Contents

[How do I unit test django urls? 2](#_Toc29145153)

[Django Testing: URL mapping to the Class Based View 3](#_Toc29145154)

[Django Testing: URL mapping to the Function Based View 3](#_Toc29145155)

[Example: 4](#_Toc29145156)

[Testing if view uses proper template? 5](#_Toc29145157)

[Testing a model’s field 5](#_Toc29145158)

# [How do I unit test django urls?](https://stackoverflow.com/questions/18987051/how-do-i-unit-test-django-urls)

One way would be to [reverse](https://docs.djangoproject.com/en/1.11/ref/urlresolvers/#reverse) URL names and validate

Example:

urlpatterns = [

url(r'^archive/(\d{4})/$', archive, name="archive"),

url(r'^archive-summary/(\d{4})/$', archive, name="archive-summary"),

]

Now, in the test:

from django.urls import reverse

url = reverse('archive', args=[1988])

assertEqual(url, '/archive/1988/')

url = reverse('archive-summary', args=[1988])

assertEqual(url, '/archive-summary/1988/')

You are probably testing the views anyways.

Now, to test that the URL connect to the right view, you could use [resolve](https://docs.djangoproject.com/en/1.11/ref/urlresolvers/#resolve)

from django.urls import resolve

resolver = resolve('/summary/')

assertEqual(resolver.view\_name, 'summary')

Now in the variable resolver ([ResolverMatch](https://docs.djangoproject.com/en/1.11/ref/urlresolvers/#django.urls.ResolverMatch) class instance), you have the following options:

'app\_name',

'app\_names',

'args',

'func',

'kwargs',

'namespace',

'namespaces',

'url\_name',

'view\_name'

# [Django Testing: URL mapping to the Class Based View](https://stackoverflow.com/questions/51414993/django-testing-url-mapping-to-the-class-based-view)

class SimpleTestCase(TestCase):

def test\_home\_resolves\_HomePage\_func(self):

        resolver = resolve('/')

        print(resolver)

   # ResolverMatch(func=pages.views.HomePage, args=(), kwargs={}, url\_name=home,  
 # app\_names=['pages'], namespaces=['pages'], route=)

        self.assertEqual(resolver.func.\_\_name\_\_, HomePage.as\_view().\_\_name\_\_)

# Django Testing: URL mapping to the Function Based View

class HomeTests(TestCase):

    ...

    def test\_home\_url\_resolves\_home\_view(self):

        view = resolve('/')

        self.assertEquals(view.func, home)

## Example:

**urls.py**

urlpatterns = [

    path('', HomePage.as\_view(), name='home'),

    path('about/', AboutPage.as\_view(), name='about'),

]

**tests.py**

from django.test import TestCase

from django.urls import reverse, resolve

from pages.views import HomePage, AboutPage

class SimpleTestCase(TestCase):

    def test\_home\_page(self):

        url = reverse('pages:home')

        print(url)  # ./

        response = self.client.get(url)

        self.assertEqual(response.status\_code, 200)

    def test\_home\_resolves\_HomePage\_func(self):

        resolver = resolve('/')

        print(resolver)

   # ResolverMatch(func=pages.views.HomePage, args=(), kwargs={}, url\_name=home,  
 # app\_names=['pages'], namespaces=['pages'], route=)

        self.assertEqual(resolver.func.\_\_name\_\_, HomePage.as\_view().\_\_name\_\_)

    def test\_about\_page(self):

        url = reverse('pages:about')

        print(url)  # /about/

        response = self.client.get(url)

        self.assertEqual(response.status\_code, 200)

    def test\_about\_resolves\_AboutPage\_function(self):

        resolver = resolve('/about/')

        print(resolver)

         # ResolverMatch(func=pages.views.AboutPage, args=(), kwargs={}, url\_name=about,

         # app\_names=['pages'], namespaces=['pages'], route=about/)

        self.assertEqual(resolver.func.\_\_name\_\_, AboutPage.as\_view().\_\_name\_\_)

# Testing if view uses proper template?

class SimpleTestCase(TestCase):

    ...

    def test\_home\_uses\_correct\_template(self):

        homepage = self.client.get(reverse('pages:home'))

        # <TemplateResponse status\_code=200, "text/html; charset=utf-8">

        self.assertTemplateUsed(homepage, 'home.html')

**ValueError:** assertTemplateUsed() and assertTemplateNotUsed() are only usable on responses fetched using the Django test Client. Can’t use it as below:

class SimpleTestCase(TestCase):

    ...

    def test\_home\_uses\_correct\_template(self):

        homepage = resolve('/')

        self.assertTemplateUsed(homepage, 'home.html')

# Testing a model’s field

from .models import Post

class TestPost(TestCase):

    def setUp(self):

        Post.objects.create(text='Lorem ipsum dolor set ...')

    def test\_post\_content(self):

        post\_obj = Post.objects.get(id=1)

        self.assertEqual(f'{post\_obj.text}', 'Lorem ipsum dolor set ...')

# Testing a blog model (title, author, body)

from django.urls import reverse

from django.test import TestCase

from django.contrib.auth import get\_user\_model

from .models import Blog

class BlogTest(TestCase):

    def setUp(self):

        self.user = get\_user\_model().objects.create\_user(

             username='John Doe', email='johndoe@mail.com', password='testpassword')

        self.blog = Blog.objects.create(

            title='Title of Blog', author=self.user,

body='Content here, lorem ipsum dolor ...')

    def blog\_string\_representation(self):

        post = Blog(title='John Doe')

        self.assertEqual(str(post), post.title)

    def test\_home(self):

        homepage = self.client.get(reverse('blog:home'))

        self.assertEqual(homepage.status\_code, 200)

        self.assertTemplateUsed(homepage, 'home.html')

        self.assertContains(homepage, 'lorem')

    def test\_blog\_detail(self):

        aboutpage\_404 = self.client.get('/blog/10000/')

        self.assertEqual(aboutpage\_404.status\_code, 404)

        aboutpage = self.client.get('/blog/1/')

        self.assertEqual(aboutpage.status\_code, 200)

        self.assertTemplateUsed(aboutpage, 'blog\_detail.html')

        self.assertContains(aboutpage, 'ipsum dolor')