Internationalization and localization

Overview

The goal of internationalization and localization is to allow a single web application to offer its content in languages and formats tailored to the audience.

Django has full support for translation of text, formatting of dates, times and numbers, and time zones.

Essentially, Django does two things:

- It allows developers and template authors to specify which parts of their apps should be translated or formatted for local languages and cultures.
- It uses these hooks to localize web apps for particular users according to their preferences.

Translation depends on the target language, and formatting usually depends on the target country. This information is provided by browsers in the `Accept-Language` header. However, the time zone isn't readily available.

Definitions

The words "internationalization" and "localization" often cause confusion; here's a simplified definition:

`internationalization`
: Preparing the software for localization. Usually done by developers.

`localization`
: Writing the translations and local formats. Usually done by translators.

More details can be found in the W3C Web Internationalization FAQ, the Wikipedia article or the GNU gettext documentation.

Warning: Translation is controlled by the `USE_I18N` setting. However, it involves internationalization and localization. The name of the setting is an unfortunate result of Django's history.

Here are some other terms that will help us to handle a common language:

`locale name`
: A locale name, either a language specification of the form `ll` or a combined language and country specification of the form `ll_CC`. Examples: `it`, `de_AT`, `es`, `pt_BR`, `sr_Latn`. The language part is always in lowercase. The country part is in titlecase if it has more than 2 characters, otherwise it's in uppercase. The separator is an underscore.

`language code`
: Represents the name of a language. Browsers send the names of the languages they accept in the `Accept-Language` HTTP header using this format. Examples: `it`, `de-at`, `es`, `pt-br`. Language codes are generally represented in lowercase, but the HTTP `Accept-Language` header is case-insensitive. The separator is a dash.

`message file`
: A message file is a plain-text file, representing a single language, that contains all available translation strings and how they should be represented in the given language. Message files have a `.po` file extension.

`translation string`
: A literal that can be translated.

`format file`
: A format file is a Python module that defines the data formats for a given locale.