pyramid_localizeDocumentation

Release 0.1.0

Grzegorz Śliwiński

Nov 28, 2018

Contents

pyramid_localize is a pyramid extension providing ready-to-use, translation functionality for your pyramid based projects. All you need is to add Babel, and add some configuration. localize provides also a web interface that allows you to reload translations live, without reloading application.

CHAPTER 1

Package status

CHAPTER 2

Package resources

- Bug tracker: https://github.com/fizyk/pyramid_localize/issues
- Documentation: http://pyramid-localize.readthedocs.org/
- PyPI: https://pypi.python.org/pypi/pyramid_localize

CHAPTER 3

Instalation:

```
pip install pyramid_localize
```

or add **pyramid_localize** to your **setup.py** requirements.

CHAPTER 4

Tests:

You'll need: packages defined in extra_requires[tests] to run tests, and then:

```
python setup.py nosetests
```

4.1 Contents

4.1.1 Basic usage

Installation

To install pyramid_localize, run:

```
pip install pyramid_localize
```

or add **pyramid_localize** to your **setup.py** requirements.

Include in your project

To use this plugin, simply include it into your configurator object:

```
config.include('pyramid_localize')
```

pyramid_localize will add translation methods both to request object and for template.

Using without configuration

Basic usage won't require any additional plugin's configuration. You just include the plugin, and make sure you have a Babel installed in the same environment. You can either install *babel* on your own, or add dependency to **pyramid_localize[babel]**. This is introduced, to allow creating code, that both works with translations and without, see *Fake translation support*.

However in this case, all translation configuration would need to be done within Your application, as described in chapter Internationalization and Localization of pyramid's documentation.

Configuration

Note: Plugins uses tzf.pyramid_yml for its configuration settings

This is full usage example, where **pyramid_localize** provides everything needed for your application, including *locale_negotiator()*, and sqlalchemy model for *Language*, to be able to store localized data in database.

```
localize:
    pybabel: pybabel # pybabel's bin localisation. Used to call compile and extract␣
↪commands
    domain: APP_DOMAIN  # your application domain name
    locales:    # available and default locale for your app
        available: [en, de, pl]
        default: en
    translation:
        # directory, where translatons can be found, might be a list,
        # defaults to empty list
        dirs: 'roots.app:resources/locale'
        # destination, where .po and .mo files will be created
        # can be same format as pyramid's asset path,
        # it's also added to dirs, as translation source
        destination: 'app:resources/locale'
        # sources, where translations can be found domain: localisation
        # can be same format as pyramid's asset path
        sources:
            APP_DOMAIN: 'app:resources/locale/'
            PACKAGE: 'package.subpackage:resources/locale/'
```

Having configured this, your app, and all subpackages will be fully localized, you'll also have the ability to automatically reload translations without having to restart application. See *Web API*.

Fake translation support

In order to be able to allow application creation, that will be usable with translated applications, and those without translations, pyramid_localize has simple switch, which detects presence of babel.

If Babel will not be found, then pyramid_localize will install dummy translation methods, that will do nothing, except placing translation string args in place, after getting all the arguments, so You can still create apps, or pyramid plugins using translation functionality.

Code behind that can be seen here *pyramid_localize.tools.dummy_autotranslate()*.

4.1.2 Advanced

Localized URLs

pyramid_localize allows to include locale definition inside routes. In order to utilise this functionality, your request factory, have to inherit *LocalizeRequestMixin*, and your routes, you wish to localise should include a _LOCALE_ parameter. You can pass it to route_url, but if you do not, this mixin will set it to current locale, making

sure, it's always set. *locale_negotiator()* set by pyramid_localize will take route locale in before any other possibility.

Web API

Web API is a collection of actions used to work on translations files.

At the moment, it includes loading them into application, creating /updating catalogues for all available translations, compiling, and reloading translations without the need to restart application.

Lists of actions per their route names:

- **localize:index** action, lists all configured translation domains, and it's file along with data such as modification date. See *index()*.
- **localize:update** action, updates all .po files from respective .pot's, and if needed initializes them. See *update_catalogue()*.
- **localize:compile** action, compiles all .po translation files into .mo used by gettext to serve translations on your site. See *compile_catalogue()*.
- **localize:reload** action is more like an example action, but still usable. Reloads translation catalogues for application. See *reload_catalogue()*.

4.1.3 API

models

Language model.

class pyramid_localize.models.**Language**(***kwargs*)
Language table model definition.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

pyramid_localize.models.**before_language_insert**(*_*, *__*, *language*)
Set name and native_name before creation.

negotiator

Locale negotiator.

pyramid_localize.negotiator.**locale_negotiator**(*request*)
Locale negotiator.

It sets best suited locale variable for given user:

1. Check for presence and value of **request._LOCALE_** value
2. Then tries the address url, if the first part has locale indicator.
3. It checks cookies, for value set here
4. **Tries to best match accepted language for browser user is visiting** website with

5. Defaults to **localize.locales.default** configuration setting value

Parameters **request** (*pyramid.request.Request*) – a request object

Returns locale name

Return type str

request

Request related code.

pyramid_localize.request.**locale_id**(*request*)
Return database id of a current locale name.

Returns database id of a language code needed for translations

Return type int

pyramid_localize.request.**database_locales**(*request*)
Return all database locales available.

Returns dictionary of Language objects language_code: Language

Return type dict

pyramid_localize.request.**locales**(*request*, *config=False*)
Return locales.

Parameters **config** (*bool*) – Whether to restrict list with config

Returns dictionary of Language objects language_code: Language

Return type dict

LocalizeRequestMixin

class pyramid_localize.request.**LocalizeRequestMixin**
Mixin adding overwriting Request methods.

default_locale(***kw*)
Set up default locale for path kwargs.

Can be used in custom route_url overwrites.
Parameters **kw** (*kwargs*) – list of route parts
Returns kw

route_url(*route_name*, **elements*, ***kw*)
Overwrite original route_url to handle default locale within route.

Note: see pyramid.request.Request.route_url()

---

tools

methods in this module are tools, thank to which pyramid_localize works most of its magic.

pyramid_localize.tools.**destination_path**(*request*)
Return absolute path of the translation destination.

Parameters **request** (*pyramid.request.Request*) – a request object

Returns A combined translation destination path

Return type str

pyramid_localize.tools.**dummy_autotranslate**(*msgid*, *domain=None*, *default=None*, *mapping=None*)

Simulate autotranslate.

Parameters

- **msgid** (*str*) – Message or message id
- **domain** (*str*) – Translation domain
- **default** (*str*) – Default message
- **mapping** (*dict*) – Mapping dictionary for message variables

Returns *translated* string

Return type str

pyramid_localize.tools.**set_localizer**(*request*, *reset=False*)

Set localizer and auto_translate methods for request.

Parameters

- **request** (*pyramid.request.Request*) – request object
- **reset** (*bool*) – flag that directs resetting localizer within app

views

catalogue

Catalogue view.

class pyramid_localize.views.catalogue.**CatalogueView**(*request*)

View class for catalogue manipulation actions.

Assign request.

Parameters **request** (*pyramid.request.Request*) –

compile_catalogue()

Compile all translation files.

For every language defined compile .po files into into .mo file that's used by gettext.

Redirects to **localize:index**.

index()

List domains, and its files of files with metadata.

Returns

```
{
    'language': {
        'domain1': {
            'po': 'modification time',
            'pot': 'modification time',
            'mo': 'modification time',
```

(continues on next page)

(continued from previous page)

```
            },
        # more domains
    },
    # more languages
}
```

reload_catalogue()
Reload translation catalogue for application it's run in.

Note: To see how is this happening, you might want to see *set_localizer()*

Returns

Only for xhr requests:

```
{
    'status': True,
    'msg': 'Localizators has been reloaded' # translated
}
```

non xhr requests: Redirects to **localize:index**.

update_catalogue()
Update or initialize translation catalogues.

Create (.po files) for each language/catalogue from their respective translation templates (.pot). This action is performed for every language defined within *localize.locales.available* config key.

Redirects itself to **localize:index**.

4.1.4 Contribute to pyramid_localize

Thank you for taking time to contribute to pyramid_localize!

The following is a set of guidelines for contributing to pyramid_localize. These are just guidelines, not rules, use your best judgment and feel free to propose changes to this document in a pull request.

Bug Reports

1. Use a clear and descriptive title for the issue - it'll be much easier to identify the problem.
2. Describe the steps to reproduce the problems in as many details as possible.
3. If possible, provide a code snippet to reproduce the issue.

Feature requests/proposals

1. Use a clear and descriptive title for the proposal
2. **Provide as detailed description as possible**
    - Use case is great to have

3. There'll be a bit of discussion for the feature. Don't worry, if it is to be accepted, we'd like to support it, so we need to understand it thoroughly.

Pull requests

1. Start with a bug report or feature request
2. Use a clear and descriptive title
3. Provide a description - which issue does it refers to, and what part of the issue is being solved
4. Be ready for code review :)

Commits

1. Make sure commits are atomic, and each atomic change is being followed by test.
2. If the commit solves part of the issue reported, include *refs #[Issue number]* in a commit message.
3. If the commit solves whole issue reported, please refer to Closing issues via commit messages for ways to close issues when commits will be merged.

Coding style

1. All python coding style are being enforced by Pylama and configured in pylama.ini file.
2. Additional, not always mandatory checks are being performed by QuantifiedCode

4.1.5 CHANGES

unreleased

- [functionality] Language object's name and native_name will filled automatically on language creation from pycountry [rmed]

0.1.0

- weaker pyramid_yml requirements. Use `registry['config']` instead of `request.config` which gets added only when explicitly including tzf.pyramid_yml package.
- deprecated request.locale in favour of request.locale_name delivered by Pyramid 1.5
- moved locale negotiator into it's own submodule

backward incompatible

- required cookie name changed to _LOCALE_ to be consistent with other places
- fixed a typo from catalog to catalogue

tests

- refactor tests to pytest
- **introduced pylama checks for:**
  - pep8
  - pyflakes
  - pep257
  - mccabe
- license information
- requires at least pyramid 1.5a1 (rely on default localizer reify method)
- py3 compatibility (require at least babel 1.0)
- locale negotiator looks first for request attribute _LOCALE_
- added pyramid_mako dependency (required by pyramid 1.5a2 changes)

0.0.5

- fixes in catalog/index template [zusel, fizyk]
- destination path added in translation_dirs as a translation source as well [fizyk]

0.0.4

- fix issue with translation files path beeing not related to cwd [fizyk]

0.0.2

- fixed MANIFEST.in [fizyk]

0.0.1

- initial release [fizyk]

4.2 License

Python Module Index

p

Index