Tio Fani

17104016

**Unit Testing Web Page Pribadi Berbasis Python**

1. **Django**

Django adalah sebuah *framework* full-stack untuk membuat aplikasi web dengan bahasa pemrograman *Python*. *Framework* akan membantu kita membuat web lebih cepat, dibandingkan menulis kode dari nol.

*Full*-*stack* artinya, django meliputi bagian *front*-*end* dan juga *back*-*end*. *Front*-*end* adalah bagian yang akan dilihat dan berinteraksi langsung dengan pengguna, sedangkan *back*-*end* adalah bagian yang berhubungan dengan *database* dan logika pada aplikasi atau web.

1. **Unit Testing**

Unit testing merupakan bagian dari *software* *testing* dimana masing-masing unit atau komponen seperti prosedur, fungsi, kelas, dan lainnya dari aplikasi diuji. Tujuannya adalah untuk memvalidasi setiap unit dari aplikasi sesuai dengan yang dirancang. Unit testing biasanya dilakukan oleh *programmer*.

1. **Contoh Unit Testing web sederhana**
2. **Software yang digunakan**

Sebelum melakukan unit testing ada beberapa yang perlu diinstal terlebih dahulu :

1. Python

Python dapat diinstal dengan mudah. Langkah pertama yang harus dilakukan adalah mendownload python-3.5.0-amd64 atau versi lain yang sesuai dengan kebutuhan dari situsnya di python.org. Ukuran Python terbaru untuk semua versi sekitar 29 MB. Setelah download selesai, cukup double click untuk menjalankan program instalasinya. Saya menggunakan python versi 3.7.1 untuk melakukan unit testing.

1. Browser

Browser digunakan untuk menampilkan hasil dari kode yang telah dibuat. Terdapat berbagai macam browser,adapun browser yang saya gunakan yaitu Google Chrome versi 80.0.3987.149 (Official Build) (64-bit)

1. Text Editor

Text Editor merupakan sebuah aplikasi dimana pengguna dapat mengedit file berupa teks. Text Editor yang saya gunakan adalah Visual Studio Code dan Notepad++.

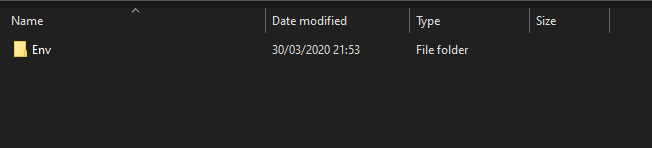
1. **Setup Windows**

Setelah semua terinstal selanjutnya mempersiapkan framework Django agar siap digunakan. Saya menggunakan sistem operasi windows jadi langkah-langkah berikut hanya berlaku untuk sistem operasi windows saja.

1. Buatlah folder baru dengan nama bebas apa saja dan lokasinya juga bebas dimana saja.
2. Selanjutnya buka command prompt lalu arahkan path ke lokasi folder yang sudah dibuat. Lalu ketikan

|  |
| --- |
| Python -m venv Env |

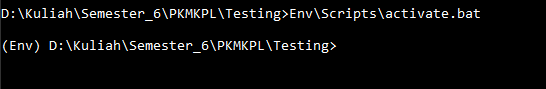
Maka otomatis terdapat folder baru bernama Env



Venv merupakan virtual environment dan Env merupakan nama foldernya. Virtual environment itu sendiri merupakan tools yang untuk membuat lingkungan python virtual yang terisolasi. Maksudnya program python yang berjalan pada virtual environment memiliki modulnya tersendiri dan program dari luar tidak bisa mengaksesnya.

1. Langkah selanjutnya yaitu mengaktifkan virtual environment dengan cara menjalankan activate.bat yang ada pada file Env tersebut

|  |
| --- |
| ..path\Scripts\activate.bat |



Virtual environment akan aktif jika terdapat (Env) pada command prompt.

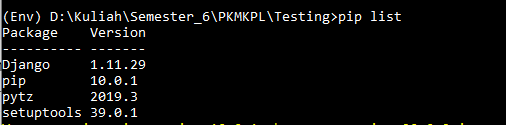
1. Selanjutnya yaitu menginstal Django dengan cara menggunakan perintah :

|  |
| --- |
| pip install Django==1.11.\* |

Untuk mengecek bisa menggunakan perintah

|  |
| --- |
| pip list |

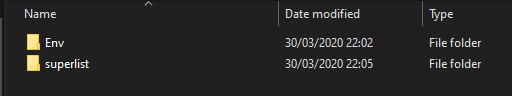
Jika sudah terinstall Django maka outputnya kurang lebih seperti berikut



1. Setelah sudah terinstal langkah selanjutnya membuat projek baru dengan menggunakan perintah

|  |
| --- |
| Django-admin startproject namaproject |

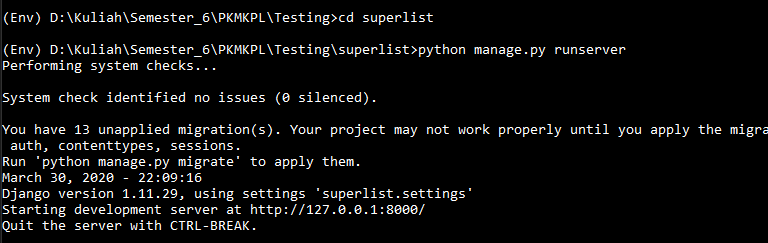
Jika berhasil maka perintah tersebut akan membuat folder baru sesuai dengan nama projeknya.



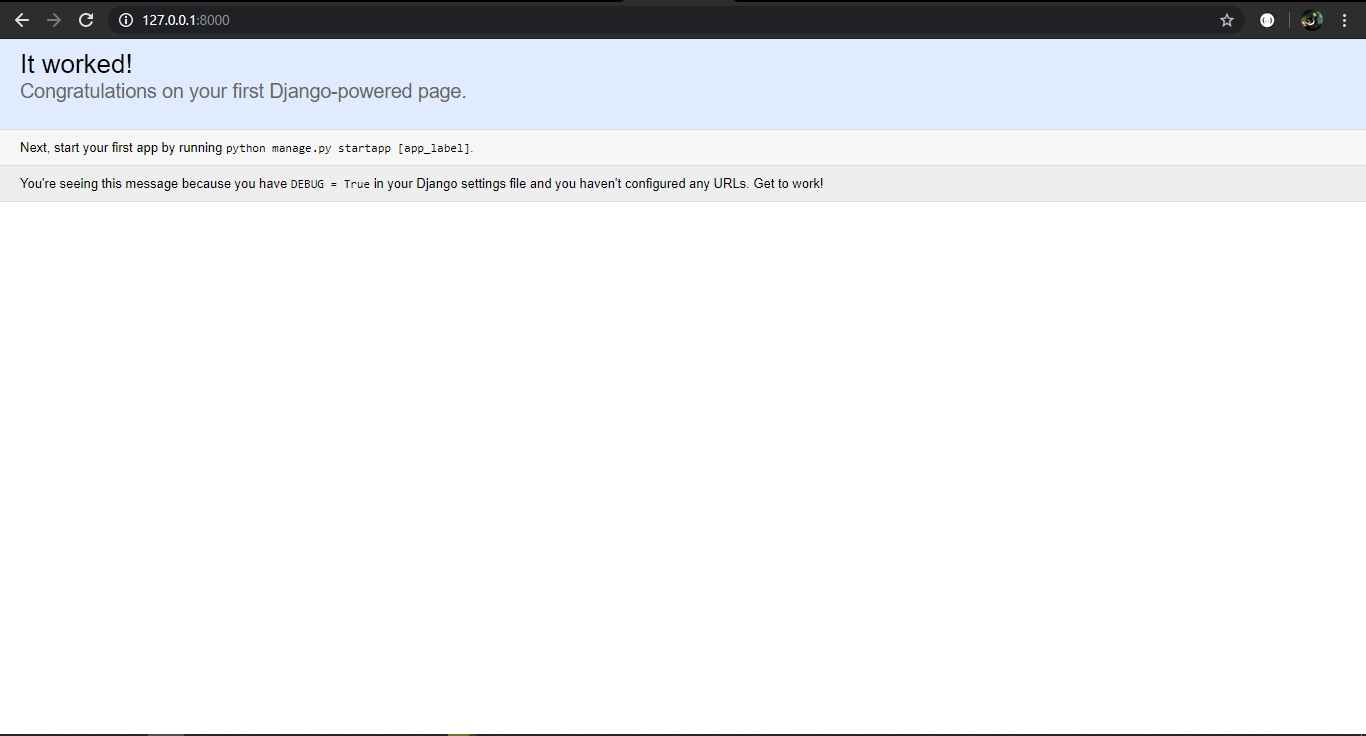
1. Persiapan sudah selesai, sekarang bisa dijalankan servernya dengan menggunakan perintah

|  |
| --- |
| Python manage.py runserver |

Maka tampilan pada command prompt kurang lebih seperti ini.



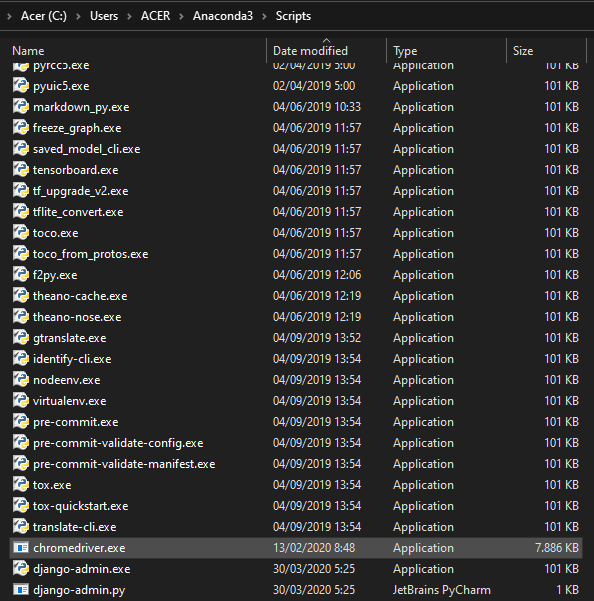
Selanjutnya silahkan buka browser dan copy IP Address lalu jalankan



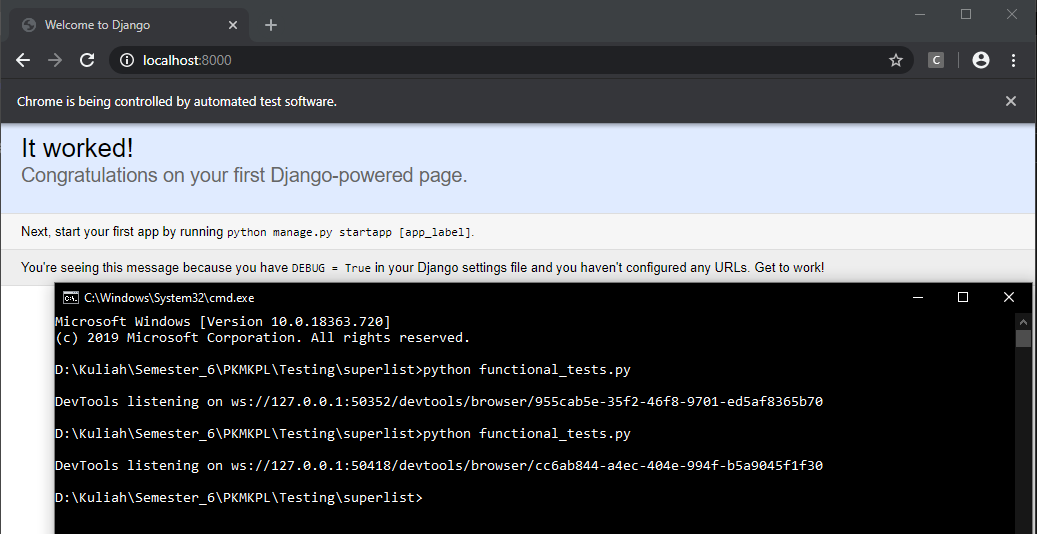
1. **Functional Test**
2. Buatlah file baru dengan nama functional\_tests.py lalu sesuaikan isinya seperti berikut

|  |
| --- |
| from selenium import webdriver  browser = webdriver.Chrome()  browser.get('http://localhost:8000')  assert 'Django' in browser.title |

Selanjutnya coba jalankan functional\_tests.py, jika terjadi error karena belum menginstal webdriver dari chrome, maka silahkan unduh pada tautan berikut <https://chromedriver.chromium.org/downloads>, lalu ekstrak dan pindahkan file webdriver tersebut ke path python Anda pada bagian Scripts



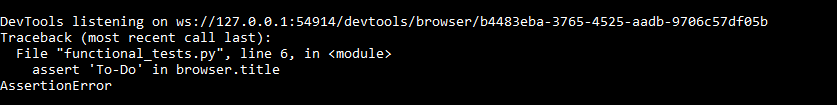
Coba jalankan file functional\_tests.py lagi maka kurang lebih akan seperti ini tampilannya



1. Ubah kode di functional\_tests.py menjadi seperti berikut

|  |
| --- |
| from selenium import webdriver  browser = webdriver.Chrome()  browser.get('http://localhost:8000')  assert 'To-Do' in browser.title  browser.quit() |

Lalu jalankan ulang program tersebut maka akan ada eror seperti berikut



Hal tersebut merupakan error yang wajar karena kita telah mengubah pada baris keenam yang maksudnya mencari kata ‘To-Do’ dari judul web dan kata tersebut memang tidak ada sehingga error. Adapun untuk mengetahui title dari web tersebut dapat menambahkan kode program berikut

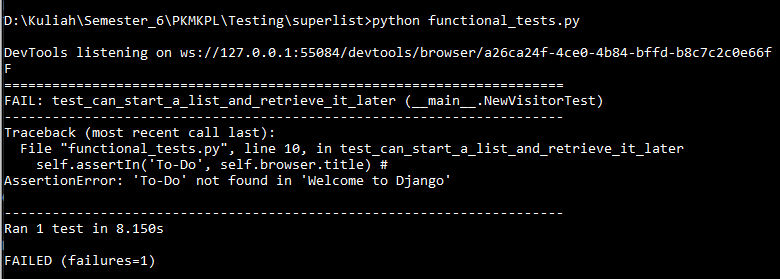
|  |
| --- |
| assert 'To-Do' in browser.title, "Browser title was " + browser.title |

Maka ketika program dijalankan akan menampilkan title dari webnya juga.

1. Sesuaikan kode di functional\_tests.py menjadi seperti berikut

|  |
| --- |
| from selenium import webdriver  import unittest  class NewVisitorTest(unittest.TestCase):  def setUp(self):  self.browser = webdriver.Chrome()  def tearDown(self):  self.browser.quit()  def test\_can\_start\_a\_list\_and\_retrieve\_it\_later(self):  self.browser.get('http://localhost:8000')  self.assertIn('To-Do', self.browser.title)  self.fail('Finish the test!')    if \_\_name\_\_ == '\_\_main\_\_':  unittest.main(warnings='ignore') |

Program di atas memiliki 3 method yaitu setUp dan terDown yang akan dijalankan sebelum dan sesudah melakukan testing. Method ketiga yaitu yang digunakan untuk melakukan testing-nya. Jika dijalankan maka program tersebut akan memiliki output seperti berikut



Hampir sama dengan output sebelumnya, tetapi kali ini output yang ditampilkan lebih jelas dan mudah dipahami. Kita juga dapat mengatur agar browser tertutup setelah 3 detik dengan menggunakan perintah

|  |
| --- |
| self.browser.implicitly\_wait(3) |

Letakan kode tersebut pada method setUp.

1. **Unit Testing Sederhana**
2. Buatlah folder baru dengan nama lists dengan menggunakan perintah berikut

|  |
| --- |
| python manage.py startapp lists |

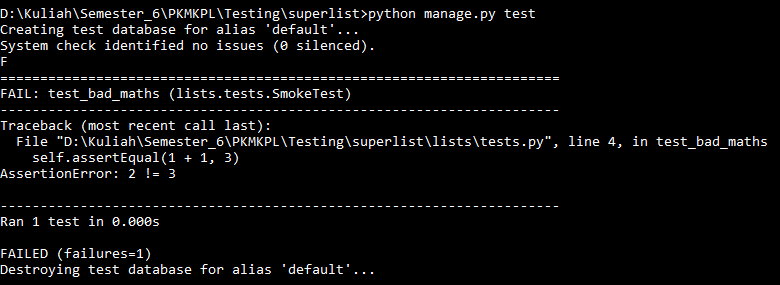
1. Selanjutnya buka file lists dan sesuaikan test.py sehingga seperti berikut

|  |
| --- |
| from django.test import TestCase  class SmokeTest(TestCase):  def test\_bad\_maths(self):  self.assertEqual(1 + 1, 3) |

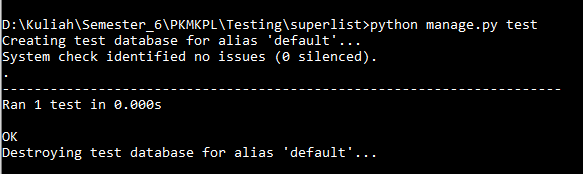
Kode di atas akan mengecek apakah 1+1 adalah 3. Untuk mengetesnya gunakan perintah

|  |
| --- |
| Python manage.py test |

Pasti akan muncul eror seperti gambar di bawah, eror dikarenakan 1+1 adalah 2 bukan 3.



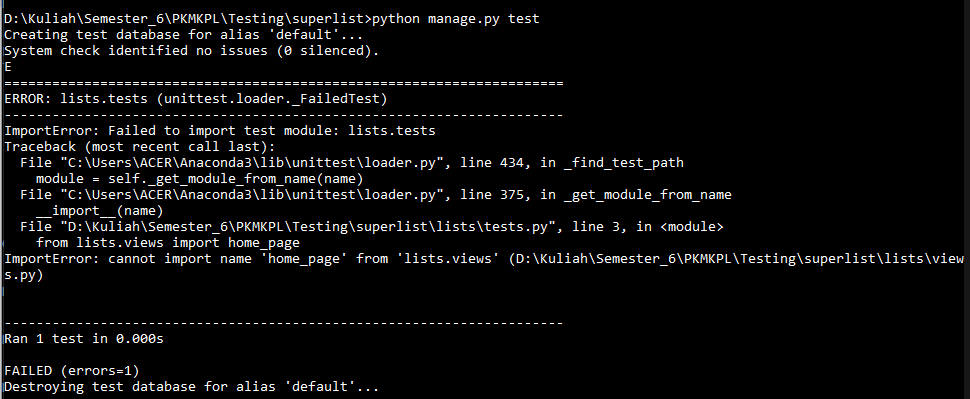
Jika kita mengganti angka tiga pada program, maka ketika dijalankan akan seperti ini outputnya, tidak ada eror yang muncul.



1. Selanjutnya sesuaikan tests.py seperti berikut

|  |
| --- |
| from django.core.urlresolvers import resolve  from django.test import TestCase  from lists.views import home\_page  class HomePageTest(TestCase):  def test\_root\_url\_resolves\_to\_home\_page\_view(self):  found = resolve('/')  self.assertEqual(found.func, home\_page) |

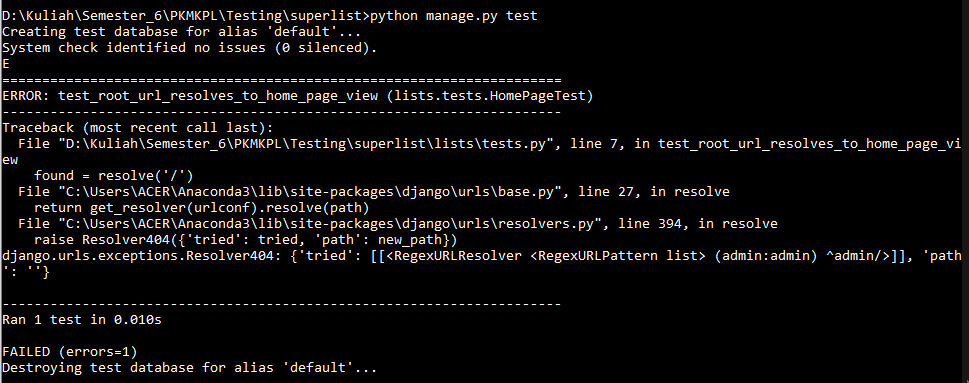
Lalu jalankan program seperti sebelumnya maka akan eror lagi, eror disini diakibatkan tidak bisa import home\_page dari kelas views karena memang belum ada home\_page dari kelas tersebut.



1. Tambahkan home\_page pada kelas views sehingga pada kelas views

|  |
| --- |
| home\_page = None |

Lalu jalankan kembali manage.py . Maka akan eror lagi namun kali ini erornya berbeda.



Error kali ini yaitu karena (‘/’), django tidak dapat menemukan pemetaan URL untuk ‘/’ .

1. Django menggunakan file urls.py untuk mendefiniskan URL map pada fungsi view.Buka urls.py dan sesuaikan kodenya menjadi seperti berikut

|  |
| --- |
| from django.conf.urls import url  from django.contrib import admin  urlpatterns = ['',  #url(r'^admin/', admin.site.urls),  #url(r'^$', 'superlists.views.home', name='home'),  #url(r'^$', 'lists.views.home\_page', name='home'),  ] |

Pada kali ini akan melakukan testing pada bagian url, silahkan hapus ‘#’ dan coba jalankan dengan menggunakan perintah

|  |
| --- |
| Pyton manage.py test |

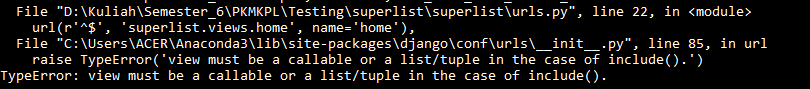
Terdapat berbagai macam eror, eror pertama yakni ketika tidak ada komen yang dihapus

|  |
| --- |
| Creating test database for alias 'default'...  SystemCheckError: System check identified some issues:  ERRORS:  ?: (urls.E004) Your URL pattern '' is invalid. Ensure that urlpatterns is a list of url() instances.  HINT: Try removing the string ''. The list of urlpatterns should not have a prefix string as the first element.  System check identified 1 issue (0 silenced). |

Selanjutnya coba hapus # pada urlpatterns baris kedua sehingga menjadi seperti berikut

|  |
| --- |
| urlpatterns = ['',  #url(r'^admin/', admin.site.urls),  url(r'^$', 'superlists.views.home', name='home'),  #url(r'^$', 'lists.views.home\_page', name='home'),  ] |

Maka ketika dijalankan akan mendapati eror seperti berikut



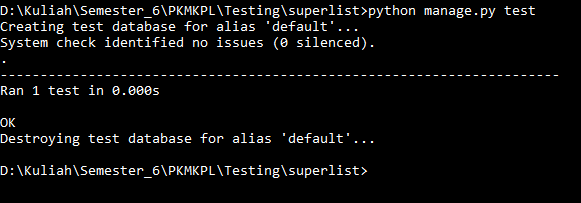
Untuk yang lainnya juga akan memiliki output yang hampir sama karena Django 1.10 tidak lagi mengizinkan menspesifikasikan views sebagai string. Maka rubah menjadi seperti berikut

|  |
| --- |
| from django.conf.urls import url  from django.contrib import admin  from lists import views  urlpatterns = [  # url(r'^admin/', admin.site.urls),  # url(r'^$', 'home', name='home'),  url(r'^$', views.home\_page, name='home'),  # url(r'^admin/', include(admin.site.urls)),  ] |

1. Selanjutnya buka views.py dan sesuaikan seperti berikut

|  |
| --- |
| from django.shortcuts import render  def home\_page():  pass |

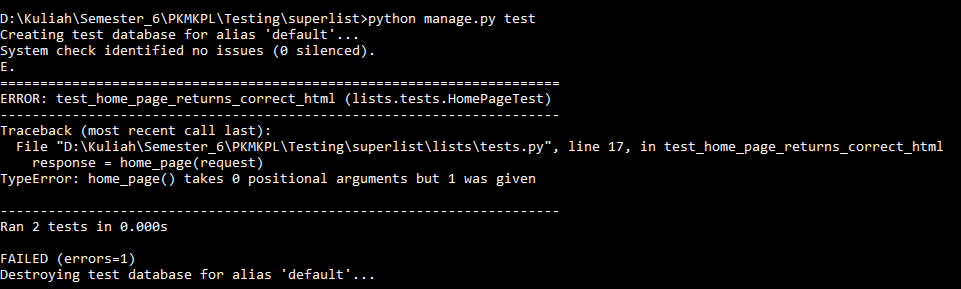
Maka output yang ditampilkan akan seperti berikut



1. Selanjutnya kita akan melakukan unit testing pada views, buka tests.py dan sesuaikan seperti berikut

|  |
| --- |
| from django.urls import resolve  from django.test import TestCase  from django.http import HttpRequest  from lists.views import home\_page  class HomePageTest(TestCase):  def test\_root\_url\_resolves\_to\_home\_page\_view(self):  found = resolve('/')  self.assertEqual(found.func, home\_page)  def test\_home\_page\_returns\_correct\_html(self):  request = HttpRequest()  response = home\_page(request)  html = response.content.decode('utf8')  self.assertTrue(html.startswith('<html>'))  self.assertIn('<title>To-Dolists</title>', html)  self.assertTrue(html.endswith('</html>')) |

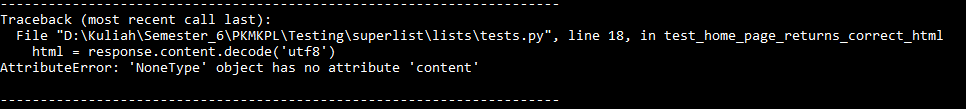
Kode di atas membuat objek HttpRequest yang akan dilihat ketika browser dibuka. Lalu akan melakukan cek pada title atau judul dari web tersebut. Ketika dijalankan maka output akan menampilkan eror seperti berikut



Sesuaikan views.py menjadi seperti berikut

|  |
| --- |
| def home\_page(request):  pass |

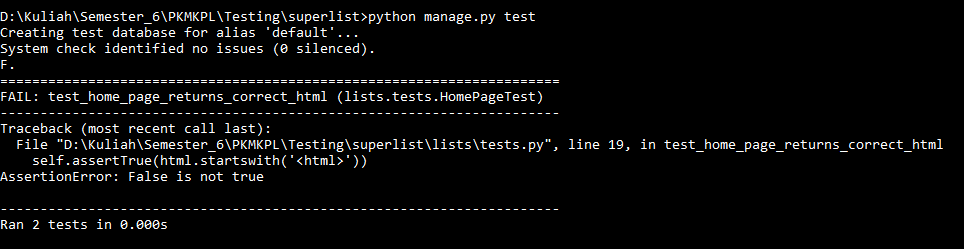
Jalankan lagi manage.py lagi maka output akan seperti berikut



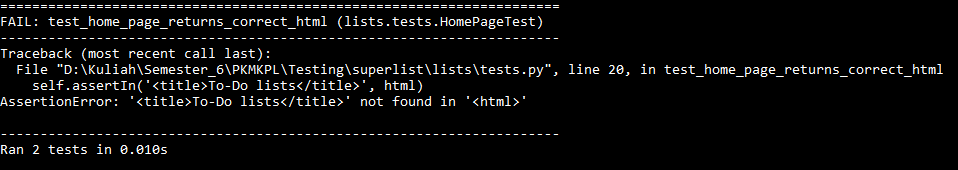
Sesuaikan lagi views.py menjadi seperti berikut

|  |
| --- |
| from django.http import HttpResponse  def home\_page(request):  return HttpResponse() |

Maka outputnya akan seperti berikut



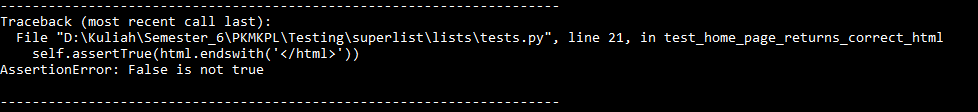
Akan muncul tidak ditemukannya ‘<html>’. Tambahkan '<html>' pada parameter HttpResponse, selanjutnya jalankan lagi



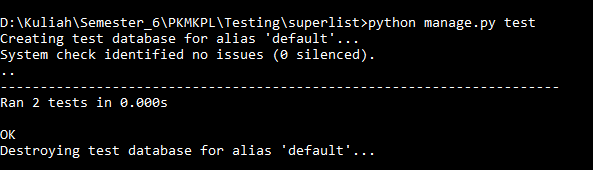
Kali ini tidak ditemukan ‘<title.....’> , jadi ubah parameter dari HttpResponse menjadi seperti berikut

|  |
| --- |
| return HttpResponse('<html><title>To-Do lists</title>') |

Jalankan lagi programmnya maka outputnya akan seperti berikut

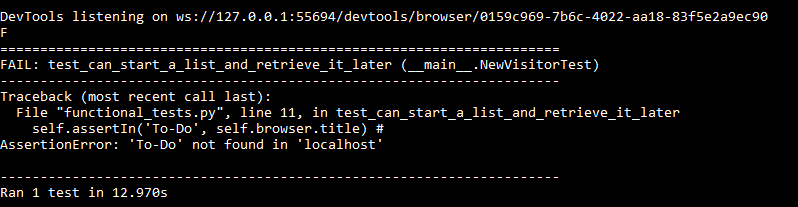


Masih eror, iya dikarenakan belum menutup tag html pada parameter HttpResponse. Tutup tag html dan coba lagi.



Akhirnya berhasil.

1. Selanjutnya jalankan functional\_tests.py , jangan lupa nyalakan servernya.



Akan eror karena browser belum diatur.

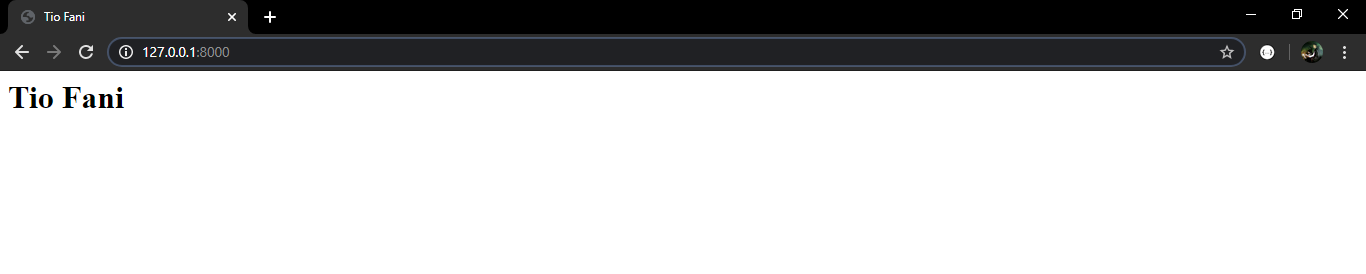
1. **Latihan Unit Testing**
2. Tambahkan kode berikut pada urls.py untuk mengganti judul web dan menambahkan nama

|  |
| --- |
| def index(request):  return HttpResponse("<title>Tio Fani</title><h1>Tio Fani</h1>") |

Jangan lupa untuk menghapus # pada bagian admin, sehingga kode dari urls.py akan seperti berikut

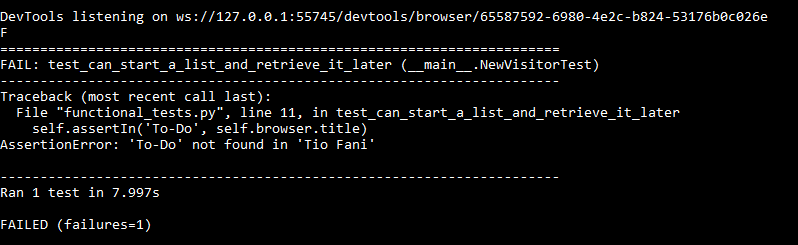
|  |
| --- |
| from django.conf.urls import url  from django.contrib import admin  from lists import views  from django.http import HttpResponse  def index(request):  return HttpResponse("<title>Tio Fani</title><h1>Tio Fani</h1>")  urlpatterns = [  #url(r'^$', 'home', name='home'),  #url(r'^$', views.home\_page, name='home'),  url(r'^admin/', admin.site.urls),  url(r'^$', index),  ] |

1. Matikan server terlebih dahulu lalu nyalakan kembali dan pastikan web sudah berubah tampilannya.



Tampilan web sudah berganti, sekarang kita akan melakukan testing untuk mengecek nama dari title/judul dari web tersebut.

1. Sebelumnya jalankan functional\_tests.py dulu untuk mengecek. Outputnya akan seperti berikut



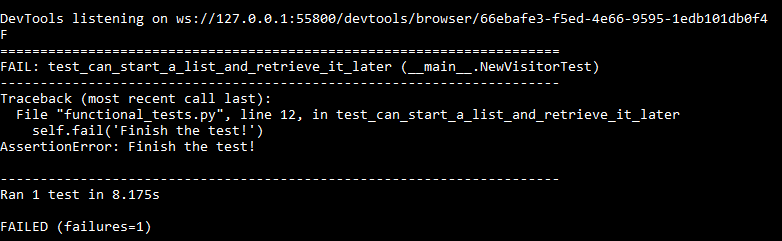
Eror di atas dikarenakan tidak ada title dengan nama ‘To-Do’ title yang tersedia adalah ‘Tio Fani’. Selanjutnya kita ubah pada bagian functional\_tests.py pada

|  |
| --- |
| self.assertIn('To-Do', self.browser.title) |

Menjadi seperti berikut

|  |
| --- |
| self.assertIn('Tio Fani, self.browser.title) |

Coba jalankan functional\_tests.py lagi, maka masih akan menampilkan eror seperti output berikut



Namun eror di atas bukanlah eror karena title yang tidak sama, melainkan karena self.fail yang merupakan turunan dari NewVisitorTest .

**Catatan :**

Saya mengirim projectnya saja dan tidak menyertakan file Env, jika mau mencoba tambahkan Env dulu dengan menggunakan perintah

|  |
| --- |
| Python -m venv Env |

**Referensi**

Buku 1, TDD, Harry J Percival, Bab 1-3