django-hilbert Documentation

Release 0.1.0

Mark Lavin

December 14, 2011

CONTENTS

This is one of many Django apps which is a loose collection of utility functions. It is a mixture of Python code and Javascript that I find myself writing over and over. Primarily it focuses around utilities for AJAX and testing.

CHAPTER

ONE

INSTALLATION REQUIREMENTS

- Django >= 1.2
- jQuery >= 1.4.2

Optional (but recommended)

- django-staticfiles

The jQuery library is not included in the distribution but should be included in your templates.

To use the *CoverageRunner* you need to install Ned Batchelder's coverage.py.

Contents:

1.1 Motivation

Why not just put these on djangosnippets? Well some of these can be found there or inspired work in this project and I've tried to note those cases. My primary problem with djangosnippets is:

1. Lack of tests
2. Lack of portablity
3. Lack of maintenance

Some might feel that the snippets are small enough that they don't need tests. Those people are wrong. Am I really supposed to stick code in my project that someone else wrote and isn't tested?

None of the snippets are pip installable. That is not the purpose of the site. However, that means the most useful snippets are repeated in a number projects and there is no way to push improvements upstream. Combined with the lack of tests this can make for a maintenance nightmare.

Snippets can indicate the Django version they were written against but they typically aren't maintained as Django deprecates functions, improves common idioms, and even elimates the need for the original snippet.

While any one of these problems could be ignored, together they have caused many to create similar snippet collections to alleviate some of these problems. I think that djangosnippets has some great content but that doesn't mean that it stops people from writing it over and over again. This project is not meant to be a replacement for djangosnippets but more a supplement to maintain my personal sanity.

1.2 Decorators

Below is the list of decorators available in this project.

1.2.1 ajax_login_required

This decorator works like the built-in Django *login_required* but it handles AJAX requests differently. For AJAX requests, this decorator passes back custom headers to tell the client to redirect to the login page. These headers are caught by a *ajaxComplete* listener contained within *jquery.dj.hilbert.js*.

This is based on answers from the Stackoverflow question "How to manage a redirect request after a JQuery Ajax call?".

1.2.2 ajax_only

The *ajax_only* decorator ensures that all requests made to a particular view are made as AJAX requests. Non-AJAX requests will recieve a 400 (Bad Request) response. This is based on snippet 771.

1.3 Http

This module defines additional HttpResponse types.

1.3.1 JsonResponse

JsonReponse takes context data and passes it through *simplejson.dumps*. It also changes the mimetype to application/json. This does not automatically handle queryset serialization.

1.4 Test

The test module defines a new test client, base testcase, testing mixins, and an alternate test runner.

1.4.1 New Test Client

hilbert.test.Client is a simple extension of the Django test client which allows for an extra argument in *get* and *post* called *is_ajax*. This will default to *False* but when passed as *True* it will make the request as an AJAX request.

1.4.2 Base TestCase

hilbert.test.TestCase is an extension of the Django TestCase which uses the above test client and defines some helpful methods.

`TestCase.`**`get_random_string`**(*length=10*, *choices=string.ascii_letters*)
This method is used to generate random string data used in various tests.

Parameters

- **length** – The length of the string to return
- **choices** – The character set from which to draw the string characters

Returns A random string

`TestCase.`**`get_random_email`**(*domain=u'example.com'*)
This method is used to generate random email for the given domain.

Parameters **domain** – The domain name for the email address

Returns A random email address as a string

`TestCase.`**`create_user`**`(`*data=None*`)`

This generates a new *django.contrib.auth.User*. If no data is given then the user will be given a random username, email, and password.

Parameters **data** – A dictionary of data for the user. Allowed keys: username, password, email

Returns A newly created User model

1.4.3 CoverageRunner

The *CoverageRunner* is a new test runner based on snippets 705 and 2052. It uses Ned Batchelder's coverage.py to determine the percent of code executed by the test suite. It can be enabled by setting *TEST_RUNNER='hilbert.test.CoverageRunner'* in your Django settings file. You must also define a set of submodules to be included in the report

```
COVERAGE_MODULES = (
    'decorators',
    'http',
    'forms',
    'models',
    'views',
)
```

Using this setting the test runner will report the coverage of listed submodules of the tested apps (if they exist).

1.4.4 ViewTestMixin

This is a testing mixin to help writing tests for your Django views. It will automatically reverse the data returned by *get_urls()* and attach it to *self.url*. It also contains one test which does a GET request on the url.

1.4.5 AuthViewMixin

AuthViewMixin extends the *ViewTestMixin* for testing views which require authentication. It automatically creates a user and signs them in for any requests. It adds an additional test to ensure that authentication is required. This must be used in conjunction with *hilbert.test.TestCase*.

CHAPTER

TWO

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*