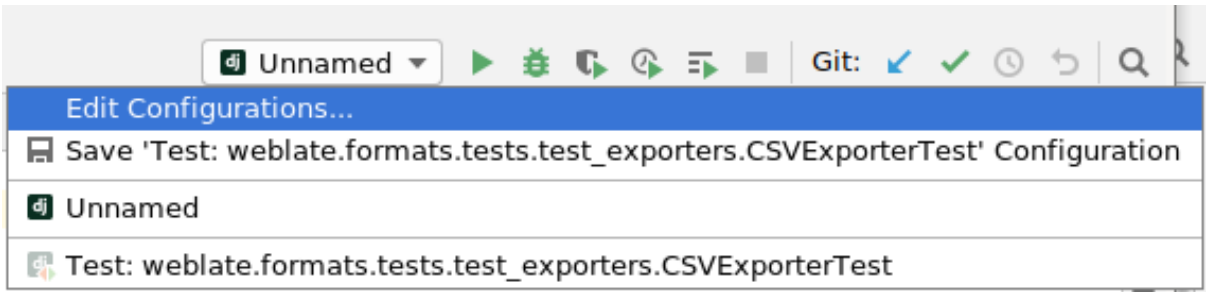
Don't forget to install the dependencies once the interpreter is set: you can do it, either through the console (the console from the IDE will directly use your virtualenv by default), or through the interface when you get a warning about missing dependencies.

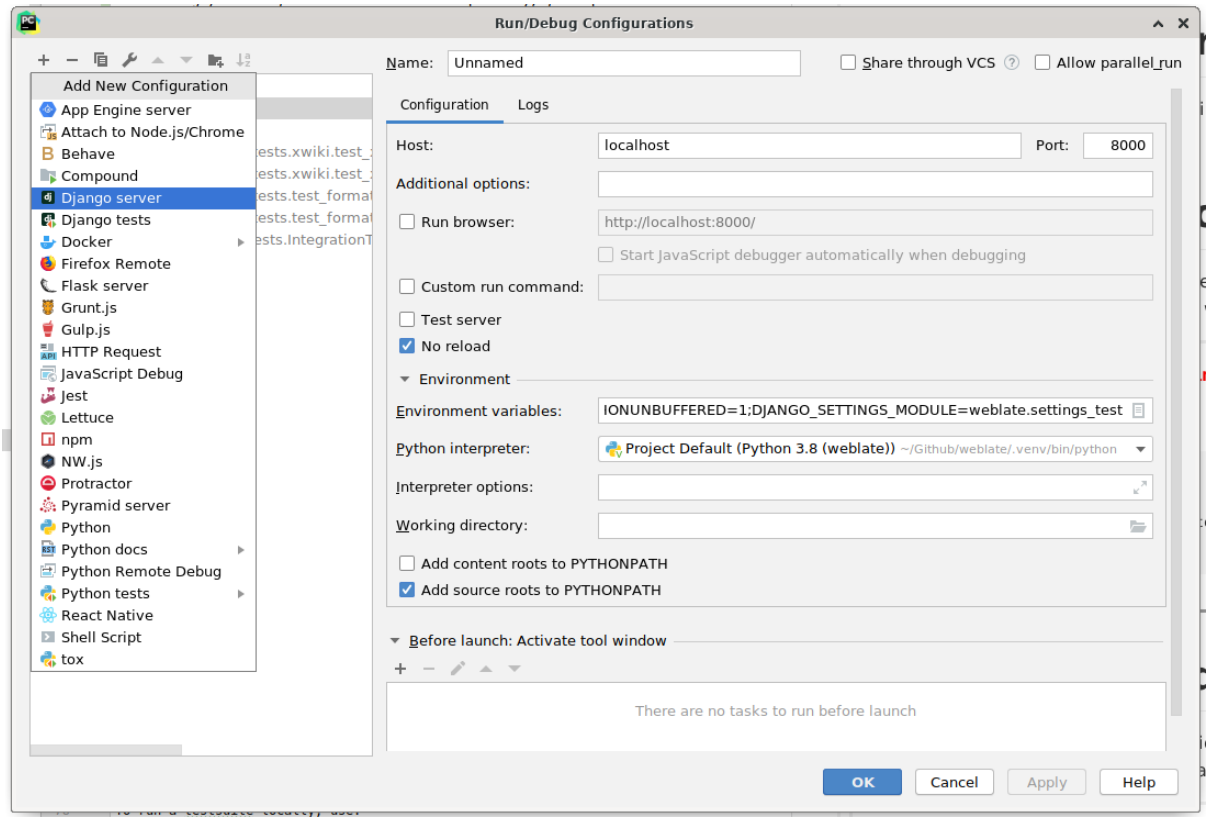
The second step is to set the right information to use natively Django inside PyCharm: the idea is to be able to immediately trigger the unit tests in the IDE. For that you need to specify the root path of Django and the path of one setting:



Be careful, the *Django project root* is the root of the repository, not the weblate sub-directory. About the settings, I personally use the *settings_test* from the repository, but you could create your own setting and set it there.

Last step is to be able to run the server and to put breakpoints on the code to be able to debug it. This is done by creating a new *Django Server* configuration:





Be careful to properly checked «No reload»: you won't get anymore the server live reload if you modify some files, but the debugger will be stopped on the breakpoint you set.

3.2.3 Running Weblate locally

The most comfortable approach to get started with Weblate development is to follow *Установка из исходников*. It will get you a virtual env with editable Weblate sources.

To install all dependencies useful for development, do:

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

To start a development server run:

```
weblate runserver
```

Depending on your configuration you might also want to start Celery workers:

```
./weblate/examples/celery start
```

Running Weblate locally in Docker

If you have Docker and docker-compose installed, you can spin up the development environment simply by running:

```
./rundev.sh
```

It will create development Docker image and start it. Weblate is running on <http://127.0.0.1:8080/> and you can sign in with `admin` user and `admin` password. The new installation is empty, so you might want to continue with *Adding translation projects and components*.

The `Dockerfile` and `docker-compose.yml` for this are located in `dev-docker` directory.

The script also accepts some parameters, to execute tests run it with `test` parameter and then specify any `test` parameters, for example:

```
./rundev.sh test --failfast weblate.trans
```

Be careful that your Docker containers are up and running before running the tests. You can check that by running the `docker ps` command.

To stop the background containers run:

```
./rundev.sh stop
```

Running the script without args will recreate Docker container and restart it.

Примечание: Эта установка не подходит для рабочей среды, она включает в себя несколько небезопасных хаков, которые, однако, упрощают разработку.

3.2.4 Bootstraping your devel instance

You might want to use *`import_demo`* to create demo translations and *`createadmin`* to create admin user.

3.3 Weblate source code

Weblate is developed on GitHub. You are welcome to fork the code and open pull requests. Patches in any other form are welcome too.

См.также:

Check out *Weblate internals* to see how Weblate looks from inside.

3.3.1 Security by Design Principles

Any code for Weblate should be written with Security by Design Principles in mind.

3.3.2 Coding standard

The code should follow PEP-8 coding guidelines and should be formatted using **`black`** code formatter.

To check the code quality, you can use **`flake8`**, the recommended plugins are listed in `.pre-commit-config.yaml` and it's configuration is placed in `setup.cfg`.

The easiest approach to enforce all this is to install pre-commit. Weblate repository contains configuration for it to verify the committed files are sane. After installing it (it is already included in the `requirements-lint.txt`) turn it on by running `pre-commit install` in Weblate checkout. This way all your changes will be automatically checked.

You can also trigger check manually, to check all files run:

```
pre-commit run --all
```

3.4 Debugging Weblate

Bugs can behave as application crashes or as misbehavior. You are welcome to collect info on any such issue and submit it to the issue tracker.

3.4.1 Режим отладки

При включении режима отладки исключения будут отображаться в браузере. Это полезно для отладки проблем в веб-интерфейсе, но для рабочей среды он не подходит, поскольку сильно влияет на производительность и может привести к утечке конфиденциальных данных.

См.также:

Отключение отладочного режима

3.4.2 Журналы Weblate

Weblate can produce detailed logs of what is going in the background. In the default configuration it uses syslog and that makes the log appear either in `/var/log/messages` or `/var/log/syslog` (depending on your syslog daemon configuration).

Docker containers log to their output (as usual in the Docker world), so you can look at the logs using `docker-compose logs`.

См.также:

Пример конфигурации contains `LOGGING` configuration.

3.4.3 Анализ падений программы

In case the application crashes, it is useful to collect as much info about the crash as possible. The easiest way to achieve this is by using third-party services which can collect such info automatically. You can find info on how to set this up in *Сбор отчётов об ошибках*.

3.4.4 Silent failures

Lots of tasks are offloaded to Celery for background processing. Failures are not shown in the user interface, but appear in the Celery logs. Configuring *Сбор отчётов об ошибках* helps you to notice such failures easier.

3.4.5 Performance issues

In case Weblate performs badly in some situation, please collect the relevant logs showing the issue, and anything that might help figuring out where the code might be improved.

In case some requests take too long without any indication, you might want to install *dogslow <https://pypi.org/project/dogslow/>* along with *Сбор отчётов об ошибках* and get pinpointed and detailed tracebacks in the error collection tool.

3.5 Weblate internals

Примечание: This chapter will give you basic overview of Weblate internals.

Weblate derives most of its code structure from, and is based on Django.

3.5.1 Directory structure

Quick overview of directory structure of Weblate main repository:

docs Source code for this documentation, built using Sphinx.

dev-docker Docker code to run development server, see *Running Weblate locally in Docker*.

weblate Source code of Weblate as a Django application, see *Weblate internals*.

weblate/static Client files (CSS, Javascript and images), see *Weblate frontend*.

3.5.2 Модули

Weblate consists of several Django applications (some optional, see *Optional Weblate modules*):

`accounts`

User account, profiles and notifications.

`addons`

Addons to tweak Weblate behavior, see *Надстройки*.

`api`

API based on Django REST framework.

`auth`

Authentication and permissions.

`billing`

The optional *Биллинг* module.

`checks`

Translation string *Проверки качества* module.

`fonts`

Font rendering checks module.

`formats`

File format abstraction layer based on translate-toolkit.

`gitexport`

The optional *Git exporter* module.

`lang`

Module defining language and plural models.

`langdata`

Language data definitions.

`legal`

The optional *Право* module.

`machinery`

Integration of machine translation services.

`memory`

Built in translation memory, see *Память переводов*.

`screenshots`

Screenshots management and OCR module.

`trans`

Main module handling translations.

`utils`

Various helper utilities.

`vcs`

Version control system abstraction.

`wladmin`

Django admin interface customization.

3.6 Weblate frontend

The frontend is currently built using Bootstrap, jQuery and few third party libraries.

3.6.1 Dependency management

The yarn package manager is used to update third party libraries. The configuration lives in `scripts/yarn` and there is a wrapper script `scripts/yarn-update` to upgrade the libraries, build them and copy to correct locations in `weblate/static/vendor`, where all third partly frontend code is located.

3.6.2 Coding style

Weblate relies on Prettier for the code formatting for both JavaScript and CSS files.

We also use ESLint to check the JavaScript code.

3.6.3 Локализация

Should you need any user visible text in the frontend code, it should be localizable. In most cases all you need is to wrap your text inside `gettext` function, but there are more complex features available:

```
document.write(gettext('this is to be translated'));

var object_count = 1 // or 0, or 2, or 3, ...
s = ngettext('literal for the singular case',
        'literal for the plural case', object_count);

fmts = ngettext('There is %s object. Remaining: %s',
        'There are %s objects. Remaining: %s', 11);
s = interpolate(fmts, [11, 20]);
// s is 'There are 11 objects. Remaining: 20'
```

См.также:

Translation topic in the Django documentation

3.6.4 Иконки

Weblate currently uses material design icons, in case you are looking for new one, check <https://materialdesignicons.com/>.

Additionally, there is `scripts/optimize-svg` to reduce size of the SVG as most of the icons are embedded inside the HTML to allow styling of the paths.

3.7 Reporting issues in Weblate

Our issue tracker is hosted at GitHub:

Feel welcome to report any issues with, or suggest improvement of Weblate there. If what you have found is a security issue in Weblate, please consult the «Security issues» section below.

3.7.1 Проблемы безопасности

In order to give the community time to respond and upgrade your are strongly urged to report all security issues privately. HackerOne is used to handle security issues, and can be reported directly at HackerOne.

Alternatively, report to security@weblate.org, which ends up on HackerOne as well.

If you don't want to use HackerOne, for whatever reason, you can send the report by e-mail to michal@cihar.com. You can choose to encrypt it using this PGP key *3CB 1DF1 EF12 CF2A C0EE 5A32 9C27 B313 42B7 511D*.

Примечание: Weblate depends on third party components for many things. In case you find a vulnerability affecting one of those components in general, please report it directly to the respective project.

Some of these are:

- Django
- Django REST framework
- Python Social Auth

3.8 Weblate testsuite and continuous integration

Testsuites exist for most of the current code, increase coverage by adding testcases for any new functionality, and verify that it works.

3.8.1 Continuous integration

Current test results can be found on GitHub Actions and coverage is reported on Codecov.

There are several jobs to verify different aspects:

- Unit tests
- Documentation build and external links
- Migration testing from all supported releases
- Code linting
- Setup verification (ensures that generated dist files do not miss anything and can be tested)

The configuration for the CI is in `.github/workflows` directory. It heavily uses helper scripts stored in `ci` directory. The scripts can be also executed manually, but they require several environment variables, mostly defining Django settings file to use and database connection. The example definition of that is in `scripts/test-database`:

```
# Simple way to configure test database from environment

# Database backend to use postgresql / mysql / mariadb
export CI_DATABASE=postgresql

# Database server configuration
export CI_DB_USER=weblate
export CI_DB_PASSWORD=weblate
export CI_DB_HOST=127.0.0.1

# Django settings module to use
export DJANGO_SETTINGS_MODULE=weblate.settings_test
```

The simple execution can look like:

```
. scripts/test-database
./ci/run-migrate
./ci/run-test
./ci/run-docs
./ci/run-setup
```

3.8.2 Local testing

To run a testsuite locally, use:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py test
```

Подсказка: You will need a database (PostgreSQL) server to be used for tests. By default Django creates separate database to run tests with `test_` prefix, so in case your settings is configured to use `weblate`, the tests will use `test_weblate` database. See *Настройка базы данных для Weblate* for setup instructions.

The `weblate/settings_test.py` is used in CI environment as well (see *Continuous integration*) and can be tuned using environment variables:

```
# Simple way to configure test database from environment

# Database backend to use postgresql / mysql / mariadb
export CI_DATABASE=postgresql

# Database server configuration
```

(continues on next page)

(продолжение с предыдущей страницы)

```
export CI_DB_USER=weblate
export CI_DB_PASSWORD=weblate
export CI_DB_HOST=127.0.0.1

# Django settings module to use
export DJANGO_SETTINGS_MODULE=weblate.settings_test
```

Prior to running tests you should collect static files as some tests rely on them being present:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py collectstatic
```

You can also specify individual tests to run:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py test weblate.gitexport
```

Подсказка: The tests can also be executed inside developer docker container, see *Running Weblate locally in Docker*.

См.также:

See Testing in Django for more info on running and writing tests for Django.

3.9 Data schemas

Weblate uses JSON Schema to define layout of external JSON files.

3.9.1 Weblate Translation Memory Schema

type	*array*		
items			
•	*The Translation Memory Item*		
	type	*object*	
	properties		
	• **category**	*The String Category*	
		1 is global, 2 is shared, 10000000+ are project specific, 20000000+ are user specific	
		type	*integer*
		examples	1
		minimum	0
		default	1
	• **origin**	*The String Origin*	
		Filename or component name	
		type	*string*
		examples	test
		pattern	^(.*)$
		default	
	• **source**	*The Source String*	
		type	*string*
		examples	Hello
		pattern	^(.+)$
		default	
	• **source_language**	*The Source Language*	
		ISO 639-1 / ISO 639-2 / IETF BCP 47	

continues on next page

Таблица 1 – продолжение с предыдущей страницы

		type	*string*
		examples	en
		pattern	^([^]+)$
		default	
	• **target**	*The Target String*	
		type	*string*
		examples	Ahoj
		pattern	^(.+)$
		default	
	• **target_language**	*The Target Language*	
		ISO 639-1 / ISO 639-2 / IETF BCP 47	
		type	*string*
		examples	cs
		pattern	^([^]+)$
		default	
	additionalProperties	False	
definitions			

3.9.2 Weblate user data export

type	*object*		
properties			
• **basic**	*Basic*		
	type	*object*	
	properties		
	• **username**	*Username*	
		type	*string*
		examples	admin
		pattern	^.*$
		default	
	• **full_name**	*Full name*	
		type	*string*
		examples	Weblate Admin
		pattern	^.*$
		default	
	• **email**	*E-mail*	
		type	*string*
		examples	noreply@example.com
		pattern	^.*$
		default	
	• **date_joined**	*Date joined*	
		type	*string*
		examples	2019-11-18T18:53:54.862Z
		pattern	^.*$
		default	
• **profile**	*Profile*		
	type	*object*	
	properties		
	• **language**	*Language*	
		type	*string*
		examples	cs
		pattern	^.*$
		default	
	• **suggested**	*Number of sugested strings*	

continues on next page

Таблица 2 – продолжение с предыдущей страницы

		type	*integer*		
		examples	1		
		default	0		
	• **translated**	*Number of translated strings*			
		type	*integer*		
		examples	24		
		default	0		
	• **uploaded**	*Number of uploaded screenshots*			
		type	*integer*		
		examples	1		
		default	0		
	• **hide_completed**	*Hide completed translations on the dashboard*			
		type	*boolean*		
		examples	False		
		default	True		
	• **secondary_in_zen**	*Show secondary translations in the Zen mode*			
		type	*boolean*		
		examples	True		
		default	True		
	• **hide_source_secondary**	*Hide source if a secondary translation exists*			
		type	*boolean*		
		examples	False		
		default	True		
	• **editor_link**	*Editor link*			
		type	*string*		
		examples			
		pattern	^.*$		
		default			
	• **translate_mode**	*Translation editor mode*			
		type	*integer*		
		examples	0		
		default	0		
	• **zen_mode**	*Zen editor mode*			
		type	*integer*		
		examples	0		
		default	0		
	• **special_chars**	*Special characters*			
		type	*string*		
		examples			
		pattern	^.*$		
		default			
	• **dashboard_view**	*Default dashboard view*			
		type	*integer*		
		examples	1		
		default	0		
	• **dashboard_component_list**	*Default component list*			
		type	None		
		anyOf	•	type	*null*
			•	type	*integer*
	• **languages**	*Translated languages*			
		type	*array*		
		default	[]		

continues on next page

Таблица 2 – продолжение с предыдущей страницы

		items		
		•	*Language code*	
			type	*string*
			examples	cs
			pattern	^.*$
			default	
	• **secondary_languages**	*Secondary languages*		
		type	*array*	
		default	[]	
		items		
		•	*Language code*	
			type	*string*
			examples	sk
			pattern	^.*$
			default	
	• **watched**	*Watched projects*		
		type	*array*	
		default	[]	
		items		
		•	*Project slug*	
			type	*string*
			examples	weblate
			pattern	^.*$
			default	
• **auditlog**	*Audit log*			
	type	*array*		
	default	[]		
	items			
	•	*Items*		
		type	*object*	
		properties		
		• **address**	*IP address*	
			type	*string*
			examples	127.0.0.1
			pattern	^.*$
			default	
		• **user_agent**	*User agent*	
			type	*string*
			examples	PC / Linux / Firefox 70.0
			pattern	^.*$
			default	
		• **timestamp**	*Timestamp*	
			type	*string*
			examples	2019-11-18T18:58:30.845Z
			pattern	^.*$
			default	
		• **activity**	*Activity*	
			type	*string*
			examples	login
			pattern	^.*$
			default	
definitions				

См.также:

Память переводов, `dump_memory`, `import_memory`

3.10 Releasing Weblate

Things to check prior to release:

1. Check newly translated languages by **./scripts/list-translated-languages**.
2. Set final version by **./scripts/prepare-release**.
3. Make sure screenshots are up to date **make -C docs update-screenshots**

Perform the release:

4. Create a release **./scripts/create-release --tag** (see below for requirements)

Post release manual steps:

5. Update Docker image.
6. Close GitHub milestone.
7. Once the Docker image is tested, add a tag and push it.
8. Update Helm chart to new version.
9. Include new version in `.github/workflows/migrations.yml` to cover it in migration testing.
10. Increase version in the repository by **./scripts/set-version**.

To create tags using the **./scripts/create-release** script you will need following:

- GnuPG with private key used to sign the release
- Push access to Weblate git repositories (it pushes tags)
- Configured **hub** tool and access to create releases on the Weblate repo
- SSH access to Weblate download server (the Website downloads are copied there)

3.11 О Weblate

3.11.1 Цели проекта

Web-based continuous localization tool with tight *Интеграция с системой контроля версий* supporting a wide range of *Поддерживаемые форматы файлов*, making it easy for translators to contribute.

3.11.2 Название проекта

«Weblate» is a portmanteau of the words «web» and «translate».

3.11.3 Веб-сайт проекта

The landing page is <https://weblate.org/> and a cloud hosted service at <https://hosted.weblate.org/>. This documentation can be found on <https://docs.weblate.org/>.

3.11.4 Логотипы проекта

The project logos and other graphics is available in <https://github.com/WeblateOrg/graphics/> repository.

3.11.5 Руководство

This project is maintained by Michal Čihař <michal@cihar.com>.

3.11.6 Авторы

Weblate was started by Michal Čihař <michal@cihar.com>. Since its inception in 2012, thousands of people have contributed.

3.12 Лицензия

Copyright (C) 2012 - 2020 Michal Čihař <michal@cihar.com>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

ГЛАВА 4

История изменений

4.1 Weblate 4.2

Released on August 18th 2020.

- Улучшены страницы пользователей и добавлен список пользователей.
- Прекращена поддержка перехода с выпусков серии 3.x, перехода через версии 4.1 или 4.0.
- Added exports into several monolingual formats.
- Улучшены графики деятельности.
- Количество отображаемых соседних строк теперь поддаётся настройке.
- Added support for locking components experiencing repository errors.
- Упрощённая основная навигация (замена кнопок на иконки).
- Improved language code handling in Google Translate integration.
- Дополнение Git для склеивания коммитов теперь генерирует завершители `Co-authored-by:`.
- Улучшен анализатор поисковых запросов.
- Улучшена обратная связь с пользователями при проверке строк формата.
- Улучшена производительность при массовых изменениях состояния.
- Добавлены перенаправления для совместимости после переименования проекта или компонента.
- Добавлены уведомления об одобрении строк, блокировке компонентов и изменении лицензии.
- Добавлена поддержка ModernMT.
- Разрешено избегать перезаписи одобренных переводов при загрузке файлов.
- Прекращена поддержка некоторых перенаправлений URL-адресов, существовавших ради совместимости.
- Добавлена проверка для шаблонных литералов ECMAScript.
- Добавлен параметр для отслеживания компонента.
- Из ключей единиц перевода JSON удалена ведущая точка.
- Удалена отдельная очередь Celery для памяти переводов.

- Allow translating all components a language at once.
- Разрешено настраивать HTTP -заголовки `Content-Security-Policy`.
- Добавлена поддержка задания псевдонимов для языков на уровне проекта.
- Новая надстройка для помощи в локализации HTML или JavaScript, смотрите раздел *CDN локализации JavaScript'а*.
- The Weblate domain is now configured in the settings, see `SITE_DOMAIN`.
- Add support for searching by component and project.

4.2 Weblate 4.1.1

Выпущен 19 июня 2020 года.

- Исправлено изменение конфигурации автоматического исправления или надстроек в Docker.
- Fixed possible crash in «About» page.
- Улучшена установка скомпилированных файлов локали.
- Исправлено добавление слов в глоссарий.
- Исправлено сочетание клавиш для автоматизированного перевода.
- Removed debugging output causing discarding log events in some setups.
- Исправлена индикация блокировки в списке проектов.
- Fixed listing GPG keys in some setups.
- Added option for which DeepL API version to use.
- Добавлена поддержка работы в качестве провайдера SAML-сервиса, смотрите *Аутентификация через SAML*.

4.3 Weblate 4.1

Выпущен 15 июня 2020 года.

- Добавлена поддержка создания новых переводов с включённым кодом страны.
- Добавлена поддержка поиска исходных строк со снимками экрана.
- Extended info available in the stats insights.
- Improved search editing on «Translate» pages.
- Улучшена обработка одновременных обновлений репозитория.
- В форму создания проекта включён исходный язык.
- В список переводчиков включено количество изменений.
- Исправлена возникающая в некоторых случаях ошибка выбора языка пользовательского интерфейса.
- Разрешение внесения методов регистрации в белый список при закрытых регистрациях.
- Улучшен поиск смежных терминов в глоссарии.
- Улучшены соответствия с памятью переводов.
- Группировка одинаковых результатов автоматизированного перевода.
- Добавление прямой ссылки для редактирования снимка экрана прямо со страницы перевода.
- Улучшен диалог подтверждения удаления.

- В скачиваемый ZIP-файл включены шаблоны.
- В объявлениях добавлена поддержка Markdown и настройка уведомлений.
- Расширенная информация в списках проверок.
- Добавлена поддержка новых форматов файлов: *Строки Laravel PHP*, *Файлы HTML*, *Формат OpenDocument*, *Формат IDML*, *Файлы ресурсов Windows*, *Переводы в INI-файлах*, *Переводы в INI-файлах Inno Setup*, *Файлы свойств GWT*, *JSON -файлы go-i18n* и *Файл ARB*.
- Последовательное использование статуса «пропущена» в качестве статуса пропущенных проверок.
- Добавлена поддержка для настройки включённых надстроек по умолчанию.
- Исправлено сочетание клавиш редактора для пропуска проверок.
- Улучшен машинный перевод строк с заполнителями.
- Показ призрачного перевода для языков пользователя, чтобы упростить ему начало перевода.
- Улучшен разбор кода языка.
- Показ переводов на язык пользователя на первом месте в списке.
- Переименование «форм» на более обобщённые «варианты».
- Добавлены новые проверки качества: *Несколько переменных без названий*, *Давно не переводилась* и *Повторяющиеся слова*.
- Вновь введена поддержка стирания памяти переводов.
- Исправлен параметр для включения в исходной строке игнорирования проверок.
- Добавлена поддержка настройки различных веток, в которые следует отправлять изменения.
- API теперь сообщает об ограничении в частоте запросов в заголовках HTTP.
- Добавлена поддержка Google Translate V3 API (расширенная).
- Добавлена возможность ограничения доступа на уровне компонентов.
- Во флагах перевода добавлена поддержка пробельных и других специальных символов, смотрите раздел *Настройка поведения*.
- Отображаемый текст, если он включён, показывается всегда.
- API теперь поддерживает фильтрацию изменений.
- Добавлена поддержка обмена глоссариями между проектами.

4.4 Weblate 4.0.4

Выпущен 7 мая 2020 года.

- Исправлено выполнение тестового набора в некоторых окружениях с Python'ом 3.8.
- Исправление опечаток в документации.
- Исправлена возникавшая в некоторых случаях ошибка создания компонентов с помощью API.
- Исправлены ошибки JavaScript, ломающие навигацию на мобильных устройствах.
- Исправлено падение при отображении некоторых проверок.
- Исправлен вывод списка снимков экрана.
- Исправлены ежемесячные выдержки с уведомлениями.
- Исправлено поведение промежуточного перевода с единицами перевода, не существовавшими в переводе.

4.5 Weblate 4.0.3

Выпущен 2 мая 2020 года.

- Исправлено возможное падение в отчётах.
- Упоминания пользователей в комментариях теперь не чувствительны к регистру.
- Исправлена миграция PostgreSQL для не суперпользователей.
- Исправлено изменение URL-адреса репозитория при создании компонента.
- Исправлено падение при исчезновении вышестоящего репозитория.

4.6 Weblate 4.0.2

Выпущен 27 апреля 2020 года.

- Улучшена производительность статистики перевода.
- Улучшена производительность смены меток.
- Улучшена производительность массовой правки.
- Улучшена производительность памяти переводов.
- Исправлено возможное падение при удалении компонента.
- Исправлено отображение изменений перевода в некоторых пограничных случаях.
- Улучшено предупреждение о слишком длинной очереди Celery.
- Исправлены возможные ложные срабатывания при проверке согласованности.
- Исправлена взаимоблокировка при изменении репозитория связанного компонента.
- В список изменений, CSV и отчёты включено расстояние редактирования.
- Обход ложных срабатываний проверки интервала между знаками препинания для канадского французского языка.
- Исправлен экспорт в формат XLIFF с заполнителями.
- Исправлено ложное срабатывание с проверкой на символ нулевой ширины.
- Улучшены отчёты об ошибках конфигурации.
- Исправлена загрузка двуязычных исходных текстов.
- Автоматическое определение поддерживаемых языков для машинного перевода DeepL.
- Исправлено отображение индикатора выполнения в некоторых пограничных случаях.
- Исправлено срабатывание некоторых проверок на непереведённых строках.

4.7 Weblate 4.0.1

Выпущен 16 апреля 2020 года.

- Исправлена установка пакета из PyPI.

4.8 Weblate 4.0

Выпущен 16 апреля 2020 года.

- Weblate теперь требует Python 3.6 или новее.
- Добавлен обзор управления предупреждений компонентов.
- Добавлено предупреждение компонента о сломанных URL-адресах браузера репозитория.
- Улучшены страницы входа и регистрации.
- Управление доступом к проекту и конфигурация рабочего процесса интегрированы в настройки проекта.
- Добавлена проверка и подсветка для интерполяции и вложенных строк i18next.
- Добавлена проверка и подсветка для процентных заполнителей.
- Отображение неудачных проверок в предложениях.
- Запись изменений исходной строки в историю.
- Обновление переводчика Microsoft Translator на третью версию его API.
- Повторная реализация серверной части памяти переводов.
- В *Поиск* добавлена поддержка нескольких `is:` поисков.
- Разрешено делать так, чтобы проверка *Перевод без изменения* не попадала во внутренний чёрный список.
- Улучшено извлечение комментариев из одноязычных po файлов.
- «Доски сообщений» переименованы в «объявления».
- Исправлены случайные проблемы с регистрационными письмами.
- Улучшена надстройка обновления LINGUAS-файла: теперь она обрабатывает большее количество синтаксических вариантов.
- Исправлено редактирование одноязычного исходного файла XLIFF.
- В *Поиск* добавлена поддержка точного соответствия.
- Расширенный API, охватывающий снимки экрана, пользователей, группы, списки компонентов и расширенное создание проектов.
- Добавлена поддержка загрузки исходных текстов в двуязычные переводы.
- Добавлена поддержка промежуточного языка от разработчиков.
- Добавлена поддержка рецензирования исходных строк.
- Расширенные параметры скачивания памяти переводов со всей платформы.

4.9 Серия Weblate 3.x

4.9.1 Weblate 3.11.3

Выпущен 11 марта 2020 года.

- Исправлен поиск полей с определённым приоритетом.
- Исправлен предопределённый запрос для недавно добавленных строк.
- Исправлен поиск, возвращающий повторяющиеся совпадения.
- Исправлена отрисовка уведомлений в Gmail.

- Исправлен возврат изменений из истории.
- Добавлены ссылки на события в выдержках с уведомлениями.
- Исправлено электронное письмо для подтверждения удаления учётной записи.
- Добавлена поддержка аутентификации через Slack в контейнере Docker.
- Устранение отправки уведомлений для языков, на которые пользователь не подписан.
- Включение в обзор производительности очередей Celery.
- Исправлены ссылки на документацию для надстроек.
- Уменьшение количества ложно-отрицательных результатов для проверки «Перевод без изменения».
- Повышение версии зависимости bleach для закрытия уязвимости CVE-2020-6802.
- Исправлено перечисление изменений в истории на уровне проекта.
- Исправлено аннулирование статистики в некоторых граничных случаях.
- Исправлен поиск определённых состояний строки.
- Улучшено поведение проверки строк формата при пропущенном проценте.
- Исправлена аутентификация с использованием некоторых сторонних провайдеров.

4.9.2 Weblate 3.11.2

Выпущен 22 февраля 2020 года.

- Исправлена отрисовка предложений.
- Исправлены некоторые строки, для которых ошибочно сообщалось, что в нет слов.

4.9.3 Weblate 3.11.1

Выпущен 20 февраля 2020 года.

- Документированы изменения в настройке Celery.
- Улучшена проверка имени файла при создании компонентов.
- Исправлены минимальные версии некоторых зависимостей.
- Исправлено добавление групп в определённых версиях Django.
- Исправлена ручная отправка изменений в вышестоящий репозиторий.
- Улучшен поиск соответствия в словаре.

4.9.4 Weblate 3.11

Выпущен 17 февраля 2020 года.

- Разрешено использование URL-адреса для отправки в систему контроля версий во время создания компонента через API.
- Проверка ширины отрисованного текста теперь показывает изображение с отрисованным текстом.
- Исправлены ссылки в электронных письмах с уведомлениями.
- Улучшен внешний вид простых текстовых электронных писем.
- Отображение проигнорированных проверок и разрешение сделать их снова активными.
- Отображение соседних ключей в одноязычных переводах.

- Добавлена поддержка группировки строковых форм.
- Рекомендация обновления до новой версии Weblate в проверках системы.
- Предоставление более подробного анализа для предупреждения о повторяющемся языке.
- Включение более подробной информации о лицензиях на страницах проекта.
- Автоматическое преобразование поверхностных локальных копий в полные, если это необходимо.
- Исправлено скачивание строк, требующих действий.
- Новое предупреждение о повторном использовании одной и той же маски файлов.
- Улучшено извлечение мест подстановки в XML.
- Теперь параметр `SINGLE_PROJECT` может включать принудительное перенаправление на выбранный проект.
- Добавлена возможность указания, что описанная в комментарии проблема решена.
- Добавлена массовая правка флагов.
- Добавлена поддержка *меток*.
- Добавлена надстройка «Массовая правка».
- Добавлен параметр для *Enforcing checks*.
- Увеличен умолчательный срок действия ссылок подтверждения.
- Улучшена интеграция с Matomo.
- Исправлена проверка *Уже переведено* для корректной обработки изменений в исходных строках.
- Расширение настроек автоматического обновления, контролируемых параметром `AUTO_UPDATE`.
- Надстройки LINGUAS теперь выполняют полную синхронизацию переводов с Weblate.

4.9.5 Weblate 3.10.3

Выпущен 18 января 2020 года.

- Поддержка translate-toolkit 2.5.0.

4.9.6 Weblate 3.10.2

Выпущен 18 января 2020 года.

- К проектам добавлена индикация блокировки.
- Исправлена ошибка в CSS, вызывавшая мерцание в некоторых веб-браузерах.
- Исправлен поиск в системах с неанглийскими локалями.
- Улучшено сопоставление репозитория для обработчиков GitHub и Bitbucket.
- Исправлена миграция данных на некоторых установках Python 2.7.
- Разрешена настройка поверхностного клонирования в Git'е.
- Улучшена обработка фоновых уведомлений.
- Исправлена ошибка отправки сломанных форм при использовании браузерной кнопки «назад».
- Новая надстройка для настройки форматирования YAML.
- Исправлена проверка «Одинаковые формы множественного числа»: теперь она не срабатывает в языках с единственной формой множественного числа.
- Исправлен поиск с использованием регулярных выражений по некоторым полям.

4.9.7 Weblate 3.10.1

Выпущен 9 января 2020 года.

- Расширенный API с созданием перевода.
- Исправлено несколько граничных случаев при миграции данных.
- Совместимость с Django 3.0.
- Улучшена производительность очистки данных.
- Добавлена поддержка настройки security.txt.
- Улучшены хлебные крошки в журнале изменений.
- Улучшен список переводов в панели управления.
- Улучшены HTTP-ответы для веб-обработчиков.
- Добавлена поддержка запросов на слияние GitLab в контейнере Docker.

4.9.8 Weblate 3.10

Выпущен 20 декабря 2019 года.

- Улучшен пользовательский интерфейс приложения.
- Добавлена проверка на двойной пробел.
- Исправлено создание новых языков.
- Избежание отправки уведомлений из журнала аудита на удалённые адреса электронной почты.
- Добавлена поддержка строк только для чтения.
- Добавлена поддержка Markdown'а в комментариях.
- Разрешение размещения текста инструкции по переводу в информации о проекте.
- Добавлено копирование в буфер обмена для дополнительных языков.
- Улучшена поддержка Mercurial.
- Улучшена производительность извлечения из Git-репозитория.
- Добавлен поиск по возрасту строки.
- Показ исходного языка для всех переводов.
- Показ контекста для соседних строк.
- Добавлена поддержка уведомлений об операциях с репозиторием.
- Улучшены списки перевода.
- Расширены возможности поиска.
- Добавлена поддержка автоматического перевода строк, помеченных на правку.
- Избежание отправки повторных уведомлений для предупреждений в связанных компонентах.
- Улучшение умолчательного сообщения о запросе на слияние.
- Улучшение индикации состояния строки в Дзен-режиме.
- Добавлена поддержка большего количества языков в Яндекс.Переводчике.
- Улучшен внешний вид электронных писем уведомлений.
- Предоставлена возможность выбора лицензии на перевод.

4.9.9 Weblate 3.9.1

Выпущен 28 октября 2019 года.

- Из резервных копий удалены кое-какие ненужные файлы.
- Исправлено возможное падение в отчётах.
- Исправлена падение при миграции из одной базы данных в другую.
- Добавлена поддержка принудительной отправки изменений в Git-репозитории.
- Снижение риска аннулирования токена регистрации.
- Исправлено удаление учётной записи, упёршейся в ограничение на частоту запросов.
- Добавлен поиск по приоритету.
- Исправлено возможное падение при добавлении строк в JSON-файл.
- Проверка и исправление небезопасного HTML теперь учитывает разметку исходной строки.
- Избегание отправки уведомлений приглашённым, но в то же время удалённым пользователям.
- Исправлено SSL-соединение с redis в Celery, находящимся в контейнере Docker'а.

4.9.10 Weblate 3.9

Выпущен 15 октября 2019 года.

- Включение в скачиваемые файлы метаданных Weblate.
- Улучшен пользовательский интерфейс для неудачных проверок.
- Указание недостающих строк в проверках формата.
- Отдельная проверка на пробелы между знаками препинания французского языка.
- Добавлена поддержка исправления некоторых ошибок проверки качества.
- Добавлено отдельное разрешение на создание новых проектов.
- Расширена статистика по количеству символов.
- Улучшение поддержки языковых кодов в стиле Java.
- Добавлена новая обобщённая проверка на заполнители.
- Добавлена поддержка заполнителей WebExtension JSON.
- Добавлена поддержка формата плоского XML-файла.
- API расширен методами удаления и создания проекта, компонента и перевода.
- Добавлена поддержка веб-обработчиков Gitea и Gitee.
- Добавлена новая пользовательская проверка на основе регулярных выражений.
- Разрешение настройки внесения вклада в общую память переводов.
- Скачивание ZIP-архива добавлено для большего количества файлов перевода.
- Разбор атрибутов maxwidth и font в формате XLIFF сделан совместимым со стандартом.
- Добавлена новая проверка и исправление для безопасной HTML-разметки при переводе веб-приложений.
- В компоненте добавлено предупреждение при обнаружении неподдерживаемой конфигурации.
- Добавлена надстройка автоматического перевода для начального перевода.
- Автоматический перевод расширен для добавления предложений.
- Отображение в обзоре параметров надстройки.

- Sentry теперь поддерживается через современный Sentry SDK вместо Raven.
- Изменены примеры настроек, чтобы они лучше соответствовали окружению рабочей среды.
- Добавлено автоматическое резервное копирование с помощью BorgBackup.
- Разбиение надстройки очистки для RESX на две надстройки очистки и обновления, чтобы избежать нежелательных обновлений файлов.
- Добавлены расширенные возможности поиска.
- Пользователям разрешено скачивать свои собственные отчёты.
- Для помощи в настройке компонентов добавлено руководство по локализации.
- Добавлена поддержка запросов на слияние GitLab.
- Улучшено отображение статуса репозитория.
- Выполнение автоматического перевода в фоновом режиме.

4.9.11 Weblate 3.8

Выпущен 15 августа 2019 года.

- Добавлена поддержка упрощённого создания похожих компонентов.
- Добавлена поддержка разбора флагов перевода из форматов файлов, основанных на XML.
- Журналирование исключений в журнал Celery.
- Улучшена производительность надстроек, работающих с репозиторием.
- Улучшен внешний вид электронных писем уведомлений.
- Исправлено поведение при сбросе пароля.
- Улучшена производительность на большинстве страниц перевода.
- Исправлен список языков, неизвестных Weblate'у.
- Add support for cloning addons to discovered components.
- Add support for replacing file content with uploaded.
- Add support for translating non VCS based content.
- Added OpenGraph widget image to use on social networks.
- Added support for animated screenshots.
- Improved handling of monolingual XLIFF files.
- Avoid sending multiple notifications for single event.
- Add support for filtering changes.
- Extended predefined periods for reporting.
- Added webhook support for Azure Repos.
- New opt-in notifications on pending suggestions or untranslated strings.
- Add one click unsubscribe link to notification e-mails.
- Fixed false positives with Has been translated check.
- New management interface for admins.
- String priority can now be specified using flags.
- Added language management views.
- Add checks for Qt library and Ruby format strings.

- Added configuration to better fit single project installations.
- Notify about new string on source string change on monolingual translations.
- Added separate view for translation memory with search capability.

4.9.12 Weblate 3.7.1

Выпущен 28 июня 2019 года.

- Обновление документации.
- Fixed some requirements constraints.
- Updated language database.
- Обновления локализации.
- Various user interface tweaks.
- Improved handling of unsupported but discovered translation files.
- More verbosely report missing file format requirements.

4.9.13 Weblate 3.7

Выпущен 21 июня 2019 года.

- Added separate Celery queue for notifications.
- Use consistent look with application for API browsing.
- Include approved stats in the reports.
- Report progress when updating translation component.
- Allow to abort running background component update.
- Extend template language for filename manipulations.
- Use templates for editor link and repository browser URL.
- Indicate max length and current characters count when editing translation.
- Improved handling of abbreviations in unchanged translation check.
- Refreshed landing page for new contributors.
- Add support for configuring msgmerge addon.
- Delay opening SMTP connection when sending notifications.
- Improved error logging.
- Allow custom location in MO generating addon.
- Added addons to cleanup old suggestions or comments.
- Added option to enable horizontal mode in the Zen editor.
- Improved import performance with many linked components.
- Fixed examples installation in some cases.
- Improved rendering of alerts in changes.
- Added new horizontal stats widget.
- Improved format strings check on plurals.
- Added font management tool.

- New check for rendered text dimensions.
- Added support for subtitle formats.
- Include overall completion stats for languages.
- Added reporting at project and global scope.
- Improved user interface when showing translation status.
- New Weblate logo and color scheme.
- New look of bitmap badges.

4.9.14 Weblate 3.6.1

Выпущен 26 апреля 2019 года.

- Improved handling of monolingual XLIFF files.
- Исправлены выдержки с уведомлениями в некоторых граничных случаях.
- Fixed addon script error alert.
- Fixed generating MO file for monolingual PO files.
- Fixed display of uninstalled checks.
- Indicate administered projects on project listing.
- Allow update to recover from missing VCS repository.

4.9.15 Weblate 3.6

Выпущен 20 апреля 2019 года.

- Add support for downloading user data.
- Addons are now automatically triggered upon installation.
- Improved instructions for resolving merge conflicts.
- Cleanup addon is now compatible with app store metadata translations.
- Configurable language code syntax when adding new translations.
- Warn about using Python 2 with planned termination of support in April 2020.
- Extract special characters from the source string for visual keyboard.
- Extended contributor stats to reflect both source and target counts.
- Admins and consistency addons can now add translations even if disabled for users.
- Fixed description of toggle disabling `Language-Team` header manipulation.
- Notify users mentioned in comments.
- Removed file format autodetection from component setup.
- Fixed generating MO file for monolingual PO files.
- Добавлены выдержки с уведомлениями.
- Added support for muting component notifications.
- Added notifications for new alerts, whiteboard messages or components.
- Notifications for administered projects can now be configured.
- Improved handling of three letter language codes.

4.9.16 Weblate 3.5.1

Выпущен 10 марта 2019 года.

- Fixed Celery systemd unit example.
- Fixed notifications from HTTP repositories with login.
- Fixed race condition in editing source string for monolingual translations.
- Include output of failed addon execution in the logs.
- Improved validation of choices for adding new language.
- Allow to edit file format in component settings.
- Update installation instructions to prefer Python 3.
- Performance and consistency improvements for loading translations.
- Терминологическая служба Майкрософт сделана совместимой с текущими выпусками Zeep.
- Обновления локализации.

4.9.17 Weblate 3.5

Выпущен 3 марта 2019 года.

- Improved performance of built-in translation memory.
- Added interface to manage global translation memory.
- Improved alerting on bad component state.
- Added user interface to manage whiteboard messages.
- Addon commit message now can be configured.
- Reduce number of commits when updating upstream repository.
- Fixed possible metadata loss when moving component between projects.
- Improved navigation in the Zen mode.
- Added several new quality checks (Markdown related and URL).
- Added support for app store metadata files.
- Added support for toggling GitHub or Gerrit integration.
- Добавлена проверка на буквы кашиды.
- Добавлен параметр для склейки коммитов на основе их авторства.
- Улучшена поддержка формата файла XLSX.
- Совместимость с Tesseract 4.0.
- Billing addon now removes projects for unpaid billings after 45 days.

4.9.18 Weblate 3.4

Выпущен 22 января 2019 года.

- Added support for XLIFF placeholders.
- Celery can now utilize multiple task queues.
- Added support for renaming and moving projects and components.
- В отчёты включено количество символов.
- Added guided adding of translation components with automatic detection of translation files.
- Customizable merge commit messages for Git.
- Added visual indication of component alerts in navigation.
- Улучшена производительность загрузки файлов перевода.
- Новая надстройка для склеивания коммитов до их отправки.
- Улучшено отображение изменений перевода.
- Changed default merge style to rebase and made that configurable.
- Better handle private use subtags in language code.
- Improved performance of fulltext index updates.
- Extended file upload API to support more parameters.

4.9.19 Weblate 3.3

Выпущен 30 ноября 2018 года.

- Добавлена поддержка удаления компонентов и проектов.
- Improved performance for some monolingual translations.
- Added translation component alerts to highlight problems with a translation.
- Expose XLIFF string resname as context when available.
- Added support for XLIFF states.
- Added check for non writable files in DATA_DIR.
- Improved CSV export for changes.

4.9.20 Weblate 3.2.2

Выпущен 20 октября 2018 года.

- Remove no longer needed Babel dependency.
- Updated language definitions.
- Improve documentation for addons, LDAP and Celery.
- Fixed enabling new dos-eol and auto-java-messageformat flags.
- Fixed running setup.py test from PyPI package.
- Improved plurals handling.
- Fixed translation upload API failure in some corner cases.
- Fixed updating Git configuration in case it was changed manually.

4.9.21 Weblate 3.2.1

Выпущен 10 октября 2018 года.

- Document dependency on backports.csv on Python 2.7.
- Fix running tests under root.
- Improved error handling in gitexport module.
- Fixed progress reporting for newly added languages.
- Correctly report Celery worker errors to Sentry.
- Fixed creating new translations with Qt Linguist.
- Fixed occasional fulltext index update failures.
- Improved validation when creating new components.
- Added support for cleanup of old suggestions.

4.9.22 Weblate 3.2

Выпущен 6 октября 2018 года.

- Добавлена команда управления install_addon для автоматизации установки надстроек.
- Allow more fine grained ratelimit settings.
- Added support for export and import of Excel files.
- Improve component cleanup in case of multiple component discovery addons.
- Rewritten Microsoft Terminology machine translation backend.
- Weblate now uses Celery to offload some processing.
- Improved search capabilities and added regular expression search.
- Added support for Youdao Zhiyun API machine translation.
- Added support for Baidu API machine translation.
- Integrated maintenance and cleanup tasks using Celery.
- Improved performance of loading translations by almost 25%.
- Removed support for merging headers on upload.
- Removed support for custom commit messages.
- Configurable editing mode (zen/full).
- Added support for error reporting to Sentry.
- Added support for automated daily update of repositories.
- Added support for creating projects and components by users.
- Built in translation memory now automatically stores translations done.
- Users and projects can import their existing translation memories.
- Better management of related strings for screenshots.
- Added support for checking Java MessageFormat.

See 3.2 milestone on GitHub for detailed list of addressed issues.

4.9.23 Weblate 3.1.1

Выпущен 27 июля 2018 года.

- Fix testsuite failure on some setups.

4.9.24 Weblate 3.1

Выпущен 27 июля 2018 года.

- Upgrades from older version than 3.0.1 are not supported.
- Allow to override default commit messages from settings.
- Improve webhooks compatibility with self hosted environments.
- Added support for Amazon Translate.
- Compatibility with Django 2.1.
- Django system checks are now used to diagnose problems with installation.
- Removed support for soon shutdown libravatar service.
- New addon to mark unchanged translations as needing edit.
- Add support for jumping to specific location while translating.
- Downloaded translations can now be customized.
- Improved calculation of string similarity in translation memory matches.
- Added support by signing Git commits by GnuPG.

4.9.25 Weblate 3.0.1

Выпущен 10 июня 2018 года.

- Fixed possible migration issue from 2.20.
- Обновления локализации.
- Removed obsolete hook examples.
- Improved caching documentation.
- Fixed displaying of admin documentation.
- Improved handling of long language names.

4.9.26 Weblate 3.0

Выпущен 1 июня 2018 года.

- Rewritten access control.
- Several code cleanups that lead to moved and renamed modules.
- New addon for automatic component discovery.
- Команда управления import_project теперь принимает несколько иные параметры.
- Добавлена базовая поддержка файлов ресурсов Windows.
- New addon to store contributor names in PO file headers.
- The per component hook scripts are removed, use addons instead.
- Add support for collecting contributor agreements.

- Access control changes are now tracked in history.
- New addon to ensure all components in a project have same translations.
- Support for more variables in commit message templates.
- Add support for providing additional textual context.

4.10 Серия Weblate 2.x

4.10.1 Weblate 2.20

Выпущен 4 апреля 2018 года.

- Improved speed of cloning subversion repositories.
- Changed repository locking to use third party library.
- Added support for downloading only strings needing action.
- Added support for searching in several languages at once.
- New addon to configure gettext output wrapping.
- New addon to configure JSON formatting.
- Added support for authentication in API using RFC 6750 compatible Bearer authentication.
- Added support for automatic translation using machine translation services.
- Added support for HTML markup in whiteboard messages.
- Added support for mass changing state of strings.
- Translate-toolkit at least 2.3.0 is now required, older versions are no longer supported.
- Added built in translation memory.
- Added componentlists overview to dashboard and per component list overview pages.
- Added support for DeepL machine translation service.
- Machine translation results are now cached inside Weblate.
- Добавлена поддержка переупорядочивания зафиксированных изменений.

4.10.2 Weblate 2.19.1

Выпущен 20 февраля 2018 года.

- Fixed migration issue on upgrade from 2.18.
- Improved file upload API validation.

4.10.3 Weblate 2.19

Выпущен 15 февраля 2018 года.

- Fixed imports across some file formats.
- Display human friendly browser information in audit log.
- Added TMX exporter for files.
- Various performance improvements for loading translation files.
- Added option to disable access management in Weblate in favor of Django one.

- Improved glossary lookup speed for large strings.
- Compatibility with django_auth_ldap 1.3.0.
- Configuration errors are now stored and reported persistently.
- Honor ignore flags in whitespace autofixer.
- Improved compatibility with some Subversion setups.
- Improved built in machine translation service.
- Added support for SAP Translation Hub service.
- Добавлена поддержка терминологической службы Майкрософт.
- Removed support for advertisement in notification e-mails.
- Improved translation progress reporting at language level.
- Improved support for different plural formulas.
- Added support for Subversion repositories not using stdlayout.
- Added addons to customize translation workflows.

4.10.4 Weblate 2.18

Выпущен 15 декабря 2017 года.

- Extended contributor stats.
- Improved configuration of special characters virtual keyboard.
- Added support for DTD file format.
- Changed keyboard shortcuts to less likely collide with browser/system ones.
- Improved support for approved flag in XLIFF files.
- Added support for not wrapping long strings in gettext PO files.
- Added button to copy permalink for current translation.
- Dropped support for Django 1.10 and added support for Django 2.0.
- Removed locking of translations while translating.
- Added support for adding new strings to monolingual translations.
- Added support for translation workflows with dedicated reviewers.

4.10.5 Weblate 2.17.1

Выпущен 13 октября 2017 года.

- Fixed running testsuite in some specific situations.
- Обновления локалей.

4.10.6 Weblate 2.17

Выпущен 13 октября 2017 года.

- Weblate by default does shallow Git clones now.
- Improved performance when updating large translation files.
- Added support for blocking certain e-mails from registration.
- Users can now delete their own comments.
- Added preview step to search and replace feature.
- Client side persistence of settings in search and upload forms.
- Расширены возможности поиска.
- More fine grained per project ACL configuration.
- Default value of BASE_DIR has been changed.
- Added two step account removal to prevent accidental removal.
- Project access control settings is now editable.
- Added optional spam protection for suggestions using Akismet.

4.10.7 Weblate 2.16

Выпущен 11 августа 2017 года.

- Various performance improvements.
- Added support for nested JSON format.
- Added support for WebExtension JSON format.
- Fixed git exporter authentication.
- Improved CSV import in certain situations.
- Improved look of Other translations widget.
- The max-length checks is now enforcing length of text in form.
- Make the commit_pending age configurable per component.
- Various user interface cleanups.
- Fixed component/project/site wide search for translations.

4.10.8 Weblate 2.15

Выпущен 30 июня 2017 года.

- Show more related translations in other translations.
- Add option to see translations of current string to other languages.
- Use 4 plural forms for Lithuanian by default.
- Fixed upload for monolingual files of different format.
- Improved error messages on failed authentication.
- Keep page state when removing word from glossary.
- Added direct link to edit secondary language translation.
- Added Perl format quality check.

- Added support for rejecting reused passwords.
- Extended toolbar for editing RTL languages.

4.10.9 Weblate 2.14.1

Выпущен 24 мая 2017 года.

- Fixed possible error when paginating search results.
- Fixed migrations from older versions in some corner cases.
- Fixed possible CSRF on project watch and unwatch.
- The password reset no longer authenticates user.
- Fixed possible CAPTCHA bypass on forgotten password.

4.10.10 Weblate 2.14

Выпущен 17 мая 2017 года.

- Add glossary entries using AJAX.
- The logout now uses POST to avoid CSRF.
- The API key token reset now uses POST to avoid CSRF.
- Weblate sets Content-Security-Policy by default.
- The local editor URL is validated to avoid self-XSS.
- The password is now validated against common flaws by default.
- Notify users about important activity with their account such as password change.
- The CSV exports now escape potential formulas.
- Various minor improvements in security.
- The authentication attempts are now rate limited.
- Suggestion content is stored in the history.
- Store important account activity in audit log.
- Ask for password confirmation when removing account or adding new associations.
- Show time when suggestion has been made.
- There is new quality check for trailing semicolon.
- Ensure that search links can be shared.
- Included source string information and screenshots in the API.
- Allow to overwrite translations through API upload.

4.10.11 Weblate 2.13.1

Выпущен 12 апреля 2017 года.

- Fixed listing of managed projects in profile.
- Fixed migration issue where some permissions were missing.
- Fixed listing of current file format in translation download.
- Return HTTP 404 when trying to access project where user lacks privileges.

4.10.12 Weblate 2.13

Выпущен 12 апреля 2017 года.

- Fixed quality checks on translation templates.
- Added quality check to trigger on losing translation.
- Add option to view pending suggestions from user.
- Add option to automatically build component lists.
- Default dashboard for unauthenticated users can be configured.
- Add option to browse 25 random strings for review.
- History now indicates string change.
- Better error reporting when adding new translation.
- Added per language search within project.
- Group ACLs can now be limited to certain permissions.
- The per project ALCs are now implemented using Group ACL.
- Added more fine grained privileges control.
- Various minor UI improvements.

4.10.13 Weblate 2.12

Выпущен 3 марта 2017 года.

- Improved admin interface for groups.
- Added support for Yandex Translate API.
- Improved speed of site wide search.
- Added project and component wide search.
- Added project and component wide search and replace.
- Improved rendering of inconsistent translations.
- Added support for opening source files in local editor.
- Added support for configuring visual keyboard with special characters.
- Improved screenshot management with OCR support for matching source strings.
- Default commit message now includes translation information and URL.
- Added support for Joomla translation format.
- Improved reliability of import across file formats.

4.10.14 Weblate 2.11

Выпущен 31 января 2017 года.

- Include language detailed information on language page.
- Mercurial backend improvements.
- Added option to specify translation component priority.
- More consistent usage of Group ACL even with less used permissions.
- Added WL_BRANCH variable to hook scripts.
- Improved developer documentation.
- Better compatibility with various Git versions in Git exporter addon.
- Included per project and component stats.
- Added language code mapping for better support of Microsoft Translate API.
- Moved fulltext cleanup to background job to make translation removal faster.
- Fixed displaying of plural source for languages with single plural form.
- Improved error handling in import_project.
- Various performance improvements.

4.10.15 Weblate 2.10.1

Выпущен 20 января 2017 года.

- Do not leak account existence on password reset form (CVE-2017-5537).

4.10.16 Weblate 2.10

Выпущен 15 декабря 2016 года.

- Added quality check to check whether plurals are translated differently.
- Fixed GitHub hooks for repositories with authentication.
- Added optional Git exporter module.
- Support for Microsoft Cognitive Services Translator API.
- Simplified project and component user interface.
- Added automatic fix to remove control characters.
- Added per language overview to project.
- Added support for CSV export.
- Added CSV download for stats.
- Added matrix view for quick overview of all translations
- Added basic API for changes and strings.
- Added support for Apertium APy server for machine translations.

4.10.17 Weblate 2.9

Выпущен 4 ноября 2016 года.

- Расширены параметры для команды управления createadmin.
- Extended import_json to be able to handle with existing components.
- Added support for YAML files.
- Project owners can now configure translation component and project details.
- Use «Watched» instead of «Subscribed» projects.
- Projects can be watched directly from project page.
- Added multi language status widget.
- Highlight secondary language if not showing source.
- Record suggestion deletion in history.
- Improved UX of languages selection in profile.
- Fixed showing whiteboard messages for component.
- Keep preferences tab selected after saving.
- Show source string comment more prominently.
- Automatically install Gettext PO merge driver for Git repositories.
- Added search and replace feature.
- Added support for uploading visual context (screenshots) for translations.

4.10.18 Weblate 2.8

Выпущен 31 августа 2016 года.

- Улучшения в документации.
- Переводы.
- Updated bundled javascript libraries.
- Добавлена команда управления list_translators.
- Django 1.8 is no longer supported.
- Fixed compatibility with Django 1.10.
- Added Subversion support.
- Separated XML validity check from XML mismatched tags.
- Fixed API to honor HIDE_REPO_CREDENTIALS settings.
- Show source change in Zen mode.
- Alt+PageUp/PageDown/Home/End now works in Zen mode as well.
- Add tooltip showing exact time of changes.
- Add option to select filters and search from translation page.
- Added UI for translation removal.
- Improved behavior when inserting placeables.
- Fixed auto locking issues in Zen mode.

4.10.19 Weblate 2.7

Выпущен 10 июля 2016 года.

- Removed Google web translate machine translation.
- Improved commit message when adding translation.
- Fixed Google Translate API for Hebrew language.
- Compatibility with Mercurial 3.8.
- Добавлена команда управления import_json.
- Correct ordering of listed translations.
- Show full suggestion text, not only a diff.
- Extend API (detailed repository status, statistics, …).
- Testsuite no longer requires network access to test repositories.

4.10.20 Weblate 2.6

Выпущен 28 апреля 2016 года.

- Исправлена проверка компонентов с языковым фильтром.
- Улучшена поддержка файлов XLIFF.
- Исправлен машинный перевод для неанглоязычных исходных текстов.
- Добавлен REST API.
- Совместимость с Django 1.10.
- К сообщениям совместной доски добавлены категории.

4.10.21 Weblate 2.5

Выпущен 10 марта 2016 года.

- Исправлен автоматический перевод для владельцев проектов.
- Улучшенная производительность выполнения коммитов и отправки в репозиторий.
- Новая команда управления для добавления предложений из командной строки.
- Добавлена поддержка объединения комментариев при загрузке файлов.
- Добавлена поддержка некоторых расширений GNU к формату функции printf C.
- Улучшения в документации.
- Добавлена поддержка генерации списка переводчиков.
- Добавлена поддержка генерации статистики участников.
- Поиск по всему сайту теперь можно вести только по одному языку.
- Улучшение проверок качества для армянского языка.
- Support for starting translation components without existing translations.
- Support for adding new translations in Qt TS.
- Improved support for translating PHP files.
- Performance improvements for quality checks.
- Исправлен поиск неудачных проверок по всему сайту.

- Added option to specify source language.
- Улучшена поддержка файлов XLIFF.
- Extended list of options for import_project.
- Improved targeting for whiteboard messages.
- Support for automatic translation across projects.
- Optimized fulltext search index.
- Добавлена команда управления для автоматического перевода.
- Added placeables highlighting.
- Added keyboard shortcuts for placeables, checks and machine translations.
- Improved translation locking.
- Added quality check for AngularJS interpolation.
- Added extensive group based ACLs.
- Clarified terminology on strings needing review (formerly fuzzy).
- Clarified terminology on strings needing action and not translated strings.
- Support for Python 3.
- Dropped support for Django 1.7.
- Dropped dependency on msginit for creating new gettext PO files.
- Added configurable dashboard views.
- Improved notifications on parse errors.
- Added option to import components with duplicate name to import_project.
- Improved support for translating PHP files
- Added XLIFF export for dictionary.
- Added XLIFF and gettext PO export for all translations.
- Улучшения в документации.
- Added support for configurable automatic group assignments.
- Improved adding of new translations.

4.10.22 Weblate 2.4

Выпущен 20 сентября 2015 года.

- Improved support for PHP files.
- Ability to add ACL to anonymous user.
- Improved configurability of import_project command.
- Added CSV dump of history.
- Avoid copy/paste errors with whitespace characters.
- Added support for Bitbucket webhooks.
- Tighter control on fuzzy strings on translation upload.
- Several URLs have changed, you might have to update your bookmarks.
- Hook scripts are executed with VCS root as current directory.
- Hook scripts are executed with environment variables describing current component.

- Добавлена команда управления для оптимизации полнотекстового индекса.
- 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.
- Добавлена команда управления для изменения имени сайта.
- Add option to configure default committer.
- Add hook after adding new translation.
- Add option to specify multiple files to add to commit.

4.10.23 Weblate 2.3

Выпущен 22 мая 2015 года.

- Dropped support for Django 1.6 and South migrations.
- Support for adding new translations when using Java Property files
- Allow to accept suggestion without editing.
- Improved support for Google OAuth 2.0
- Added support for Microsoft .resx files.
- Tuned default robots.txt to disallow big crawling of translations.
- Simplified workflow for accepting suggestions.
- Added project owners who always receive important notifications.
- Allow to disable editing of monolingual template.
- More detailed repository status view.
- Direct link for editing template when changing translation.
- Allow to add more permissions to project owners.
- Allow to show secondary language in Zen mode.
- Support for hiding source string in favor of secondary language.

4.10.24 Weblate 2.2

Выпущен 19 февраля 2015 года.

- Улучшения производительности.
- Fulltext search on location and comments fields.
- New SVG/javascript based activity charts.
- Support for Django 1.8.
- Support for deleting comments.
- Added own SVG badge.
- Added support for Google Analytics.
- Improved handling of translation filenames.
- Added support for monolingual JSON translations.
- Record component locking in a history.
- Support for editing source (template) language for monolingual translations.
- Added basic support for Gerrit.

4.10.25 Weblate 2.1

Выпущен 5 декабря 2014 года.

- Added support for Mercurial repositories.
- Replaced Glyphicon font by Awesome.
- Added icons for social authentication services.
- Better consistency of button colors and icons.
- Улучшения в документации.
- Исправления различных ошибок.
- Automatic hiding of columns in translation listing for small screens.
- Changed configuration of filesystem paths.
- Improved SSH keys handling and storage.
- Improved repository locking.
- Customizable quality checks per source string.
- Allow to hide completed translations from dashboard.

4.10.26 Weblate 2.0

Выпущен 6 ноября 2014 года.

- New responsive UI using Bootstrap.
- Rewritten VCS backend.
- Улучшения в документации.
- Added whiteboard for site wide messages.
- Configurable strings priority.
- Added support for JSON file format.

- Fixed generating mo files in certain cases.
- Added support for GitLab notifications.
- Added support for disabling translation suggestions.
- Django 1.7 support.
- ACL projects now have user management.
- Extended search possibilities.
- Give more hints to translators about plurals.
- Fixed Git repository locking.
- Compatibility with older Git versions.
- Improved ACL support.
- Added buttons for per language quotes and other special characters.
- Support for exporting stats as JSONP.

4.11 Серия Weblate 1.x

4.11.1 Weblate 1.9

Выпущен 6 мая 2014 года.

- Django 1.6 compatibility.
- No longer maintained compatibility with Django 1.4.
- Команды управления для блокировки/разблокировки переводов.
- Improved support for Qt TS files.
- Users can now delete their account.
- Avatars can be disabled.
- Merged first and last name attributes.
- Avatars are now fetched and cached server side.
- Added support for shields.io badge.

4.11.2 Weblate 1.8

Выпущен 7 ноября 2013 года.

- Please check manual for upgrade instructions.
- Nicer listing of project summary.
- Better visible options for sharing.
- More control over anonymous users privileges.
- Supports login using third party services, check manual for more details.
- Users can login by e-mail instead of username.
- Улучшения в документации.
- Improved source strings review.
- Searching across all strings.

- Better tracking of source strings.
- Captcha protection for registration.

4.11.3 Weblate 1.7

Выпущен 7 октября 2013 года.

- Please check manual for upgrade instructions.
- Support for checking Python brace format string.
- Per component customization of quality checks.
- Detailed per translation stats.
- Changed way of linking suggestions, checks and comments to strings.
- Users can now add text to commit message.
- Support for subscribing on new language requests.
- Support for adding new translations.
- Widgets and charts are now rendered using Pillow instead of Pango + Cairo.
- Add status badge widget.
- Dropped invalid text direction check.
- Changes in dictionary are now logged in history.
- Performance improvements for translating view.

4.11.4 Weblate 1.6

Выпущен 25 июля 2013 года.

- Nicer error handling on registration.
- Browsing of changes.
- Fixed sorting of machine translation suggestions.
- Improved support for MyMemory machine translation.
- Added support for Amagama machine translation.
- Various optimizations on frequently used pages.
- Highlights searched phrase in search results.
- Support for automatic fixups while saving the message.
- Tracking of translation history and option to revert it.
- Added support for Google Translate API.
- Added support for managing SSH host keys.
- Various form validation improvements.
- Various quality checks improvements.
- Performance improvements for import.
- Added support for voting on suggestions.
- Cleanup of admin interface.

4.11.5 Weblate 1.5

Выпущен 16 апреля 2013 года.

- Please check manual for upgrade instructions.
- Added public user pages.
- Better naming of plural forms.
- Added support for TBX export of glossary.
- Added support for Bitbucket notifications.
- Activity charts are now available for each translation, language or user.
- Extended options of import_project admin command.
- Compatible with Django 1.5.
- Avatars are now shown using libravatar.
- Added possibility to pretty print JSON export.
- Various performance improvements.
- Indicate failing checks or fuzzy strings in progress bars for projects or languages as well.
- Added support for custom pre-commit hooks and committing additional files.
- Rewritten search for better performance and user experience.
- New interface for machine translations.
- Added support for monolingual po files.
- Extend amount of cached metadata to improve speed of various searches.
- Now shows word counts as well.

4.11.6 Weblate 1.4

Выпущен 23 января 2013 года.

- Fixed deleting of checks/comments on string deletion.
- Добавлен параметр для отключения автоматического распространения переводов.
- Added option to subscribe for merge failures.
- Correctly import on projects which needs custom ttkit loader.
- Added sitemaps to allow easier access by crawlers.
- Provide direct links to string in notification e-mails or feeds.
- Various improvements to admin interface.
- В интерфейс администратора добавлены подсказки по настройке Weblate для рабочей среды.
- Added per language widgets and engage page.
- Improved translation locking handling.
- Show code snippets for widgets in more variants.
- Indicate failing checks or fuzzy strings in progress bars.
- More options for formatting commit message.
- Fixed error handling with machine translation services.
- Improved automatic translation locking behaviour.

- Support for showing changes from previous source string.
- Added support for substring search.
- Various quality checks improvements.
- Support for per project ACL.
- Basic string tests coverage.

4.11.7 Weblate 1.3

Выпущен 16 ноября 2012 года.

- Compatibility with PostgreSQL database backend.
- Removes languages removed in upstream git repository.
- Improved quality checks processing.
- Added new checks (BB code, XML markup and newlines).
- Support for optional rebasing instead of merge.
- Possibility to relocate Weblate (for example to run it under /weblate path).
- Support for manually choosing file type in case autodetection fails.
- Better support for Android resources.
- Support for generating SSH key from web interface.
- More visible data exports.
- New buttons to enter some special characters.
- Support for exporting dictionary.
- Support for locking down whole Weblate installation.
- Checks for source strings and support for source strings review.
- Support for user comments for both translations and source strings.
- Better changes log tracking.
- Changes can now be monitored using RSS.
- Improved support for RTL languages.

4.11.8 Weblate 1.2

Выпущен 14 августа 2012 года.

- Weblate now uses South for database migration, please check upgrade instructions if you are upgrading.
- Fixed minor issues with linked git repos.
- New introduction page for engaging people with translating using Weblate.
- Added widgets which can be used for promoting translation projects.
- Added option to reset repository to origin (for privileged users).
- Project or component can now be locked for translations.
- Possibility to disable some translations.
- Configurable options for adding new translations.
- Configuration of git commits per project.

- Simple antispam protection.
- Better layout of main page.
- Support for automatically pushing changes on every commit.
- Support for e-mail notifications of translators.
- List only used languages in preferences.
- Improved handling of not known languages when importing project.
- Support for locking translation by translator.
- Optionally maintain `Language-Team` header in po file.
- Include some statistics in about page.
- Supports (and requires) django-registration 0.8.
- Caching of counted strings with failing checks.
- Checking of requirements during setup.
- Улучшения в документации.

4.11.9 Weblate 1.1

Выпущен 4 июля 2012 года.

- Improved several translations.
- Better validation while creating component.
- Added support for shared git repositories across components.
- Do not necessary commit on every attempt to pull remote repo.
- Added support for offloading indexing.

4.11.10 Weblate 1.0

Выпущен 10 мая 2012 года.

- Improved validation while adding/saving component.
- Experimental support for Android component files (needs patched ttkit).
- Updates from hooks are run in background.
- Improved installation instructions.
- Improved navigation in dictionary.

4.12 Серия Weblate 0.x

4.12.1 Weblate 0.9

Выпущен 18 апреля 2012 года.

- Fixed import of unknown languages.
- Improved listing of nearby messages.
- Improved several checks.
- Обновление документации.

- Added definition for several more languages.
- Various code cleanups.
- Улучшения в документации.
- Changed file layout.
- Update helper scripts to Django 1.4.
- Improved navigation while translating.
- Better handling of po file renames.
- Better validation while creating component.
- Integrated full setup into syncdb.
- Added list of recent changes to all translation pages.
- Check for not translated strings ignores format string only messages.

4.12.2 Weblate 0.8

Выпущен 3 апреля 2012 года.

- Replaced own full text search with Whoosh.
- Various fixes and improvements to checks.
- New command updatechecks.
- Lot of translation updates.
- Added dictionary for storing most frequently used terms.
- Added /admin/report/ for overview of repositories status.
- Machine translation services no longer block page loading.
- Management interface now contains also useful actions to update data.
- Records log of changes made by users.
- Ability to postpone commit to Git to generate less commits from single user.
- Possibility to browse failing checks.
- Automatic translation using already translated strings.
- New about page showing used versions.
- Django 1.4 compatibility.
- Ability to push changes to remote repo from web interface.
- Added review of translations done by others.

4.12.3 Weblate 0.7

Выпущен 16 февраля 2012 года.

- Direct support for GitHub notifications.
- Added support for cleaning up orphaned checks and translations.
- Displays nearby strings while translating.
- Displays similar strings while translating.
- Improved searching for string.

4.12.4 Weblate 0.6

Выпущен 14 февраля 2012 года.

- Added various checks for translated messages.
- Tunable access control.
- Improved handling of translations with new lines.
- Added client side sorting of tables.
- Please check upgrading instructions in case you are upgrading.

4.12.5 Weblate 0.5

Выпущен 12 февраля 2012 года.

- **Support for machine translation using following online services:**
  - Apertium
  - Microsoft Translator
  - MyMemory
- Several new translations.
- Improved merging of upstream changes.
- Better handle concurrent git pull and translation.
- Propagating works for fuzzy changes as well.
- Propagating works also for file upload.
- Fixed file downloads while using FastCGI (and possibly others).

4.12.6 Weblate 0.4

Выпущен 8 февраля 2012 года.

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

4.12.7 Weblate 0.3

Выпущен 8 февраля 2012 года.

- Better display of source for plural translations.
- New documentation in Sphinx format.
- Displays secondary languages while translating.
- Improved error page to give list of existing projects.
- New per language stats.

4.12.8 Weblate 0.2

Выпущен 7 февраля 2012 года.

- Improved validation of several forms.
- Warn users on profile upgrade.
- Remember URL for login.
- Naming of text areas while entering plural forms.
- Automatic expanding of translation area.

4.12.9 Weblate 0.1

Выпущен 6 февраля 2012 года.

- Начальный выпуск.

Содержание модулей Python

W

wlc, 145
wlc.config, 146
wlc.main, 146

HTTP Routing Table

/

ANY /, 101

/api

GET /api/, 103
GET /api/changes/, 133
GET /api/changes/(int:id)/, 133
GET /api/component-lists/, 136
GET /api/component-lists/(str:slug)/, 136
GET /api/components/, 118
GET /api/components/(string:project)/(string:component)/, 118
GET /api/components/(string:project)/(string:component)/changes/, 121
GET /api/components/(string:project)/(string:component)/lock/, 122
GET /api/components/(string:project)/(string:component)/monolingual_base/, 124
GET /api/components/(string:project)/(string:component)/new_template/, 124
GET /api/components/(string:project)/(string:component)/repository/, 123
GET /api/components/(string:project)/(string:component)/screenshots/, 122
GET /api/components/(string:project)/(string:component)/statistics/, 126
GET /api/components/(string:project)/(string:component)/translations/, 124
GET /api/groups/, 107
GET /api/groups/(int:id)/, 107
GET /api/languages/, 111
GET /api/languages/(string:language)/, 111
GET /api/languages/(string:language)/statistics/, 112
GET /api/projects/, 113
GET /api/projects/(string:project)/, 113
GET /api/projects/(string:project)/changes/, 114
GET /api/projects/(string:project)/components/, 115
GET /api/projects/(string:project)/languages/, 117
GET /api/projects/(string:project)/repository/, 114
GET /api/projects/(string:project)/statistics/, 117
GET /api/roles/, 110
GET /api/roles/(int:id)/, 110
GET /api/screenshots/, 134
GET /api/screenshots/(int:id)/, 134
GET /api/screenshots/(int:id)/file/, 134
GET /api/translations/, 126
GET /api/translations/(string:project)/(string:component)/changes/, 126
GET /api/translations/(string:project)/(string:component)/lock/, 129
GET /api/translations/(string:project)/(string:component)/monolingual_base/, 130
GET /api/translations/(string:project)/(string:component)/new_template/, 130
GET /api/translations/(string:project)/(string:component)/repository/, 131
GET /api/translations/(string:project)/(string:component)/screenshots/, 129
GET /api/units/, 132
GET /api/units/(int:id)/, 132
GET /api/users/, 104
GET /api/users/(str:username)/, 104
GET /api/users/(str:username)/notifications/, 105
GET /api/users/(str:username)/notifications/(int:s 106
POST /api/component-lists/(str:slug)/components/, 137
POST /api/components/(string:project)/(string:comp 122
POST /api/components/(string:project)/(string:comp 123
POST /api/components/(string:project)/(string:comp 125
POST /api/groups/, 107
POST /api/groups/(int:id)/componentlists/,

109
POST /api/groups/(int:id)/components/, 108
POST /api/groups/(int:id)/languages/, 109
POST /api/groups/(int:id)/projects/, 109
POST /api/groups/(int:id)/roles/, 108
POST /api/languages/, 111
POST /api/projects/, 113
POST /api/projects/(string:project)/components/, 115
POST /api/projects/(string:project)/repository/, 114
POST /api/roles/, 110
POST /api/screenshots/, 135
POST /api/screenshots/(int:id)/file/, 134
POST /api/screenshots/(int:id)/units/, 134
POST /api/translations/(string:project)/(string:component)/(string:language)/autotranslate/, 129
POST /api/translations/(string:project)/(string:component)/(string:language)/file/, 130
POST /api/translations/(string:project)/(string:component)/(string:language)/repository/, 131
POST /api/translations/(string:project)/(string:component)/(string:language)/statistics/, 129
POST /api/users/, 104
POST /api/users/(str:username)/groups/, 105
POST /api/users/(str:username)/notifications/, 105
PUT /api/component-lists/(str:slug)/, 136
PUT /api/components/(string:project)/(string:component)/, 121
PUT /api/groups/(int:id)/, 108
PUT /api/languages/(string:language)/, 112
PUT /api/roles/(int:id)/, 110
PUT /api/screenshots/(int:id)/, 135
PUT /api/users/(str:username)/, 104
PUT /api/users/(str:username)/notifications/(int:subscription_id)/, 106
DELETE /api/component-lists/(str:slug)/, 136
DELETE /api/component-lists/(str:slug)/components/(str:component_slug), 137
DELETE /api/components/(string:project)/(string:component)/, 121
DELETE /api/groups/(int:id)/, 108
DELETE /api/groups/(int:id)/componentlists/(int:component_list_id), 109
DELETE /api/groups/(int:id)/components/(int:component_id), 108
DELETE /api/groups/(int:id)/languages/(string:language_code), 109
DELETE /api/groups/(int:id)/projects/(int:project_, 109
DELETE /api/languages/(string:language)/, 112
DELETE /api/projects/(string:project)/, 114
DELETE /api/roles/(int:id)/, 110
DELETE /api/screenshots/(int:id)/, 136
DELETE /api/screenshots/(int:id)/units/(int:unit_, 135
DELETE /api/translations/(string:project)/(string, 128
DELETE /api/users/(str:username)/, 105
DELETE /api/users/(str:username)/notifications/(in, 106
PATCH /api/component-lists/(str:slug)/, 136
PATCH /api/components/(string:project)/(string:com, 120
PATCH /api/components/(string:project)/(string:component)/(string:language)/autotransla, 108
PATCH /api/languages/(string:language)/, 112
PATCH /api/roles/(int:id)/, 110
PATCH /api/screenshots/(int:id)/, 135
PATCH /api/users/(str:username)/, 105
PATCH /api/users/(str:username)/notifications/(int, 106

/exports

GET /exports/rss/, 141
GET /exports/rss/(string:project)/, 140
GET /exports/rss/(string:project)/(string:componen, 140
GET /exports/rss/(string:project)/(string:componen, 140
GET /exports/rss/language/(string:language)/, 140
GET /exports/stats/(string:project)/(string:compon, 139

/hooks

GET /hooks/update/(string:project)/, 137
GET /hooks/update/(string:project)/(string:compone, 137
POST /hooks/azure/, 138
POST /hooks/bitbucket/, 138
POST /hooks/gitea/, 138
POST /hooks/gitee/, 139
POST /hooks/github/, 137
POST /hooks/gitlab/, 138
POST /hooks/pagure/, 138

Алфавитный указатель

Символы

.XML resource file
file format, 87
--add
auto_translate command line option, 328
--addon ADDON
install_addon command line option, 334
--age HOURS
commit_pending command line option, 329
--author USER@EXAMPLE.COM
add_suggestions command line option, 328
--base-file-template TEMPLATE
import_project command line option, 332
--check
importusers command line option, 334
--config PATH
wlc command line option, 142
--config-section SECTION
wlc command line option, 142
--configuration CONFIG
install_addon command line option, 334
--convert
wlc command line option, 143
--email USER@EXAMPLE.COM
createadmin command line option, 330
--file-format FORMAT
import_project command line option, 332
--force
loadpo command line option, 335
--force-commit
pushgit command line option, 336
--format {csv,json,text,html}
wlc command line option, 142
--ignore
import_json command line option, 331
--inconsistent
auto_translate command line option, 328
--input
wlc command line option, 143
--key KEY
wlc command line option, 142
--lang LANGUAGE
loadpo command line option, 335
--language-code
list_translators command line option, 335
--language-map LANGMAP
import_memory command line option, 331
--language-regex REGEX
import_project command line option, 332
--license NAME
import_project command line option, 332
--license-url URL
import_project command line option, 332
--main-component
import_project command line option, 332
--main-component COMPONENT
import_json command line option, 331
--mt MT
auto_translate command line option, 328
--name
createadmin command line option, 330
--name-template TEMPLATE
import_project command line option, 332
--new-base-template TEMPLATE
import_project command line option, 332
--no-password
createadmin command line option, 330
--no-privs-update
setupgroups command line option, 336
--no-projects-update

setupgroups command line option, 336
--no-update
setuplang command line option, 337
--output
wlc command line option, 143
--overwrite
auto_translate command line option, 328
wlc command line option, 143
--password PASSWORD
createadmin command line option, 330
--project PROJECT
import_json command line option, 331
--source PROJECT/COMPONENT
auto_translate command line option, 328
--threshold THRESHOLD
auto_translate command line option, 328
--update
createadmin command line option, 330
import_json command line option, 331
install_addon command line option, 334
--url URL
wlc command line option, 142
--user USERNAME
auto_translate command line option, 328
--username USERNAME
createadmin command line option, 330
--vcs NAME
import_project command line option, 332

A

add_suggestions
weblate admin command, 328
add_suggestions command line option
--author USER@EXAMPLE.COM, 328
ADMINS
setting, 184
AKISMET_API_KEY
setting, 289
ALLOWED_HOSTS
setting, 184
Android
file format, 82
ANONYMOUS_USER_NAME
setting, 289
API, 100, 141, 145
Apple strings
file format, 83
ARB
file format, 86
AUDITLOG_EXPIRY
setting, 289
AUTH_LOCK_ATTEMPTS
setting, 289
AUTH_TOKEN_VALID
setting, 290
auto_translate
weblate admin command, 328
auto_translate command line option
--add, 328
--inconsistent, 328
--mt MT, 328
--overwrite, 328
--source PROJECT/COMPONENT, 328
--threshold THRESHOLD, 328
--user USERNAME, 328
AUTO_UPDATE
setting, 289
AUTOFIX_LIST
setting, 290
AVATAR_URL_PREFIX
setting, 290

B

BASE_DIR
setting, 291

C

celery_queues
weblate admin command, 329
changes
wlc command line option, 143
CHECK_LIST
setting, 291
checkgit
weblate admin command, 329
cleanup
wlc command line option, 143
cleanuptrans
weblate admin command, 329
Comma separated values
file format, 87
Command (*класс в wlc.main*), 146
COMMENT_CLEANUP_DAYS
setting, 292
commit
wlc command line option, 142
commit_pending
weblate admin command, 329
commit_pending command line option
--age HOURS, 329
COMMIT_PENDING_HOURS
setting, 292
commitgit
weblate admin command, 329
createadmin
weblate admin command, 330
createadmin command line option
--email USER@EXAMPLE.COM, 330
--name, 330
--no-password, 330

--password PASSWORD, 330
--update, 330
--username USERNAME, 330
CSP_CONNECT_SRC
setting, 291
CSP_FONT_SRC
setting, 291
CSP_IMG_SRC
setting, 291
CSP_SCRIPT_SRC
setting, 291
CSP_STYLE_SRC
setting, 291
CSV
file format, 87

D

DATA_DIR
setting, 292
DATABASE_BACKUP
setting, 293
DATABASES
setting, 185
DEBUG
setting, 185
DEFAULT_ACCESS_CONTROL
setting, 293
DEFAULT_ADD_MESSAGE
setting, 294
DEFAULT_ADDON_MESSAGE
setting, 294
DEFAULT_ADDONS
setting, 294
DEFAULT_COMMIT_MESSAGE
setting, 294
DEFAULT_COMMITER_EMAIL
setting, 294
DEFAULT_COMMITER_NAME
setting, 294
DEFAULT_DELETE_MESSAGE
setting, 294
DEFAULT_FROM_EMAIL
setting, 185
DEFAULT_MERGE_MESSAGE
setting, 294
DEFAULT_MERGE_STYLE
setting, 295
DEFAULT_PULL_MESSAGE
setting, 295
DEFAULT_RESTRICTED_COMPONENT
setting, 293
DEFAULT_TRANSLATION_PROPAGATION
setting, 295
download
wlc command line option, 143
DTD
file format, 89
dump_memory
weblate admin command, 330
dumpuserdata
weblate admin command, 330

E

ENABLE_AVATARS
setting, 295
ENABLE_HOOKS
setting, 295
ENABLE_HTTPS
setting, 295
ENABLE_SHARING
setting, 296

F

file format
.XML resource file, 87
Android, 82
Apple strings, 83
ARB, 86
Comma separated values, 87
CSV, 87
DTD, 89
gettext, 77
go-i18n, 85
GWT properties, 80
i18next, 85
INI translations, 80, 81
Java properties, 80
Joomla translations, 81
JSON, 84
PHP strings, 83
PO, 77
Qt, 82
RC, 89
RESX, 87
Ruby YAML, 88
Ruby YAML Ain't Markup Language, 88
string resources, 82
TS, 82
XLIFF, 78
XML, 89
YAML, 88
YAML Ain't Markup Language, 88

G

get() (*метод wlc.Weblate*), 145
gettext
file format, 77
GITHUB_USERNAME
setting, 296
GITLAB_USERNAME
setting, 296
go-i18n
file format, 85
GOOGLE_ANALYTICS_ID
setting, 296
GWT properties

file format, 80

H

HIDE_REPO_CREDENTIALS
setting, 296

I

i18next
file format, 85
import_demo
weblate admin command, 330
import_json
weblate admin command, 331
import_json command line option
--ignore, 331
--main-component COMPONENT, 331
--project PROJECT, 331
--update, 331
import_memory
weblate admin command, 331
import_memory command line option
--language-map LANGMAP, 331
import_project
weblate admin command, 332
import_project command line option
--base-file-template TEMPLATE, 332
--file-format FORMAT, 332
--language-regex REGEX, 332
--license NAME, 332
--license-url URL, 332
--main-component, 332
--name-template TEMPLATE, 332
--new-base-template TEMPLATE, 332
--vcs NAME, 332
importuserdata
weblate admin command, 334
importusers
weblate admin command, 334
importusers command line option
--check, 334
INI translations
file format, 80, 81
install_addon
weblate admin command, 334
install_addon command line option
--addon ADDON, 334
--configuration CONFIG, 334
--update, 334
IP_BEHIND_REVERSE_PROXY
setting, 296
IP_PROXY_HEADER
setting, 297
IP_PROXY_OFFSET
setting, 297
iPad
перевод, 83
iPhone
перевод, 83

J

Java properties
file format, 80
Joomla translations
file format, 81
JSON
file format, 84

L

LEGAL_URL
setting, 297
LICENSE_EXTRA
setting, 298
LICENSE_FILTER
setting, 298
LICENSE_REQUIRED
setting, 298
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH
setting, 298
list_languages
weblate admin command, 334
list_translators
weblate admin command, 335
list_translators command line option
--language-code, 335
list_versions
weblate admin command, 335
list-components
wlc command line option, 142
list-languages
wlc command line option, 142
list-projects
wlc command line option, 142
list-translations
wlc command line option, 142
load() (*метод wlc.config.WeblateConfig*), 146
loadpo
weblate admin command, 335
loadpo command line option
--force, 335
--lang LANGUAGE, 335
LOCALIZE_CDN_PATH
setting, 299
LOCALIZE_CDN_URL
setting, 299
lock
wlc command line option, 143
lock_translation
weblate admin command, 335
lock-status
wlc command line option, 143
LOGIN_REQUIRED_URLS
setting, 299
LOGIN_REQUIRED_URLS_EXCEPTIONS
setting, 299
ls
wlc command line option, 142

M

MACHINE_TRANSLATION_SERVICES
    setting, 300
main() (*в модуле wlc.main*), 146
MATOMO_SITE_ID
    setting, 300
MATOMO_URL
    setting, 300
move_language
    weblate admin command, 336
MT_APERTIUM_APY
    setting, 301
MT_AWS_ACCESS_KEY_ID
    setting, 301
MT_AWS_REGION
    setting, 301
MT_AWS_SECRET_ACCESS_KEY
    setting, 301
MT_BAIDU_ID
    setting, 301
MT_BAIDU_SECRET
    setting, 301
MT_DEEPL_API_VERSION
    setting, 302
MT_DEEPL_KEY
    setting, 302
MT_GOOGLE_CREDENTIALS
    setting, 302
MT_GOOGLE_KEY
    setting, 302
MT_GOOGLE_LOCATION
    setting, 302
MT_GOOGLE_PROJECT
    setting, 302
MT_MICROSOFT_BASE_URL
    setting, 303
MT_MICROSOFT_COGNITIVE_KEY
    setting, 303
MT_MICROSOFT_ENDPOINT_URL
    setting, 303
MT_MICROSOFT_REGION
    setting, 303
MT_MODERNMT_KEY
    setting, 303
MT_MODERNMT_URL
    setting, 303
MT_MYMEMORY_EMAIL
    setting, 304
MT_MYMEMORY_KEY
    setting, 304
MT_MYMEMORY_USER
    setting, 304
MT_NETEASE_KEY
    setting, 304
MT_NETEASE_SECRET
    setting, 304
MT_SAP_BASE_URL
    setting, 305
MT_SAP_PASSWORD
    setting, 305
MT_SAP_SANDBOX_APIKEY
    setting, 305
MT_SAP_USE_MT
    setting, 305
MT_SAP_USERNAME
    setting, 305
MT_SERVICES
    setting, 300
MT_TMSERVER
    setting, 304
MT_YANDEX_KEY
    setting, 304
MT_YOUDAO_ID
    setting, 305
MT_YOUDAO_SECRET
    setting, 305

N

NEARBY_MESSAGES
    setting, 306

P

PHP strings
    file format, 83
PIWIK_SITE_ID
    setting, 300
PIWIK_URL
    setting, 300
PO
    file format, 77
post() (*метод wlc.Weblate*), 145
pull
    wlc command line option, 142
push
    wlc command line option, 142
pushgit
    weblate admin command, 336
pushgit command line option
    --force-commit, 336
Python, 145

Q

Qt
    file format, 82

R

RATELIMIT_ATTEMPTS
    setting, 306
RATELIMIT_LOCKOUT
    setting, 306
RATELIMIT_WINDOW
    setting, 306
RC
    file format, 89
register_command() (*в модуле wlc.main*), 146
REGISTRATION_ALLOW_BACKENDS

setting, 306
REGISTRATION_CAPTCHA
setting, 307
REGISTRATION_EMAIL_MATCH
setting, 307
REGISTRATION_OPEN
setting, 307
repo
wlc command line option, 143
REPOSITORY_ALERT_THRESHOLD
setting, 307
reset
wlc command line option, 142
REST, 100
RESX
file format, 87
RFC
RFC 4646, 74
Ruby YAML
file format, 88
Ruby YAML Ain't Markup Language
file format, 88

S

SECRET_KEY
setting, 185
SENTRY_DSN
setting, 308
SERVER_EMAIL
setting, 185
SESSION_ENGINE
setting, 185
setting
ADMINS, 184
AKISMET_API_KEY, 289
ALLOWED_HOSTS, 184
ANONYMOUS_USER_NAME, 289
AUDITLOG_EXPIRY, 289
AUTH_LOCK_ATTEMPTS, 289
AUTH_TOKEN_VALID, 290
AUTO_UPDATE, 289
AUTOFIX_LIST, 290
AVATAR_URL_PREFIX, 290
BASE_DIR, 291
CHECK_LIST, 291
COMMENT_CLEANUP_DAYS, 292
COMMIT_PENDING_HOURS, 292
CSP_CONNECT_SRC, 291
CSP_FONT_SRC, 291
CSP_IMG_SRC, 291
CSP_SCRIPT_SRC, 291
CSP_STYLE_SRC, 291
DATA_DIR, 292
DATABASE_BACKUP, 293
DATABASES, 185
DEBUG, 185
DEFAULT_ACCESS_CONTROL, 293
DEFAULT_ADD_MESSAGE, 294
DEFAULT_ADDON_MESSAGE, 294
DEFAULT_ADDONS, 294
DEFAULT_COMMIT_MESSAGE, 294
DEFAULT_COMMITER_EMAIL, 294
DEFAULT_COMMITER_NAME, 294
DEFAULT_DELETE_MESSAGE, 294
DEFAULT_FROM_EMAIL, 185
DEFAULT_MERGE_MESSAGE, 294
DEFAULT_MERGE_STYLE, 295
DEFAULT_PULL_MESSAGE, 295
DEFAULT_RESTRICTED_COMPONENT, 293
DEFAULT_TRANSLATION_PROPAGATION, 295
ENABLE_AVATARS, 295
ENABLE_HOOKS, 295
ENABLE_HTTPS, 295
ENABLE_SHARING, 296
GITHUB_USERNAME, 296
GITLAB_USERNAME, 296
GOOGLE_ANALYTICS_ID, 296
HIDE_REPO_CREDENTIALS, 296
IP_BEHIND_REVERSE_PROXY, 296
IP_PROXY_HEADER, 297
IP_PROXY_OFFSET, 297
LEGAL_URL, 297
LICENSE_EXTRA, 298
LICENSE_FILTER, 298
LICENSE_REQUIRED, 298
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH, 298
LOCALIZE_CDN_PATH, 299
LOCALIZE_CDN_URL, 299
LOGIN_REQUIRED_URLS, 299
LOGIN_REQUIRED_URLS_EXCEPTIONS, 299
MACHINE_TRANSLATION_SERVICES, 300
MATOMO_SITE_ID, 300
MATOMO_URL, 300
MT_APERTIUM_APY, 301
MT_AWS_ACCESS_KEY_ID, 301
MT_AWS_REGION, 301
MT_AWS_SECRET_ACCESS_KEY, 301
MT_BAIDU_ID, 301
MT_BAIDU_SECRET, 301
MT_DEEPL_API_VERSION, 302
MT_DEEPL_KEY, 302
MT_GOOGLE_CREDENTIALS, 302
MT_GOOGLE_KEY, 302
MT_GOOGLE_LOCATION, 302
MT_GOOGLE_PROJECT, 302
MT_MICROSOFT_BASE_URL, 303
MT_MICROSOFT_COGNITIVE_KEY, 303
MT_MICROSOFT_ENDPOINT_URL, 303
MT_MICROSOFT_REGION, 303
MT_MODERNMT_KEY, 303
MT_MODERNMT_URL, 303
MT_MYMEMORY_EMAIL, 304
MT_MYMEMORY_KEY, 304
MT_MYMEMORY_USER, 304

MT_NETEASE_KEY, 304
MT_NETEASE_SECRET, 304
MT_SAP_BASE_URL, 305
MT_SAP_PASSWORD, 305
MT_SAP_SANDBOX_APIKEY, 305
MT_SAP_USE_MT, 305
MT_SAP_USERNAME, 305
MT_SERVICES, 300
MT_TMSERVER, 304
MT_YANDEX_KEY, 304
MT_YOUDAO_ID, 305
MT_YOUDAO_SECRET, 305
NEARBY_MESSAGES, 306
PIWIK_SITE_ID, 300
PIWIK_URL, 300
RATELIMIT_ATTEMPTS, 306
RATELIMIT_LOCKOUT, 306
RATELIMIT_WINDOW, 306
REGISTRATION_ALLOW_BACKENDS, 306
REGISTRATION_CAPTCHA, 307
REGISTRATION_EMAIL_MATCH, 307
REGISTRATION_OPEN, 307
REPOSITORY_ALERT_THRESHOLD, 307
SECRET_KEY, 185
SENTRY_DSN, 308
SERVER_EMAIL, 185
SESSION_ENGINE, 185
SIMPLIFY_LANGUAGES, 308
SINGLE_PROJECT, 309
SITE_DOMAIN, 308
SITE_TITLE, 308
SPECIAL_CHARS, 309
STATUS_URL, 309
SUGGESTION_CLEANUP_DAYS, 309
URL_PREFIX, 309
VCS_BACKENDS, 309
VCS_CLONE_DEPTH, 310
WEBLATE_ADDONS, 310
WEBLATE_EXPORTERS, 311
WEBLATE_FORMATS, 311
WEBLATE_GPG_IDENTITY, 311
setupgroups
weblate admin command, 336
setupgroups command line option
--no-privs-update, 336
--no-projects-update, 336
setuplang
weblate admin command, 337
setuplang command line option
--no-update, 337
show
wlc command line option, 142
SIMPLIFY_LANGUAGES
setting, 308
SINGLE_PROJECT
setting, 309
SITE_DOMAIN
setting, 308
SITE_TITLE
setting, 308
SPECIAL_CHARS
setting, 309
statistics
wlc command line option, 143
STATUS_URL
setting, 309
string resources
file format, 82
SUGGESTION_CLEANUP_DAYS
setting, 309

T

TS
file format, 82

U

unlock
wlc command line option, 143
unlock_translation
weblate admin command, 336
updatechecks
weblate admin command, 337
updategit
weblate admin command, 337
upload
wlc command line option, 143
URL_PREFIX
setting, 309

V

VCS_BACKENDS
setting, 309
VCS_CLONE_DEPTH
setting, 310
version
wlc command line option, 142

W

Weblate (*класс в wlc*), 145
weblate admin command
add_suggestions, 328
auto_translate, 328
celery_queues, 329
checkgit, 329
cleanuptrans, 329
commit_pending, 329
commitgit, 329
createadmin, 330
dump_memory, 330
dumpuserdata, 330
import_demo, 330
import_json, 331
import_memory, 331
import_project, 332
importuserdata, 334
importusers, 334

install_addon, 334
list_languages, 334
list_translators, 335
list_versions, 335
loadpo, 335
lock_translation, 335
move_language, 336
pushgit, 336
setupgroups, 336
setuplang, 337
unlock_translation, 336
updatechecks, 337
updategit, 337
WEBLATE_ADDONS
setting, 310
WEBLATE_ADMIN_EMAIL, 151, 152, 156
WEBLATE_ADMIN_NAME, 151, 152
WEBLATE_ADMIN_PASSWORD, 148, 151, 152
WEBLATE_ALLOWED_HOSTS, 151, 185, 189, 308
WEBLATE_EMAIL_PORT, 161
WEBLATE_EMAIL_USE_SSL, 161
WEBLATE_EMAIL_USE_TLS, 161
WEBLATE_EXPORTERS
setting, 311
WEBLATE_FORMATS
setting, 311
WEBLATE_GITHUB_USERNAME, 98
WEBLATE_GITLAB_USERNAME, 99
WEBLATE_GPG_IDENTITY
setting, 311
WEBLATE_SILENCED_SYSTEM_CHECKS, 209
WEBLATE_SITE_DOMAIN, 187, 205, 308
WeblateConfig (*класс в wlc.config*), 146
WeblateException, 145
wlc, 141
модуль, 145
wlc command line option
--config PATH, 142
--config-section SECTION, 142
--convert, 143
--format {csv,json,text,html}, 142
--input, 143
--key KEY, 142
--output, 143
--overwrite, 143
--url URL, 142
changes, 143
cleanup, 143
commit, 142
download, 143
list-components, 142
list-languages, 142
list-projects, 142
list-translations, 142
lock, 143
lock-status, 143
ls, 142
pull, 142
push, 142
repo, 143
reset, 142
show, 142
statistics, 143
unlock, 143
upload, 143
version, 142
wlc.config
модуль, 146
wlc.main
модуль, 146

X

XLIFF
file format, 78
XML
file format, 89

Y

YAML
file format, 88
YAML Ain't Markup Language
file format, 88

⍰

двуязычный формат
перевод, 74
модуль
wlc, 145
wlc.config, 146
wlc.main, 146
одноязычный формат
перевод, 74
перевод
iPad, 83
iPhone, 83
двуязычный формат, 74
одноязычный формат, 74
переменная окружения
CELERY_BACKUP_OPTIONS, 162
CELERY_BEAT_OPTIONS, 162
CELERY_MAIN_OPTIONS, 162
CELERY_MEMORY_OPTIONS, 162
CELERY_NOTIFY_OPTIONS, 162
CELERY_TRANSLATE_OPTIONS, 162
POSTGRES_DATABASE, 159
POSTGRES_HOST, 159
POSTGRES_PASSWORD, 159
POSTGRES_PORT, 159
POSTGRES_SSL_MODE, 159
POSTGRES_USER, 159
REDIS_DB, 160
REDIS_HOST, 160
REDIS_PASSWORD, 160
REDIS_PORT, 160
REDIS_TLS, 160
REDIS_VERIFY_SSL, 160

ROLLBAR_ENVIRONMENT, 162
ROLLBAR_KEY, 162
SENTRY_DSN, 162
SENTRY_ENVIRONMENT, 162
SOCIAL_AUTH_SLACK_SECRET, 159
UWSGI_WORKERS, 163
WEBLATE_ADD_ADDONS, 162
WEBLATE_ADD_APPS, 162
WEBLATE_ADD_AUTOFIX, 162
WEBLATE_ADD_CHECK, 162
WEBLATE_ADD_LOGIN_REQUIRED_URLS_EXCEPTIONS, 154
WEBLATE_ADMIN_EMAIL, 151, 152, 156
WEBLATE_ADMIN_NAME, 151, 152
WEBLATE_ADMIN_PASSWORD, 148, 151, 152
WEBLATE_AKISMET_API_KEY, 155
WEBLATE_ALLOWED_HOSTS, 151, 152, 185, 189, 308
WEBLATE_AUTH_LDAP_BIND_DN, 156
WEBLATE_AUTH_LDAP_BIND_PASSWORD, 156
WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS, 156
WEBLATE_AUTH_LDAP_SERVER_URI, 156
WEBLATE_AUTH_LDAP_USER_ATTR_MAP, 156
WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE, 156
WEBLATE_AUTH_LDAP_USER_SEARCH, 156
WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER, 156
WEBLATE_CSP_CONNECT_SRC, 155
WEBLATE_CSP_FONT_SRC, 155
WEBLATE_CSP_IMG_SRC, 155
WEBLATE_CSP_SCRIPT_SRC, 155
WEBLATE_CSP_STYLE_SRC, 155
WEBLATE_DATABASE_BACKUP, 160
WEBLATE_DEBUG, 151
WEBLATE_DEFAULT_ACCESS_CONTROL, 154
WEBLATE_DEFAULT_FROM_EMAIL, 152
WEBLATE_DEFAULT_RESTRICTED_COMPONENT, 154
WEBLATE_DEFAULT_TRANSLATION_PROPAGATION, 154
WEBLATE_EMAIL_BACKEND, 161
WEBLATE_EMAIL_HOST, 161
WEBLATE_EMAIL_HOST_PASSWORD, 161
WEBLATE_EMAIL_HOST_USER, 161
WEBLATE_EMAIL_PORT, 161
WEBLATE_EMAIL_USE_SSL, 161
WEBLATE_EMAIL_USE_TLS, 161
WEBLATE_ENABLE_HTTPS, 153
WEBLATE_GITHUB_USERNAME, 98, 154
WEBLATE_GITLAB_HOST, 154
WEBLATE_GITLAB_TOKEN, 154
WEBLATE_GITLAB_USERNAME, 99, 154
WEBLATE_GOOGLE_ANALYTICS_ID, 154
WEBLATE_GPG_IDENTITY, 155
WEBLATE_IP_PROXY_HEADER, 153
WEBLATE_LOGIN_REQUIRED_URLS_EXCEPTIONS, 154
WEBLATE_LOGLEVEL, 151
WEBLATE_MT_AWS_ACCESS_KEY_ID, 155
WEBLATE_MT_AWS_REGION, 155
WEBLATE_MT_AWS_SECRET_ACCESS_KEY, 155
WEBLATE_MT_DEEPL_API_VERSION, 155
WEBLATE_MT_DEEPL_KEY, 155
WEBLATE_MT_GLOSBE_ENABLED, 156
WEBLATE_MT_GOOGLE_KEY, 155
WEBLATE_MT_MICROSOFT_BASE_URL, 155
WEBLATE_MT_MICROSOFT_COGNITIVE_KEY, 155
WEBLATE_MT_MICROSOFT_ENDPOINT_URL, 155
WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED, 156
WEBLATE_MT_MODERNMT_KEY, 155
WEBLATE_MT_MYMEMORY_ENABLED, 156
WEBLATE_MT_SAP_BASE_URL, 156
WEBLATE_MT_SAP_PASSWORD, 156
WEBLATE_MT_SAP_SANDBOX_APIKEY, 156
WEBLATE_MT_SAP_USE_MT, 156
WEBLATE_MT_SAP_USERNAME, 156
WEBLATE_NO_EMAIL_AUTH, 159
WEBLATE_REGISTRATION_ALLOW_BACKENDS, 152
WEBLATE_REGISTRATION_OPEN, 152
WEBLATE_REMOVE_ADDONS, 162
WEBLATE_REMOVE_APPS, 162
WEBLATE_REMOVE_AUTOFIX, 162
WEBLATE_REMOVE_CHECK, 162
WEBLATE_REMOVE_LOGIN_REQUIRED_URLS_EXCEPTIONS, 154
WEBLATE_REQUIRE_LOGIN, 154
WEBLATE_SAML_IDP_ENTITY_ID, 159
WEBLATE_SAML_IDP_URL, 159
WEBLATE_SAML_IDP_X509CERT, 159
WEBLATE_SECURE_PROXY_SSL_HEADER, 153
WEBLATE_SERVER_EMAIL, 152
WEBLATE_SILENCED_SYSTEM_CHECKS, 155, 209
WEBLATE_SIMPLIFY_LANGUAGES, 154
WEBLATE_SITE_DOMAIN, 151, 187, 205, 308
WEBLATE_SITE_TITLE, 151
WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_KEY, 158
WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET, 158
WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY, 158
WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECR 158
WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENA 158

WEBLATE_SOCIAL_AUTH_BITBUCKET_KEY, 157
WEBLATE_SOCIAL_AUTH_BITBUCKET_SECRET, 157
WEBLATE_SOCIAL_AUTH_FACEBOOK_KEY, 157
WEBLATE_SOCIAL_AUTH_FACEBOOK_SECRET, 157
WEBLATE_SOCIAL_AUTH_FEDORA, 159
WEBLATE_SOCIAL_AUTH_GITHUB_KEY, 157
WEBLATE_SOCIAL_AUTH_GITHUB_SECRET, 157
WEBLATE_SOCIAL_AUTH_GITLAB_API_URL, 158
WEBLATE_SOCIAL_AUTH_GITLAB_KEY, 158
WEBLATE_SOCIAL_AUTH_GITLAB_SECRET, 158
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_KEY, 158
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET, 158
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_DOMAINS, 158
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_EMAILS, 158
WEBLATE_SOCIAL_AUTH_KEYCLOAK_ACCESS_TOKEN_URL, 158
WEBLATE_SOCIAL_AUTH_KEYCLOAK_ALGORITHM, 158
WEBLATE_SOCIAL_AUTH_KEYCLOAK_AUTHORIZATION_URL, 158
WEBLATE_SOCIAL_AUTH_KEYCLOAK_KEY, 158
WEBLATE_SOCIAL_AUTH_KEYCLOAK_PUBLIC_KEY, 158
WEBLATE_SOCIAL_AUTH_KEYCLOAK_SECRET, 158
WEBLATE_SOCIAL_AUTH_OPENSUSE, 159
WEBLATE_SOCIAL_AUTH_SLACK_KEY, 159
WEBLATE_SOCIAL_AUTH_UBUNTU, 159
WEBLATE_TIME_ZONE, 153
WEBLATE_URL_PREFIX, 155
WL_BRANCH, 285
WL_COMPONENT_NAME, 286
WL_COMPONENT_SLUG, 286
WL_COMPONENT_URL, 286
WL_ENGAGE_URL, 286
WL_FILE_FORMAT, 286
WL_FILEMASK, 286
WL_LANGUAGE, 286
WL_NEW_BASE, 286
WL_PATH, 285
WL_PREVIOUS_HEAD, 286
WL_PROJECT_NAME, 286
WL_PROJECT_SLUG, 286
WL_REPO, 285
WL_TEMPLATE, 286
WL_VCS, 285