Flask-Menu Documentation

Release 0.7.0

CERN

Dec 12, 2017

Contents

Flask-Menu is a Flask extension that adds support for generating menus.

CHAPTER 1

Contents

1.1 Installation

Flask-Menu is on PyPI so all you need is:

```
$ pip install Flask-Menu
```

The development version can be downloaded from its page at GitHub.

```
$ git clone https://github.com/inveniosoftware/flask-menu.git
$ cd flask-menu
$ python setup.py develop
$ ./run-tests.sh
```

1.1.1 Requirements

Flask-Menu has the following dependencies:

- Flask
- six

Flask-Menu requires Python version 2.7 or 3.3+.

1.2 Usage

This guide assumes that you have successfully installed `Flask-Menu` package already. If not, please follow the *Installation* instructions first.

1.2.1 Simple Example

Here is a simple Flask-Menu usage example:

```
from flask import Flask
from flask import render_template_string
from flask_menu import Menu, register_menu

app = Flask(__name__)
Menu(app=app)

def tmpl_show_menu():
    return render_template_string(
        """
        {%- for item in current_menu.children %}
            {% if item.active %}*{% endif %}{{ item.text }}
        {% endfor -%}
        """)

@app.route('/')
@register_menu(app, '.', 'Home')
def index():
    return tmpl_show_menu()

@app.route('/first')
@register_menu(app, '.first', 'First', order=0)
def first():
    return tmpl_show_menu()

@app.route('/second')
@register_menu(app, '.second', 'Second', order=1)
def second():
    return tmpl_show_menu()

if __name__ == '__main__':
    app.run(debug=True)
```

If you save the above as `app.py`, you can run the example application using your Python interpreter:

```
$ python app.py
 * Running on http://127.0.0.1:5000/
```

and you can observe generated menu on the example pages:

```
$ firefox http://127.0.0.1:5000/
$ firefox http://127.0.0.1:5000/first
$ firefox http://127.0.0.1:5000/second
```

You should now be able to emulate this example in your own Flask applications. For more information, please read the *Templating* guide, the *Blueprint Support* guide, and peruse the *API*.

1.3 Templating

By default, a proxy object to *current_menu* is added to your Jinja2 context as *current_menu* to help you with creating navigation bar. For example:

```
<ul>
  {%- for item in current_menu.children recursive -%}
  <li>
    <a href="{{ item.url }}">{{ item.text }}</a>
    {%- if item.children -%}
    <ul>
      {{ loop(item.children) }}
    </ul>
    {%- endif -%}
  </li>
  {%- endfor -%}
</ul>
```

1.4 Blueprint Support

The most import part of an modular Flask application is Blueprint. You can create one for your application somewhere in your code and decorate your view function, like this:

```
from flask import Blueprint
from flask_menu import register_menu

bp_account = Blueprint('account', __name__, url_prefix='/account')

@bp_account.route('/')
@register_menu(bp_account, '.account', 'Your account')
def index():
    pass
```

Sometimes you want to combine multiple blueprints and organize the navigation to certain hierarchy.

```
from flask import Blueprint
from flask_menu import register_menu

bp_social = Blueprint('social', __name__, url_prefix='/social')

@bp_account.route('/list')
@register_menu(bp_social, '.account.list', 'Social networks')
def list():
    pass
```

As a result of this, your *current_menu* object will contain a list with 3 items while processing a request for */social/list*.

```
>>> from example import app
>>> from flask_menu import current_menu
>>> import account
>>> import social
>>> app.register_blueprint(account.bp_account)
>>> app.register_blueprint(social.bp_social)
>>> with app.test_client() as c:
...     c.get('/social/list')
...     assert current_menu.submenu('account.list').active
...     current_menu.children
```

1.5 Flask-Classy

Flask-Classy is a library commonly used in Flask development and gives additional structure to apps which already make use of blueprints as well as apps which do not use blueprints.

Using Flask-Menu with Flask-Classy is rather simple:

```
from flask_classy import FlaskView
from flask_menu.classy import classy_menu_item

class MyEndpoint(FlaskView):
    route_base = '/'

    @classy_menu_item('frontend.account', 'Home', order=0)
    def index(self):
        # Do something.
        pass
```

Instead of using the *@menu.register_menu* decorator, we use classy_menu_item. All usage is otherwise the same to *register_menu*, however you do not need to provide reference to the blueprint/app.

You do have to register the entire class with flask-menu at runtime however.

```
from MyEndpoint import MyEndpoint
from flask import Blueprint
from flask_menu.classy import register_flaskview

bp = Blueprint('bp', __name__)

MyEndpoint.register(bp)
register_flaskview(bp, MyEndpoint)
```

1.6 API

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

1.6.1 Flask extension

class `flask_menu.`**`Menu`**(*app=None*)

Flask extension implementation.

init_app(*app*)
Initialize a Flask application.

static root()
Return a root entry of current application's menu.

class flask_menu.**MenuEntryMixin**(*name*, *parent*)
Represent a entry node in the menu tree.

Provides information for displaying links (text, url, visible, active). Navigate the hierarchy using *children()* and *submenu()*.

active
Return True if the menu item is active.

active_item
Return the active item from the menu's tree.

Return self if the item itself is active. Return an active child if there is one. If there are no active menu items, None will be returned.

children
Return list of sorted children.

dynamic_list
Return list from dynamic list constructor.

has_active_child(*recursive=True*)
Return True if the menu has an active child.

has_visible_child(*recursive=True*)
Return True if the menu has a visible child.

hide()
Make the entry always hidden.

list_path(*from_path*, *to_path*)
Return all items on path between two specified entries.

Only if one of them is an ancestor of the other.

Parameters

- **from_path** – The ancestor entry.
- **to_path** – The child entry.

Returns List of entries between those items or None if they are on different branches.

register(*endpoint=None*, *text=None*, *order=0*, *external_url=None*, *endpoint_arguments_constructor=None*, *dynamic_list_constructor=None*, *active_when=None*, *visible_when=None*, *expected_args=None*, ***kwargs*)
Assign endpoint and display values.

New in version 0.6.0: The *external_url* parameter is mutually exclusive with *endpoint*.

submenu(*path*, *auto_create=True*)
Return submenu placed at the given path in the hierarchy.

If it does not exist, a new one is created. Return None if path string is invalid.

Parameters

- **path** – Path to submenu as a string 'qua.bua.cua'
- **auto_create** – If True, missing entries will be created to satisfy the given path.

Returns Submenu placed at the given path in the hierarchy.

url
Generate url from given endpoint and optional dynamic arguments.

visible
Return True if the menu item is visible.

1.6.2 Decorators

flask_menu.**register_menu**(*app*, *path*, *text*, *order=0*, *endpoint_arguments_constructor=None*, *dynamic_list_constructor=None*, *active_when=None*, *visible_when=None*, ***kwargs*)
Decorate endpoints that should be displayed in a menu.

Example:

```
@register_menu(app, '.', _('Home'))
def index():
    pass
```

Parameters

- **app** – Application or Blueprint which owns the function view.
- **path** – Path to this item in menu hierarchy, for example 'main.category.item'. Path can be an object with custom __str__ method: it will be converted on first request, therefore you can use current_app inside this __str__ method.
- **text** – Text displayed as link.
- **order** – Index of item among other items in the same menu.
- **endpoint_arguments_constructor** – Function returning dict of arguments passed to url_for when creating the link.
- **active_when** – Function returning True when the item should be displayed as active.
- **visible_when** – Function returning True when this item should be displayed.
- **dynamic_list_constructor** – Function returning a list of entries to be displayed by this item. Every object should have 'text' and 'url' properties/dict elements. This property will not be directly affect the menu system, but allows other systems to use it while rendering.
- **kwargs** – Additional arguments will be available as attributes on registered *MenuEntryMixin* instance.

Changed in version 0.2.0: The *kwargs* arguments.

1.6.3 Proxies

flask_menu.**current_menu**
Root of a menu item.

1.6.4 Flask-Classy

flask_menu.classy.**register_flaskview**(*app*, *classy_view*)
Register a Flask-Classy FlaskView's menu items with the menu register.

Example:

```
bp = Blueprint('bp', __name__)
menu.register_flaskview(bp, MyEndpoint)
```

Parameters

- **app** – Application or Blueprint which owns the function view.
- **classy_view** – The Flask-Classy FlaskView class to register menu items for.

flask_menu.classy.**classy_menu_item**(*path*, *text*, ***kwargs*)
Decorator to register an endpoint within a Flask-Classy class.

All usage is otherwise the same to *register_menu*, however you do not need to provide reference to the blueprint/app.

Example:

```
class MyEndpoint(FlaskView):
    route_base = '/'

    @menu.classy_menu_item('frontend.account', 'Home', order=0)
    def index(self):
        # Do something.
        pass
```

Parameters

- **path** – Path to this item in menu hierarchy, for example 'main.category.item'. Path can be an object with custom __str__ method: it will be converted on first request, therefore you can use current_app inside this __str__ method.
- **text** – Text displayed as link.
- **order** – Index of item among other items in the same menu.
- **endpoint_arguments_constructor** – Function returning dict of arguments passed to url_for when creating the link.
- **active_when** – Function returning True when the item should be displayed as active.
- **visible_when** – Function returning True when this item should be displayed.
- **dynamic_list_constructor** – Function returning a list of entries to be displayed by this item. Every object should have 'text' and 'url' properties/dict elements. This property will not be directly affect the menu system, but allows other systems to use it while rendering.
- **kwargs** – Additional arguments will be available as attributes on registered *flask_menu.MenuEntryMixin* instance.

Changed in version 0.2.0: The *kwargs* arguments.

1.7 Changes

CHAPTER 2

Version 0.7.0 (released 2017-12-12)

- Uses whole segment of the URL instead of a prefix to determine active menu item. Currently the menu item is marked as active when there is a prefix match on the URL. This creates situations where multiple different menu items appear to be active just because they share a prefix. (#62)

CHAPTER 3

Version 0.6.0 (released 2017-08-03)

- Fixes Python 3 deprecation warnings.
- Adds the *external_url* parameter to MenuEntryMixin's *register* function, allowing menu items with external urls not tied to an endpoint.

CHAPTER 4

Version 0.5.1 (released 2016-01-04)

- Improves tests for checking when an item is active.

CHAPTER 5

Version 0.5.0 (released 2015-10-30)

- Drops support for Python 2.6.
- Adds new property to MenuEntryMixin which allows the user to retrieve the current active item from the MenuEntryMixin's tree. (#43)
- Modifies project structure to be in line with other newer Invenio project packages. This includes renaming files to match with files in other projects, revising structures of certain files and adding more tools for testing. (#42)
- Fixes incompatibility with pytest>=2.8.0 which removed the method consider_setuptools_entrypoints(). (#41)
- Updates to the new standard greeting phrase

CHAPTER 6

Version 0.4.0 (released 2015-07-23)

- Flask-Classy support and automatic detection of parameters for *url_for*. (#33)
- Improves how the default active state of items is determined. (#32)
- Adds *.dockerignore* excluding among others Python cache files. This solves a problem when using both *tox* and *docker* to run the test suite on the same host. (#29)

CHAPTER 7

Version 0.3.0 (released 2015-03-17)

- New method *has_active_child(recursive=True)* in *MenuEntryMixin*. (#25)
- Fixed documentation of blueprint example. (#21)
- Configuration for Docker and demo app. (#22 #29)
- Fixed template example and added code block types. (#14)

CHAPTER 8

Version 0.2.0 (released 2014-11-04)

- The Flask-Menu extension is now released under more permissive Revised BSD License. (#12)
- New support for additional keyword arguments stored as *MenuItem* attributes. (#19)
- Richer quick-start usage example. (#16)
- Support for Python 3.4. (#6)
- Documentation improvements. (#3)

CHAPTER 9

Version 0.1.0 (released 2014-06-27)

- Initial public release.

9.1 Contributing

Bug reports, feature requests, and other contributions are welcome. If you find a demonstrable problem that is caused by the code of this library, please:

1. Search for already reported problems.
2. Check if the issue has been fixed or is still reproducible on the latest *master* branch.
3. Create an issue with **a test case**.

If you create a feature branch, you can run the tests to ensure everything is operating correctly:

```
$ ./run-tests.sh
```

9.2 License

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

9.3 Authors

Flask-Menu is developed for use in Invenio digital library software.

Contact us at info@inveniosoftware.org

- Krzysztof Lis <krzysztof.lis@cern.ch>
- Jiri Kuncar <jiri.kuncar@cern.ch>
- Tibor Simko <tibor.simko@cern.ch>
- Matthew Dillon <mrdillon@alaska.edu>
- Eirini Psallida <eirini.psallida@cern.ch>
- Florian Merges <fmerges@fstarter.org>
- Marco Neumann <marco@crepererum.net>
- Nick Whyte <nick@nickwhyte.com>
- Sami Hiltunen <sami.mikael.hiltunen@cern.ch>
- Sylvain Boily <sboily@proformatique.com>
- Marlin Forbes <marlinf@datashaman.com>

Python Module Index

f

Index