django-bidi-utils Documentation

Release 1.0

Meir Kriheli

October 18, 2013

CONTENTS

Contents:

CHAPTER

ONE

DJANGO-BIDI-UTILS

Provides context processors and filters for handling Bi-directional (BiDi) in django templates, adding some needed functionality to Django's LANGUAGE_BIDI template var.

1.1 Documentation

The full documentation is at http://django-bidi-utils.rtfd.org.

1.2 Quickstart

Install django-bidi-utils:

```
pip install django-bidi-utils
```

To use it in a Django project add *bidiutils* the project's INSTALLED_APPS setting:

```
INSTALLED_APPS = (
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sites",
    ...
    "bidiutils",
    ...
)
```

To enable the context processor, add it to TEMPLATE_CONTEXT_PROCESSORS settings:

```
TEMPLATE_CONTEXT_PROCESSORS = (
    "django.core.context_processors.auth",
    "django.core.context_processors.debug",
    "django.core.context_processors.i18n",
    "django.core.context_processors.media",
    ...
    "bidiutils.context_processors.bidi",
)
```

1.3 Features

- Context processor adding to templates passed a RequestContext BiDi releated variables.

- *add_direction* template filter, for adding direction to media resource (images, stylesheets, etc)

CHAPTER

TWO

INSTALLATION

At the command line:

```
$ easy_install django-bidi-utils
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-bidi-utils
$ pip install django-bidi-utils
```

CHAPTER

THREE

USAGE

3.1 Settings

To use it in a Django project add *bidiutils* the project's INSTALLED_APPS setting:

```
INSTALLED_APPS = (
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sites",
    ...
    "bidiutils",
    ...
)
```

To enable the context processor, add it to TEMPLATE_CONTEXT_PROCESSORS settings:

```
TEMPLATE_CONTEXT_PROCESSORS = (
    "django.core.context_processors.auth",
    "django.core.context_processors.debug",
    "django.core.context_processors.i18n",
    "django.core.context_processors.media",
    ...
    "bidiutils.context_processors.bidi",
)
```

3.2 Template context variables

If you've added *bidiutils.context_processors.bidi* to *TEMPLATE_CONTEXT_PROCESSORS* settings, you'll have additional variables in the template context for templates using a RequestContext:

LANGUAGE_DIRECTION Direction of current language (*ltr* or *rtl*). The following common idiom:

```
{% load i18n %}<html dir="{{ LANGUAGE_BIDI|yesno:"rtl,ltr" }}">
```

Can be re-written as:

```
{% load i18n %}<html dir="{{LANGUAGE_DIRECTION}}">
```

LANGUAGE_START Start of language layout (*right* for RTL, *left* for LTR). Useful in cases you want to align elements with inline style etc:

```
<div style="float:{{LANGUAGE_START}}">
```

LANGUAGE_END End of language layout (*left* for RTL, *right* for LTR). Just like *LANGUAGE_START*, but denotes the other side. A sample usage with Bootstrap's glyphicons to display an arrow going from language start to end:

```
<span class="glyphicon glyphicon-arrow-{{ LANGUAGE_END }}"></span>
```

LANGUAGE_MARKER Language marker entity (*‏* for RTL, *‎* for LTR). Those marker are inserted into a location to make an enclosed weak character inherit its writing direction.

Those are important to keep the flow correct in case of a weak directional character between to elements of the other direction. Take this template fragment for example:

```
{{ user1.full_name }}: {{post1.title}}
{{ user2.full_name }}: {{post2.title}}
```

assuming your current language's direction is RTL, and user2's full name and post title are in LTR (e.g: both in English) while user one's are RTL The result will be rendered as (using UPPERCASE without actual content do demonstrate):

```
USER1_POST_TITLE  :USER1_FULLNAME
USER2_FULLNAME: USER2_POST_TITLE
```

This is because the ':' is weak character inheriting the direction for surrounding element(s). Inserting the current language marker ensures the correct layout

```
{{ user1.full_name }}{{LANGUAGE_MARKER}}: {{post1.title}}
{{ user2.full_name }}{{LANGUAGE_MARKER}}: {{post2.title}}
```

For more information see Bi-Directional text - Unicode support.

3.3 *add_direction* filter

Adds direction to the element. Make sure to load *bidiutils_tags* to use the filter which accepts a single, optional, argument:

rtl_only Add the direction only in case of a right-to-left language. The default if no argument is passed.

both Add the direction in both cases

ltr_only Add the direction only in case of a left-to-right language

Assuming the current language is RTL, and *image* in the context is *"arrow.png"*:

```
{% load bidiutils_tags %}

<img src="{{image|add_direction}}" alt="arrow">
<img src="{{image|add_direction:"rtl_only"}}" alt="arrow">
<img src="{{image|add_direction:"both"}}" alt="arrow">
<img src="{{image|add_direction:"ltr_only"}}" alt="arrow">
```

The rendered result will be:

```
<img src="/media/img/arrow_rtl.png" alt="arrow">
<img src="/media/img/arrow_rtl.png" alt="arrow">
<img src="/media/img/arrow_rtl.png" alt="arrow">
<img src="/media/img/arrow.png" alt="arrow">
```

On the other hand, Assuming the current language is LTR, and *image* in the context is *"arrow.png"*, the rendered result of the above template would be:

```
<img src="/media/img/arrow.png" alt="arrow">
<img src="/media/img/arrow.png" alt="arrow">
<img src="/media/img/arrow_ltr.png" alt="arrow">
<img src="/media/img/arrow_ltr.png" alt="arrow">
```

CHAPTER

FOUR

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/MeirKriheli/django-bidi-utils/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

django-bidi-utils could always use more documentation, whether as part of the official django-bidi-utils docs, in doc-strings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/MeirKriheli/django-bidi-utils/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 :)

4.2 Get Started!

Ready to contribute? Here's how to set up *django-bidi-utils* for local development.

1. Fork the *django-bidi-utils* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/django-bidi-utils.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 django-bidi-utils
   $ cd django-bidi-utils/
   $ 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:

```
$ flake8 bidiutils tests
    $ python setup.py test
$ tox
```

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

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.

4.3 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.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/MeirKriheli/django-bidi-utils/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_bidiutils
```

CHAPTER

FIVE

CREDITS

5.1 Development Lead

- Meir Kriheli <mkriheli@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER

SIX

HISTORY

6.1 1.0 (2013-10-19)

- Testing for Python 3.3, 2.7 and 2.6 Compatibility.
- Removed buildout.
- Re-Based on Cookiecutter.
- Sphinx for docs, added more examples.

6.2 0.2.1 (2009-05-18)

- Forgot CHANGELOG.rst in MANIFEST.in.

6.3 0.2 (2009-05-17)

- Added template filter *add_direction*.
- Removed *README*, add MAINFEST.in which includes *README.rst*.

6.4 0.1 (2009-04-28)

- Initial release, context processors.