warouter

Release 0.1.1

February 24, 2015

Contents

Warouter is a simple routing wrapper around webapp2.

It provides url inheritance for handlers and a convenient `uri_for()` function which can be called with a handler, instead of a string name.

A WSGIApplication mixin is provided which can handle a list of `url()` decorator handlers.

Example:

```
>>> import warouter
>>> import webapp2
>>>
>>> @warouter.url('/')
... class RootHandler(webapp2.RequestHandler):
...     def get(self):
...         self.response.write('root')
...
>>> @warouter.url('/child/<child_param:([a-z]+)>')
... class ChildHandler(RootHandler):
...     def get(self, child_param):
...         self.response.write(child_param)
...     def put(self, child_param):
...         pass
...
>>> @warouter.url('/grandchild/<grandchild_param:([a-z]+)>')
... class GrandChildHandler(ChildHandler):
...     def get(self, child_param, grandchild_param):
...         self.response.write('
'.join([child_param, grandchild_param]))
...     def post(self, child_param, grandchild_param):
...         self.response.write(warouter.uri_for(ChildHandler,
...                                              child_param=child_param))
...
>>> assert RootHandler.url == '/'
>>> assert ChildHandler.url == '/child/<child_param:([a-z]+)>'
>>> assert GrandChildHandler.url == (
...     '/child/<child_param:([a-z]+)>/grandchild/<grandchild_param:([a-z]+)>')
>>> assert GrandChildHandler.put is None
>>>
>>> app = warouter.WSGIApplication([
...     RootHandler,
...     ChildHandler,
...     GrandChildHandler
... ])
>>>
>>> if __name__ == '__main__':
...     from paste import httpserver
...     httpserver.serve(app, port='8080')
...
>>>
```

The above example requires Paste, which can be installed using:

```
$ pip install paste
```

class warouter.**WSGIApplicationMixin**(*mapping*, **args*, ***kwargs*)

This mixin allows a list of `url()` decorated request handlers to be passed to the WSGI application mapping instead of `webapp2.Route` objects or tuples.

warouter_logger = logging

warouter_logging_level = logging.NOTSET

warouter_logging_format = '{0} → {1.__module__}.{1.__name__}'

class warouter.**WSGIApplication**(*mapping*, **args*, ***kwargs*)
Bases: warouter.WSGIApplicationMixin, webapp2.WSGIApplication

Implements WSGIApplicationMixin and webapp2.WSGIApplication for convenience.

warouter.**uri_for**(*handler*, **args*, ***kwargs*)
Gets the uri for a handler based on its url.

Parameters

- **handler** (*webapp2.RequestHandler*) – The handler for which to find a url. If this is a string, it will be tried to import a handler from this string.
- **args** – passed to webapp2.uri_for().
- **kwargs** – passed to webapp2.uri_for().

Returns A string containing the url for the handler.

Return type basestring

warouter.**url**(*url*, ***param_mapping*)
Specifies a url for a webapp2.RequestHandler.

This decorator specifies a url for a handler by appending it to the url of its parent handlers.

A param mapping may be specified as keyword arguments. the url parameters will be matched against these values and be converted by the callable specified as the kwarg value.

Parameters

- **url** (*basestring*) – The url to append to the decorated handler.
- ****param_mapping** – A dict of key, value pairs where the key specifies the name of a url parameter to vonvert and the value a callable used to convert the url parameter.

W

U

W