rutter Documentation

Release 0.3

Tres Seaver, Agendaless Consulting

Mar 13, 2022

Contents

`rutter` forks the `paste.urlmap` module in order to provide a Python3-compatible impolementation, as well as the improved test coverage needed to support using the module across all supported Python versions.

The primary export of `rutter` is the `rutter.urlmap.URLMap` class. `URLMap` instances are dictionary-like objects which dispatch to WSGI applications based on the URL.

The keys in a `URLMap` are URL patterns, which match as prefixes of the request URL (e.g., like `PATH_INFO.startswith(key)`). Its values are WSGI applications to which matching requests are dispatched. On finding a match, the `URLMap` adjusts the `SCRIPT_NAME` and `PATH_INFO` environmental variables to indicate the new context.

CHAPTER 1

URL Matching Rules

- URLs are matched most-specific-first, i.e., longest URL first.
- URL prefixes can also include domains, e.g. http://blah.com/foo. Domains can also be specified as tuples ('blah.com', '/foo').
- If a given pattern includes a domain, its path will only be tested if the `HTTP_HOST` environment variable matches.
- Patterns which do not have domains will be tested only if no domain-specifc pattern matches.

CHAPTER 2

Examples

2.1 Sample Applications

Assume we want to serve two WSGI applications provided by separate modules, `alpha`:

```
from pyramid.config import Configurator
from pyramid.view import view_config

@view_config(renderer='string')
def hello_alpha(request):
    return 'Hello, Alpha'

def main(global_config=None, **local_config):
    config = Configurator()
    config.scan()
    return config.make_wsgi_app()
```

and `bravo`.

```
from pyramid.config import Configurator
from pyramid.view import view_config

@view_config(renderer='string')
def hello_bravo(request):
    return 'Hello, Bravo'

def main(global_config=None, **local_config):
    config = Configurator()
    config.scan()
    return config.make_wsgi_app()
```

Note: Although these examples use `pyramid`; any WSGI-compliant application can be used as a dispatch target.

2.2 Imperative Configuration in Python

```
from wsgiref.simple_server import make_server

from rutter.urlmap import URLMap

from alpha import main as alpha_main
from bravo import main as bravo_main

def main():
    # Grab the config, add a view, and make a WSGI app
    urlmap = URLMap()
    urlmap['/alpha'] = alpha_main()
    urlmap['/bravo'] = bravo_main()
    return urlmap

if __name__ == '__main__':
    # When run from command line, launch a WSGI server and app
    app = main()
    server = make_server('0.0.0.0', 6543, app)
    server.serve_forever()
```

Assuming that `alpha` and `bravo` are importable, along with `rutter`, we can run this application:

```
$ /path/to/python example/imperative.py
```

and then visit the two applications at http://localhost:6543/alpha and http://localhost:6543/bravo.

2.3 Declarative Configuration using `paste.deploy` INI files

Assuming that we have a `paste.deploy`-compatible server starter (such as the **`pserve`** script installed by `pyramid`), we can configure the *:class:'~rutter.urlmap.URLMap* via an INI file:

```
[app:alpha]
use = egg:rutter_example#alpha

[app:bravo]
use = egg:rutter_example#bravo

[composite:main]
use = egg:rutter#urlmap
/bravo = bravo
/alpha = alpha

[server:main]
use = egg:pyramid#wsgiref
port = 6543
host = 127.0.0.1
```

And then run the composite application using the starter:

```
$ /path/to/pserve example/development.ini
```

The two applications are again available at http://localhost:6543/alpha and http://localhost:6543/bravo.

Index