Watson - Routing

Release 1.2.1

Mar 30, 2017

Contents

Process and route HTTP Request messages.

CHAPTER 1

Build Status

CHAPTER 2

Installation

```
pip install watson-routing
```

CHAPTER 3

Testing

Watson can be tested with py.test. Simply activate your virtualenv and run `python setup.py test`.

CHAPTER 4

Contributing

If you would like to contribute to Watson, please feel free to issue a pull request via Github with the associated tests for your code. Your name will be added to the AUTHORS file under contributors.

CHAPTER 5

Table of Contents

Usage

Tip: LiteralRoutes are faster than SegmentRoutes for standard path mapping.

Creating a route

Routes can be created multiple different ways, instantiating it as a class on its own, or adding definitions to a router.

Standalone

```
from watson.routing import routes

routes.LiteralRoute('home', path='/')
```

Tip: You must always specify at least name and a path for the route, unless you are creating a regex route.

From the router

```
from watson.routing import routers

routers.DictRouter({
    'home': {
        'path': '/'
    }
})
```

Types of routes

watson-routing currently provides two distinct types of routes, LiteralRoutes and SegmentRoutes. LiteralRoutes provide direct URL path to route mapping. SegmentRoutes allow you to map required or optional parameters in the URL path.

```
from watson.routing import LiteralRoute, SegmentRoute

LiteralRoute('home', path='/')  # a standard segment route
SegmentRoute('content', path='/:content')
```

SegmentRoutes also have the ability to have their paths matched via regex. This can be done by supplying either a regular expression, or a string that is to be converted into a regular expression into the constructor.

```
from watson.routing import SegmentRoute

SegmentRoute('about', regex='^/about')
```

Child routes

Defining the same top level routes over and over can get cumbersome quickly, so Watson provides a way to specify custom child routes in a route definition.

```
# a route definition

{
    'blog': {
        'path': '/blog',
        'children': {
            'categories': {
                'path': /categories
            }
        }
    }
}
```

When the above route definition is added to a router, two routes will be created, mapping to the following urls:

- /blog
- /blog/categories

Assembling Routes

Instead of manually trying to create links to urls within your application, you can easily use the assemble method. A shortcut to this is also available on the Router.

```
segment = SegmentRoute('blog', path='/blog[/:category[/:post]]')

segment.assemble(category='python', post='watson')

router = routes.DictRouter()
```

```
router.add_route(segment)
router.assemble('blog', category='python', post='watson')
```

Putting it all together

Using watson-router in a simple WSGI application is quite straightfoward.

```
from watson.http.messages import Request, Response
from watson.routing import routers

def application(environ, start_response):
    request = Request.from_environ(environ)
    router = routers.DictRoute({
        'home': {
            'path': '/'
        }
    })
    match = router.match(request)
    response = Response(body='Match found: {0}'.format(match))
    return response(start_response)
```

We do recommend however that you use it with watson-framework, where you only need to worry about defining your routes within a configuration file.

Reference Library

watson.routing.routers

watson.routing.routes