openl10n Documentation

Release v0.1

Matthieu Moquet

November 24, 2016

Contents

OpenLocalization is a open-source localization solution. Its goal is to provide a **simple** & **flexible** tool to help you translating any of your applications.

Note: Note that this project in currently in **active development**. However, even if many new features and bug fixes are added, you should be able to use it easily. Migration scripts are provided to upgrade the application seamlessly.

CHAPTER 1

Guide

1.1 Introduction

1.1.1 Philosophy

WIP

1.1.2 Projects organisation

WIP

1.2 Installation

The OpenLocalization application is composed of a REST API for the backend based on the Symfony framework, and a Javascript web-app for the frontend.

Note: Initially the Javascript frontend was included in the main repository (openl10n/openl10n) as a BackboneJS application. However a new version of the frontend is being developped in a separate repository (openl10n/openl10n-app) using the AngularJS framework.

1.2.1 Requirements

- PHP 5.4 (or higher)
- MySQL or PostgreSQL server

Tip: The following section explains how to install the application from the source files. But you can skip it and directly grap the latest release archive (**openl10n-vX_X_X.zip**) from the release section on GitHub which contains every compiled files needed to run the app.

1.2.2 Install from the sources

First, clone the source files via git:

```
git clone https://github.com/openl10n/openl10n.git
cd openl10n
```

Then install the project's dependencies. PHP dependencies are managed via Composer. Download them using the `install` command:

```
curl -sS https://getcomposer.org/installer | php
php composer.phar install
```

Tip: You may want to install the Composer executable globally. Please refer to the official Composer documentation for more details.

For the new frontend side, you have nothing to do since the compiled scripts is already available from a CDN. But since the legacy app is being used, it's recommended to have NodeJS and npm installed locally.

CSS compilation depends on Sass (which requires Ruby).

```
gem install sass
```

Compile the assets the Gulp task manager.

```
npm install
npm install -g gulp
gulp build --prod
```

1.2.3 Setup the database

When installing PHP dependencies via the `composer install` command, a prompt should have asked you about application configuration (eg. database credentials). But you can also fill the configuration file `app/config/parameters.yml` manually (see `app/config/parameters.yml` as an example).

Once the credentials configured, you can create the database schema by running the following commands:

```
app/console doctrine:database:create --env=prod --no-debug
app/console doctrine:schema:create --env=prod --no-debug
```

You can also create a new user by running this interactive command:

```
app/console openl10n:user:new --env=prod --no-debug
```

1.2.4 Run the application

In order to access the application, access the `web/app.php` file via your web browser.

If you are using Nginx or Apache, please refer to the official Symfony documentation: Configuring a Web Server

If you want try it locally, just run the basic PHP built-in server via the `php app/console server:run` command and then access `http://localhost:8000` with your browser.

1.2.5 Deployment

WIP

1.2.6 Upgrades

WIP

1.3 Basic usage

1.3.1 Download the command-line tool

Download the CLI tool and move it to your `$PATH` (you may need to use `sudo`):

```
curl -sSL "https://cdn.openl10n.io/cli/releases/current/openl10n.phar" -o /usr/local/bin/openl10n
chmod a+x /usr/local/bin/openl10n
```

1.3.2 Configure your project

See https://github.com/openl10n/openl10n-cli/blob/master/README.md#usage

1.3.3 Synchronize your translations

WIP

CHAPTER 2

Use cases

Angular Basic example of an Angular application using the **angular-translate** directives.

Silex Example of a Silex application. The architecture of the project is inspired from the **Silex Kitchen Edition**.

Symfony Example of a typical Symfony application. This is a clone of the **Symfony Standard Edition**, plus additional bundles like the **JMSTranslationBundle** to easily extract translations from templates.

CHAPTER 3

Contributing

3.1 How to contribute

3.1.1 General guidelines

WIP

3.1.2 License

MIT

3.2 Contributing to the backend (Symfony)

3.2.1 Guidelines

WIP

3.2.2 Database model

WIP

3.2.3 Running the tests

WIP

3.2.4 Security issues

WIP

3.3 Contributing to the frontend (Angular)

3.3.1 Guidelines

WIP

3.4 Contributing to the CLI & SDK (PHP)

3.4.1 Guidelines

WIP

3.4.2 Running the tests

WIP

3.4.3 Security issues

WIP

3.5 Improving the documentation

WIP

3.6 Creating more example projects

WIP