tg-react Documentation

Release 2.1.0

Thorgate

May 13, 2019

Contents

Helpers for react based applications running on django.

Contents:

CHAPTER 1

Getting started

1.1 Installation

Install tg-react with pip:

```
pip install tg-react
```

Then use it in your project:

```
import tg_react
```

1.2 Features

- TODO

CHAPTER 2

Recipes

2.1 Adding a new language locally

If you're translating a new language you'll need to translate the existing tg-react messages:

1. Make a new folder where you want to store the internationalization resources. Add this path to your LOCALE_PATHS setting.
2. Now create a subfolder for the language you want to translate. The folder should be named using locale name notation. For example: de, pt_BR, es_AR.
3. Now copy the base translations file from the tg-react source code into your translations folder.
4. Edit the django.po file you've just copied, translating all the messages.
5. Run manage.py compilemessages -l pt_BR to make the translations available for Django to use. You should see a message like processing file django.po in <. . . >/locale/pt_BR/LC_MESSAGES.
6. Restart your development server to see the changes take effect.

2.2 Changing the built-in translations

Follow the process described in *Adding a new language locally* but instead of creating a new directory use the name of an existing one.

CHAPTER 3

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

3.1 Types of Contributions

3.1.1 Report Bugs

Report bugs at https://github.com/thorgate/tg-react/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

3.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

3.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

3.1.4 Write Documentation

tg-react could always use more documentation, whether as part of the official tg-react docs, in docstrings, or even on the web in blog posts, articles, and such.

3.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/thorgate/tg-react/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.1.6 Translate

Add a new locale using:

```
$ LOCALE='<locale>' make add-locale
```

This will create a new directory under *tg_react/locale/<locale>* and a django.po file inside it. First edit the comments and the PO file header of the generated file (use tg_react/locale/en/LC_MESSAGES/django.po for refrence) and then use tools like Poedit to add translations.

After you are done, update compiled translations via:

```
$ make update-messages
```

3.2 Get Started!

Ready to contribute? Here's how to set up *tg-react* for local development.

1. Fork the *tg-react* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/tg-react.git
   ```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv tg-react
   $ cd tg-react/
   $ python setup.py develop
   ```

4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ make lint
$ make test
$ make test-all
```

   To get flake8 and tox, just pip install them into your virtualenv:

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

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

3.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst. You should also update the documentation source files via:

```
$ make docs
```

3. If the pull request modifies/adds translations don't forget to run:

```
$ make update-messages
```

4. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6. Check https://travis-ci.org/thorgate/tg-react/pull_requests and make sure that the tests pass for all supported Python versions.

3.4 Tips

Run full test suite via tox (all python and django version combinations):

```
$ make test-all
```

To run a subset of tests:

```
$ py.test tests.test_tg_react
```

Update documentation source files and generate it:

```
$ make docs
```

To see all make commands:

```
$ make help
```

CHAPTER 4

Modules

4.1 tg_react package

4.1.1 Subpackages

tg_react.api package

Subpackages

tg_react.api.accounts package

Submodules

tg_react.api.accounts.serializers module

tg_react.api.accounts.urls module

tg_react.api.accounts.views module

Module contents

Module contents

tg_react.management package

Subpackages

tg_react.management.commands package

Submodules

tg_react.management.commands.webpack_constants module

class tg_react.management.commands.webpack_constants.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: django.core.management.base.BaseCommand

add_arguments(*parser*)

handle(**args*, ***options*)

help = 'Output all configured constants as json'

Module contents

Module contents

4.1.2 Submodules

4.1.3 tg_react.apiurls module

tg_react.apiurls.**flatten_patterns**(*urlconf*, *base_path=None*, *namespace=None*)

tg_react.apiurls.**flatten_urls**(*module_path*, *base_path*)

tg_react.apiurls.**get_url_regex_pattern**(*urlpattern*)

tg_react.apiurls.**to_camelcase**(*value*)

tg_react.apiurls.**tokenize_pattern**(*regex*)

tg_react.apiurls.**ucfirst**(*word*)

4.1.4 tg_react.apps module

class tg_react.apps.**TgReactConfig**(*app_name*, *app_module*)

Bases: django.apps.config.AppConfig

name = 'tg_react'

verbose_name = 'Tg react'

4.1.5 tg_react.catalogue_legacy module

4.1.6 tg_react.language module

class tg_react.language.**DjangoLocaleData**(*domain=None*, *packages=None*)

Bases: object

collect_translations()

Collect all *domain* translations and return *Tuple[languages, locale_data]*

domain = 'djangojs'

get_catalog(*locale*)
Create Django translation catalogue for *locale*.

classmethod get_catalogue_header_value(*catalog*, *key*)
Get *.po* header value.

classmethod get_paths(*packages*)
Create list of matching packages for translation engine.

classmethod get_plural(*catalog*)
Special handling for plural forms.

languages = (('en', 'English'), ('et', 'Estonian'), ('ru', 'Russian'))

make_header(*locale*, *catalog*)
Populate header with correct data from top-most locale file.

packages = None

tg_react.language.**constants**(*context*)

4.1.7 tg_react.middleware module

class tg_react.middleware.**LocaleMiddleware**
Bases: object

This is a very simple middleware that parses a request and decides what translation object to install in the current thread context depending on the selected language.

This also allows us to update the language cookie whenever our api endpoint is used.

get_language_for_user(*request*)

process_request(*request*)

process_response(*request*, *response*)

4.1.8 tg_react.models module

4.1.9 tg_react.routers module

4.1.10 tg_react.settings module

tg_react.settings.**configure**()

tg_react.settings.**exclude_fields_from_user_details**()

tg_react.settings.**get_email_case_sensitive**()

tg_react.settings.**get_password_recovery_url**()

tg_react.settings.**get_post_login_handler**()

tg_react.settings.**get_post_logout_handler**()

tg_react.settings.**get_signup_skipped_fields**()

tg_react.settings.**get_static_dir**()

tg_react.settings.**get_user_signup_fields**()

4.1.11 tg_react.webpack module

class tg_react.webpack.**WebpackConstants**
Bases: object

classmethod collect()
Load all constant generators from settings.WEBPACK_CONSTANT_PROCESSORS and concat their values.

classmethod get_constant_processors()

tg_react.webpack.**default_constants**(*context*)

4.1.12 Module contents

CHAPTER 5

Changelog

Changes are documented under Github Releases.

Python Module Index

t

Index

T

U

V

W