

---

# **django-hilbert Documentation**

***Release 0.4.1***

**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.



# 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`.





# INSTALLATION INSTRUCTIONS

It is easiest to install django-hilbert from [PyPi](#):

```
pip instal django-hilbert
```



# CONTENTS

## 3.1 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 portability
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 eliminates 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.

### 3.1.1 Related Projects

There are some similar projects. If interested you should check out:

1. [django-annoying](#)
2. [djblets](#)
3. [django-utils](#)

I appologize in advance if you felt I've left out a project.

## 3.2 Decorators

Below is the list of decorators available in this project.

### 3.2.1 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?”.

```
@ajax_login_required
def authenticated_view(request):
    return HttpResponse()
```

### 3.2.2 ajax\_only

The *ajax\_only* decorator ensures that all requests made to a particular view are made as AJAX requests. Non-AJAX requests will receive a 400 (Bad Request) response. This is based on [snippet 771](#).

```
@ajax_only
def ajax_view(request):
    return HttpResponse()
```

### 3.2.3 anonymous\_required

New in version 0.2. This decorator is the opposite of `login_required`. It ensures that users attempting to view this page are not authenticated. By default this will redirect authenticated users to the server root `/` or you can specify another url as either an absolute path or as a named url pattern.

```
@anonymous_required(url='/hilbert/test/simple/')
def anonymous_only_view(request):
    return HttpResponse()
```

### 3.2.4 secure

New in version 0.2. This decorator is similar to *SSLRedirectMiddleware* but works independently. Decorating a view with *secure* will force this view to be accessed only over SSL.

```
@secure
def secure_view(request):
    return HttpResponse()
```

## 3.3 Finders

The finders module defines a new staticfiles storage and finder.

### 3.3.1 AppMediaDirectoriesFinder

New in version 0.3. There are still a number of Django applications out in the wild which put their static files (js/css/images) in a media folder rather than a static folder. This finder extends the built-in *AppDirectoriesFinder* to gather files from the media directories.

To use simply add to your `STATICFILES_FINDERS` setting.

```
STATICFILES_FINDERS = (
    ...
    'hilbert.finders.AppMediaDirectoriesFinder',
    ...
)
```

## 3.4 Http

This module defines additional `HttpResponse` types.

### 3.4.1 JsonResponse

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

## 3.5 Middleware

Below is the list of middlewares available in this project.

### 3.5.1 SSLRedirectMiddleware

New in version 0.2. This middleware handles redirecting the user onto SSL in two different ways. One way was made popular by [snippet 85](#) is to pass an extra kwarg `SSL` in the url pattern for the view. This is handy when you are writing the views yourself and have control over the url patterns but is gets messy when you including third party urls. This also will not work for *contrib.flatpage* since they are not tied to a view.

The second method is taken from [snippet 880](#) which adds a setting `SSL_PATTERNS` which are used to match urls that should be forced onto SSL. You might want to force the admin to be used only on SSL such as

```
SSL_PATTERNS = (r'^/admin/', )
```

This is a much more convenient method for handling large groups of urls or third party application urls than the first. However, in some ways it feels like double work of defining the url regular expressions.

Together these middleware provides a good amount of flexibility in defining views/urls which require SSL. See also *SSLUserMiddleware*. New in version 0.4. You can optionally redirect requests off of SSL by enabling `SSL_WHITELIST`. When enabled any HTTPS request which does not pass the `SSL` kwarg, use the *secure* decorator or match one of `SSL_PATTERNS` will be redirected back to HTTP. Additional middleware can also mark the request with *keep\_secure* to keep the request from being redirected.

### 3.5.2 SSLUserMiddleware

New in version 0.2. This middleware is a complement to *SSLRedirectMiddleware*. If it is included it will force authenticated users to always use SSL.

To use this middleware you must be using `django.contrib.auth.middleware.AuthenticationMiddleware` and it must be included above *SSLUserMiddleware*. Note if you would like to use both *SSLRedirectMiddleware* and *SSLUserMiddleware* then *SSLUserMiddleware* should be included first.

```
MIDDLEWARE_CLASSES = (
    ...
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    ...
    'hilbert.middleware.SSLUserMiddleware',
    'hilbert.middleware.SSLRedirectMiddleware',
)
```

## 3.6 Test

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

### 3.6.1 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.

### 3.6.2 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

### 3.6.3 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 using the setting *COVERAGE\_MODULES*.

```
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). New in version 0.4. If you usually want to pass the same set of test labels when you run tests, you can set *DEFAULT\_TEST\_LABELS* in your settings.

```
DEFAULT_TEST_LABELS = ['app1', 'app2.TestClass', 'app3.TestClass.test_method']
```

Then *django-admin.py test* will act like

```
django-admin.py test app1 app2.TestClass app3.TestClass.test_method
```

If you've done that, you can still pass 'all' on the command line to run tests as if you had not passed any test labels, e.g. *run test all*.

### 3.6.4 ViewTestMixin

This is a testing mixin to help writing tests for your Django views. It will automatically reverse the data returned by *get\_url()* and attach it to *self.url*. It also contains one test which does a GET request on the url. New in version 0.3. The *ViewTestMixin* changed in version 0.3 to expect a class attribute *url\_name*.

```
class DashboardTestCase(TestCase, ViewTestMixin):  
    url_name = 'dashboard'
```

If your url needs either args or kwargs you can override *get\_url\_args* or *get\_url\_kwargs*.

### 3.6.5 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*.

## 3.7 Available Settings

### 3.7.1 SSL\_ENABLED

New in version 0.2. *SSL\_ENABLED* is used to configure the *SSLRedirectMiddleware*, *SSLUserMiddleware* and *secure* decorator. This allows you to disable the SSL redirects in your development environment or while testing. *SSL\_ENABLED* defaults to *False*.

### 3.7.2 SSL\_PATTERNS

New in version 0.2. *SSL\_PATTERNS* is by *SSLRedirectMiddleware*. It defines a set of regular expressions for urls which should be accessed only over SSL. Use can still use *SSLRedirectMiddleware* without this setting. In that case you would need to use the *SSL* keyword argument in your url definitions. See *SSLRedirectMiddleware* for more detail.

### 3.7.3 SSL\_WHITELIST

New in version 0.4. *SSL\_WHITELIST* is used to configure the *SSLRedirectMiddleware*. When enabled any secure request that which is not marked as secure by either the keyword argument, *secure* decorator or matches one of *SSL\_PATTERNS* will be redirected off SSL. *SSL\_WHITELIST* defaults to *False*.

### 3.7.4 COVERAGE\_MODULES

*COVERAGE\_MODULES* is used by the *CoverageRunner* and defines a list of submodules which should be included the coverage report. If the submodule does not exist for a given app it will be skipped.

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

If you are not using *CoverageRunner* then you do not need to define this setting.

### 3.7.5 DEFAULT\_TEST\_LABELS

New in version 0.4. *DEFAULT\_TEST\_LABELS* is used by the *CoverageRunner*. It defines the default set of test labels when none are passed in invoking the test runner. This allows running tests on the same set of apps, test classes, and test methods each time.

```
DEFAULT_TEST_LABELS = ['app1', 'app2.TestClass', 'app3.TestClass.test_method']
```

If this is set, passing 'all' as the only test label on the command line will run all the tests.

## 3.8 Change Log

### 3.8.1 v0.4.1

- Fix issue with *SSLRedirectMiddleware* when using *SSL\_WHITELIST* and *SSL* keyword.

### 3.8.2 v0.4

- Added *DEFAULT\_TEST\_LABELS* setting for *CoverageRunner* (Thanks poirier)
- Added *SSL\_WHITELIST* setting for *SSLRedirectMiddleware*
- Test suite cleanup

### 3.8.3 v0.3

- Added *AppMediaDirectoriesFinder*
- **BACKWARDS INCOMPATIBLE** change to *ViewTestMixin* url -> url\_name



### 3.8.4 v0.2

- Added *SSLRedirectMiddleware*
- Added *SSLUserMiddleware*
- Added *anonymous\_required* decorator
- Added *secure* decorator
- Documentation updates

### 3.8.5 v0.1

- Initial public release



# INDICES AND TABLES

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