Respite Documentation

Release 1.4.0

Johannes Gorset

January 11, 2016

Contents

CHAPTER 1

About

Respite conforms Django to Representational State Transfer (and, incidentally, HTTP).

CHAPTER 2

Installation

Stable releases of Respite are distributed via the python package index. See the *installation* page for detailed instructions.

CHAPTER 3

Documentation

3.1 Overview

If you're new to Respite, please see the *overview-and-tutorial* for an introduction. The rest of the documentation will assume you're at least passingly familiar with the material contained within.

3.2 Usage documentation

Respite's usage documentation expands upon the concepts outlined in the *overview-and-tutorial*.

3.2.1 Decorators

`respite.decorators.`**`before`**(*method_name*)

Run the given method prior to the decorated view.

If you return anything besides `None` from the given method, its return values will replace the arguments of the decorated view.

If you return an instance of `HttpResponse` from the given method, Respite will return it immediately without delegating the request to the decorated view.

Example usage:

```
class ArticleViews(Views):

    @before('_load')
    def show(self, request, article):
        return self._render(
            request = request,
            template = 'show',
            context = {
                'article': article
            }
        )

    def _load(self, request, id):
        try:
            return request, Article.objects.get(id=id)
        except Article.DoesNotExist:
            return self._error(request, 404, message='The article could not be found.')
```

Parameters **method** – A string describing a class method.

respite.decorators.**override_supported_formats**(*formats*)
Override the views class' supported formats for the decorated function.

Arguments: formats – A list of strings describing formats, e.g. `['html', 'json']`.

3.2.2 Routing

Respite connects views to URLs through `resource` declarations, each of which define routes for a particular collection of views.

respite.urls.**resource**(*views*, *routes*, *prefix=''*)
Route a collection of views.

Parameters

- **views** – A reference to the class that defines the views.
- **routes** – A list of routes.
- **prefix** – A string to prefix the routes by, or `''` by default.

```
# urls.py

urlpatterns = resource(
    prefix = 'posts/',
    views = PostViews,
    routes = [
        ...
    ]
)
```

Routes

There are two ways in which you might populate the resource's routes: you can declare them inline using the `route` function, or reference views that have been decorated with the `route` decorator.

Inline routes

respite.urls.routes.**route**(*regex*, *view*, *method*, *name*)
Route the given view.

Parameters

- **regex** – A string describing a regular expression to which the request path will be matched.
- **view** – A string describing the name of the view to delegate the request to.
- **method** – A string describing the HTTP method that this view accepts.
- **name** – A string describing the name of the URL pattern.

`regex` may also be a lambda that accepts the parent resource's `prefix` argument and returns a string describing a regular expression to which the request path will be matched.

`name` may also be a lambda that accepts the parent resource's `views` argument and returns a string describing the name of the URL pattern.

```
# urls.py

urlpatterns = resource(
    prefix = 'posts/',
    views = PostViews,
    routes = [
        # Route GET requests to 'posts/' to the 'index' view.
        routes.route(
            regex = r'^(?:$|index(?:\.[a-zA-Z]+)?$)',
            view = 'index',
            method = 'GET',
            name = 'blog_posts'
        ),
        # Route GET requests 'posts/1' to the 'show' view.
        routes.route(
            regex = r'^(?P<id>[0-9]+)(?:\.[a-zA-Z]+)?$',
            view = 'show',
            method = 'GET',
            name = 'blog_post'
        )
    ]
)
```

Referenced routes

respite.decorators.**route**(*regex*, *method*, *name*)

Route the decorated view.

Parameters

- **regex** – A string describing a regular expression to which the request path will be matched.
- **method** – A string describing the HTTP method that this view accepts.
- **name** – A string describing the name of the URL pattern.

`regex` may also be a lambda that accepts the parent resource's `prefix` argument and returns a string describing a regular expression to which the request path will be matched.

`name` may also be a lambda that accepts the parent resource's `views` argument and returns a string describing the name of the URL pattern.

```
# views.py

class PostViews:

    @route(
        regex = r'^(?:$|index(?:\.[a-zA-Z]+)?$)',
        method = 'GET',
        name = 'blog_posts'
    )
    def index(request):
        ...

    @route(
        regex = r'^(?P<id>[0-9]+)(?:\.[a-zA-Z]+)?$',
        method = 'GET',
        name = 'blog_post'
    )
```

```
    def show(request, id):
        ...

# urls.py

urlpatterns = resource(
    prefix = 'posts/',
    views = PostViews,
    routes = [
        PostViews.index.route,
        PostViews.show.route
    ]
)
```

3.2.3 Views

In Respite, views are encapsulated in classes according to the model they supervise. You are not required to subclass Respite's `Views` class, but doing so will yield some things you might find useful:

class `respite.`**Views**
Base class for views.

Attribute template_path A string describing a path to prefix templates with, or `''` by default.

Attribute supported_formats A list of strings describing formats supported by these views, or `['html']` by default.

_render(*request*, *template=None*, *status=200*, *context={}*, *headers={}*, *prefix_template_path=True*)
Render a HTTP response.

Parameters

- **request** – A django.http.HttpRequest instance.
- **template** – A string describing the path to a template.
- **status** – An integer describing the HTTP status code to respond with.
- **context** – A dictionary describing variables to populate the template with.
- **headers** – A dictionary describing HTTP headers.
- **prefix_template_path** – A boolean describing whether to prefix the template with the view's template path.

Please note that `template` must not specify an extension, as one will be appended according to the request format. For example, a value of `blog/posts/index` would populate `blog/posts/index.html` for requests that query the resource's HTML representation.

If no template that matches the request format exists at the given location, or if `template` is `None`, Respite will attempt to serialize the template context automatically. You can change the way your models are serialized by defining `serialize` methods that return a dictionary:

```
class NuclearMissile(models.Model):
    serial_number = models.IntegerField()
    is_armed = models.BooleanField()
    launch_code = models.IntegerField()

    def serialize(self):
        return {
            'serial_number': self.serial_number,
```

```
        'is_armed': self.is_armed
    }
```

If the request format is not supported by the view (as determined by the `supported_formats` property or a specific view's `override_supported_formats` decorator), this function will yield HTTP 406 Not Acceptable.

Default views

Respite defines a collection of views that facilitate for common features like viewing a list of items, viewing a specific item by its id, rendering a form to create a new item and so on. You can leverage these views by adding Respite's `Resource` class to the base classes of your views class:

```
class PostViews(Views, Resource):
  model = Post
  template_path = 'posts/'
  supported_formats = ['html', 'json']
```

class `respite.`**`Resource`**

A collection of views that facilitate for common features.

Attribute model A reference to a model.

Attribute form A reference to a form, or `None` to generate one automatically.

`index`(*request*)
Render a list of objects.

`show`(*request*, *id*)
Render a single object.

`new`(*request*)
Render a form to create a new object.

`create`(*request*)
Create a new object.

`update`(*request*, *id*)
Update an object.

`replace`(*request*, *id*)
Replace an object.

`destroy`(*request*, *id*)
Delete an object.

`Resource` automatically generates routes for each of its views and names them appriopriately. In our example, the following routes would be generated:

HTTP path	HTTP method	View	Name
posts/	GET	index	posts
posts/	POST	create	posts
posts/new	GET	new	new_post
posts/1	GET	show	post
posts/1/edit	GET	edit	edit_post
posts/1	PUT	replace	post
posts/1	PATCH	update	post
posts/1	DELETE	destroy	post

CHAPTER 4

Development

Please see the *development* page for comprehensive information on contributing to Respite.

r

Symbols

B

C

D

I

N

O

R

S

U

V