django-robust-i18n-urls Documentation

Release 1.0.0

Karol Majta

May 21, 2015

Contents

Contents:

CHAPTER 1

Rationale

django-robust-i18n-urls: is a Django application providing sane defaul behavior for internationalized urls.

1.1 The problem

For the case of this discussion let's assume your `urlpatterns` variable contains a pattern like:

```
url(_(r'^hello/'), my_view, name='my_view')
```

If the pattern `^hello/` gets translated into Spanish `^ola/` the expected behavior would be for the controller to be accessible using both urls. Unfortunately this is not the case with Django. Urls get resolved in a context of a single locale. This way, if a user with `Accept-Language` header set to *en* tries to access **/ola/**, he will be shown an error 404 page.

1.2 The solution

django-robust-i18n-urls provides a middleware that, facing a response with `status_code` set to *404* will try to resolve the url again in context of all currently installed languages. In case of a first successful match, it will be returned instead of the error, and a matched language will be set as default for current user's session.

Some other minor issues can arise when dealing with i18n urls, an this library deals with some of them. The default workings of these helpers is detailed in Details section.

CHAPTER 2

Tutorial

2.1 Installing

You can install the most recent version directly from pypi:

```
pip install django-robust-i18n-urls
```

2.1.1 For development

If you want to develop the application yourself (or just like living on the bleeding edge) just checkout the code:

```
git checkout https://github.com/karolmajta/django-robust-i18n-urls.git
```

Install it as a code drop:

```
pip install -e .
```

Install development requirements (this will fetch Sphinx and Mock):

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

To run tests issue:

```
python setup.py test
```

2.2 Configuring

To make sure your users won't get 404 responses when using urls for locales other than reported by their browser just modify your `MIDDLEWARE_CLASSES` setting, by adding `robust_urls.middleware.RobustI18nLocaleMiddleware`.

If you plan on providing users with an url for changing their current language just inlcude in your `urls.py`:

```
import robust_urls.urls

# ...

urlpatterns += patterns(url(r'/i18n/', include(robust_urls.urls)))
```

CHAPTER 3

Details

While *django-robust-i18n-urls* does not expose any particular APIs to end users (it's designed as a plug&play app) it won't hurt to know what is going on under the hood.

3.1 robust_urls.urls

This module contains *urlpatterns* variable that can be used as a drop-in replacement for *urplatterns* contained in *django.conf.urls.i18n*.

3.2 robust_urls.views

This module contains a single view that is used to change locale of current user, and can be used as a drop-in replacement for *set_language* view from *django.views.i18n.set_language*

3.2.1 set_language

set_language view works in a fasion similar to Django's original one. The key differences are:

- It does not allow GET requests, they will result in a 405 response. Only POST method is allowed.
- If a URL path specified in *request.REQUEST['next']* can be matched against a specific view, instead of issuing an immediate redirect, the *set_language* will first reverse the match to obtain a request path in language of user's choosing.
- If a URL path specified in *request.REQUEST['next']* cannot be matched to any view, a redirection will be issued to it anyway.

3.3 robust_urls.middleware

This module containse the *RobustI18nLocaleMiddleware* that, next to *robust_urls.view.set_language* is the app's main component.

3.3.1 RobustI18nLocaleMiddleware

This middleware's *process_response* method will touch only responses with *status_code* 404. It will try to match the URL using languages in order specified in *LANGUAGES* setting. If a match is not found, the response is returned unchanged. If a match is found, the result of rendering the view's response is returned instead. This method also takes care to set proper (matched) locale in user's session or language cookie.

3.4 robust_urls.utils

3.4.1 locale_context

This context manager will execute given block making sure that language provided as argument is active during execution. On exit will call *translation.deactivate*.

```
with locale_context('pl_PL'):
    print _('Good Morning')  # will print 'Dzień Dobry'
```

Warning! 'locale_context' calls 'transaction.deactivate' so it is ill suited for use inside views. In future versions it will probably hold to a locale used before the manager was endered, and activate it on exit instead of calling 'transaction.deactivate'. This api will change!

3.4.2 try_uri_for_language

A simple helper that takes *path*, *language* and *resolver* arguments. Will try to resolve path using resolver, in context of given language. If no match is found will return *None* instead of raising an exception. If match is found will return whatever resolver returns (*ResolverMatch* instance).

Indices and tables

- genindex
- modindex
- search