# Learning Django

## 

### 

django 1.8

* Basic Application
  + 2 Start Project
  + 3 First Migration
  + 4 Admin, SuperUser acticatation
  + 5 Apps
  + 6 First View & URL Routing
* Configuration
  + 7 Django Setting Overview
  + 8 Template Configuration
* 9 Model
* 10 Admin
* Form, Template (frontend) & View (backend)
  + 11 Model Form
  + 12 Form Validation
  + 13 View and Template Context
  + 14 Form in View
  + 15 Custom form in a view (NON ModelForm)
  + 21 Third Party Library Crispy Form
* 16 Setup email
* Static file
  + 17 static file in django1.8
  + 18 Serving static files
  + Bootstrap
    - 19 Add bootstrap to django
    - 22 Bootstrap Grid System
  + 23 Styling MVP Landing Part1
  + 26 Styling MVP Landing Part2
  + 31 Styling MVP Landing Part3
  + 32 Logo image in navbar
  + 33 Promo Video and Image
  + 34 Icon with Font Awesome
* Template
  + 20 Django template Include, Inheritance, Blocks
  + 24 CSS with blocks
  + 25 URL Names as Links
* Registration
  + 27 Django Registration Redux
  + 28 Update django login URL to Custom URL
  + 29 Authentication Links in Navbar
  + 30 Login Form in bootstrap Navbar
  + 35 Content with authenticated users
* 36 QuerySetBasics

# Start Project

## virtualenv

D:\>mkdir virtualdir

D:\>cd virtualdir

D:\>virtualenv trydjango18

New python executable in trydjango18\Scripts\python.exe

Installing setuptools, pip, wheel...done.

D:\virtualdir\trydjango18>ls

Include Lib Scripts

视频教程里面苹果系统生成的目录是 bin include lib

激活命令 source bin/activate

Windows激活

D:\virtualdir\trydjango18>Scripts\activate

(trydjango18) D:\virtualdir\trydjango18>

(trydjango18) D:\virtualdir\trydjango18>pip freeze

wheel==0.24.0

视频教程里面生成的目录是 wsgiref==0.1.2

## 安装Django

可以从django网站获取最新的django版本信息

<https://www.djangoproject.com/download/>

通过pip安装django

pip install django 会安装最新的版本

(trydjango18) D:\virtualdir\trydjango18**>**pip install django**==**1.8

Collecting django**==**1.8

D:\virtualdir\trydjango18\lib\site-packages\pip\\_vendor\requests\packages\urllib3\util\ssl\_.py:315: SNIMissingWarning: An HTTPS request has been made, but the SNI (Subject Name Indication) extension to TLS is **not** available on this platform. This may cause the server to present an incorrect TLS certificate, which can cause validation failures. **For** more information, see https://urllib3.readthedocs.org/en/latest/security.html#snimissingwarning.

SNIMissingWarning

D:\virtualdir\trydjango18\lib\site-packages\pip\\_vendor\requests\packages\urllib3\util\ssl\_.py:120: InsecurePlatformWarning: A true SSLContext object is **not** available. This prevents urllib3 from configuring SSL appropriately and may cause certain SSL connections to fail. **For** more information, see https://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarning.

InsecurePlatformWarning

Using cached Django-1.8-py2.py3-none-any.whl

Installing collected packages: django

Successfully installed django-1.8

安装完成之后可以通过pip freeze查看当前的版本信息

(trydjango18) D:\virtualdir\trydjango18**>**pip freeze

Django**==**1.8

wheel**==**0.24.0

pip freeze > requirements.txt将安装软件信息存储到requirements.txt

后面可以通过pip install –r requirements.txt直接安装全部软件

## Start Project

Django提供来django-admin.py简化project相关的工作。Unix下，它默认安装在/usr/bin，Win32下，它在python安装目录的Script目录下，例如c:\python27\Script。确保django-admin.py在你的系统环境Path里，这样你可以在命令行直接执行该命令。

(trydjango18) D:\virtualdir\trydjango18**>**django-admin.py startproject trydjango18

(trydjango18) D:\virtualdir\trydjango18**>**ls

Include Lib Scripts pip-selfcheck.json trydjango18

执行startproject命令后，会生成下列文件

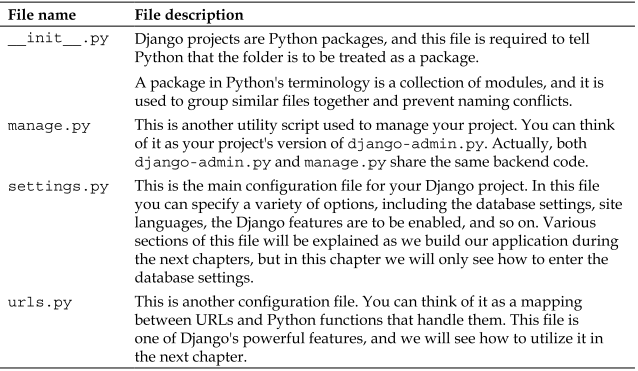
trydjango18/

\_\_init\_\_.py

manage.py

settings.py

urls.py



manage.py django项目的重要工具。

setting.py 配置文件，包括数据库信息，调试标志和其他重要的变量。这里定义的变量，多余所有安装的app都是可以访问的。

urls.py 配置文件，映射URL模式和app功能。

为避免与virtualenv名字冲突，把项目目录改成 “src”

(trydjango18) D:\virtualdir\trydjango18**>rename** trydjango18 src

(trydjango18) D:\virtualdir\trydjango18**>dir**

Volume **in** drive D is HP\_RECOVERY

Volume Serial Number is E237-2AC8

Directory of D:\virtualdir\trydjango18

2016/02/09 19:45 **<DIR>** .

2016/02/09 19:45 **<DIR>** ..

2015/12/11 23:42 **<DIR>** Include

2016/02/09 19:28 **<DIR>** Lib

2016/02/09 19:33 60 pip-selfcheck.json

2016/02/09 19:37 **<DIR>** Scripts

2016/02/09 19:42 **<DIR>** src

1 File(s) 60 bytes

6 Dir(s) 38,498,160,640 bytes free

(trydjango18) D:\virtualdir\trydjango18**>**ls

Include Lib Scripts pip-selfcheck.json src

## emulate a django server

django内置web server是最方便的内置功能之一。

优势包括：(1) 不需要安装Web Server (apache), (2) 自动检测和重新装载你的代码改动，相比重启Web Server，这个非常节省时间 (3) 它知道怎样找到和显示你的静态文件。

(trydjango18) D:\virtualdir\trydjango18\src**>**python manage.py runserver

Performing system checks...

System check identified no issues (0 silenced).

February 09, 2016 - 19:48:35

Django version 1.8, using settings 'trydjango18.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

这时，如果你打开调试窗口，会有下面打印

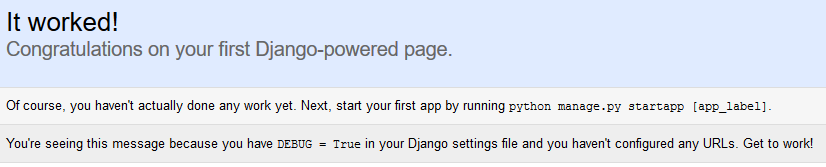
左右到右分别是：timestamp, request, HTTP response code, byte count.如果你访问一个不存在的网址或URL没配置好，返回码则为404.

[05/Mar/2017 16:38:23]"GET / HTTP/1.1" 200 12845

默认端口是8000, 也可以通过下列命令制定端口

python manage.py runserver 8080

python manage.py runserver 0.0.0.0:8000



## 将代码提交到github

先创建repository trydjango18course

echo "# trydjango18course" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin git@github.com:navicester/trydjango18course.git

git push -u origin master

# First Migration

**(**trydjango18**)** D**:**\virtualdir\trydjango18\src**>**python manage**.**py migrate

如果直接运行migrate命令，会有报错: settings**.**DATABASES **is** improperly configured,这是因为我们的database没有配置好

## 设置database

如果数据库还没有安装，推荐使用SQLite这个最快和简单的方式。SQLite非常快，广泛应用，存储为单个文件。

如果使用SQLite之外的数据库，如PostgreSQL，MySQL，Oracle，MSSQL，都需要在settings.py里面配置DATABASES。

添加 “ENGINE” 和 “NAME” 配置，例子中我们选用sqlite3，关于mysql的配置会有专门章节介绍

SQLite是最常用的选择，甚至如何没有大量同时写操作的情况下也可以部署。

DATABASES **=** **{**

'default'**:** **{**

'ENGINE'**:** 'django.db.backends.sqlite3'**,** # Add 'postgresql\_psycopg2', 'mysql', 'sqlite3' or 'oracle'.

'NAME'**:** r'db.sqlite3'**,** # Or path to database file if using sqlite3.

# The following settings are not used with sqlite3:

'USER'**:** ''**,**

'PASSWORD'**:** ''**,**

'HOST'**:** ''**,** # Empty for localhost through domain sockets or '127.0.0.1' for localhost through TCP.

'PORT'**:** ''**,** # Set to empty string for default.

**}**

**}**

在Python里，单引号和双引号在处理上是没有差别的，但必须保证开闭处符号相同。

在文件夹名字前，我加了一个r，表示’raw’，不要进行字符转义。例如：\n通常表示换行，但是在’raw’字符串里，它代表两个字符\和n。所以r仅用于dos文件系统，告诉python不要进行转义。

## 创建表

Database配置好了之后，再次运行migrate

老的django版本里，用syncdb命令。

(trydjango18) D:\virtualdir\trydjango18\src>python manage.py migrate

Operations to perform:

Synchronize unmigrated apps: staticfiles, messages

Apply all migrations: contenttypes, sites, auth, sessions

Synchronizing apps without migrations:

Creating tables...

Running deferred SQL...

Installing custom SQL...

**Running migrations**:

Rendering model states... DONE

Applying contenttypes.0001\_initial... OK

Applying contenttypes.0002\_remove\_content\_type\_name... OK

Applying auth.0001\_initial... OK

Applying auth.0002\_alter\_permission\_name\_max\_length... OK

Applying auth.0003\_alter\_user\_email\_max\_length... OK

Applying auth.0004\_alter\_user\_username\_opts... OK

Applying auth.0005\_alter\_user\_last\_login\_null... OK

Applying auth.0006\_require\_contenttypes\_0002... OK

Applying sessions.0001\_initial... OK

Applying sites.0001\_initial... OK

python manage.py syncdb从1.9开始会被删除掉

# Admin & Superuser

Createsuperuser命令可以用来创建超级用户，该命令任意时间都能执行

(trydjango18) D:\virtualdir\trydjango18\src>python manage.py createsuperuser

Username (leave blank to use 'alu'): alu

Email address: navicester@qq.com

Password:

Password (again):

Superuser created successfully.

(trydjango18) D:\virtualdir\trydjango18\src>python manage.py syncdb

D:\virtualdir\trydjango18\lib\site-packages\django\core\management\commands\syncdb.py:24: RemovedInDjango19Warning: The syncdb command will be removed in Django 1.9

warnings.warn("The syncdb command will be removed in Django 1.9", RemovedInDjango19Warning)

Operations to perform:

Synchronize unmigrated apps: staticfiles, admindocs, messages

Apply all migrations: admin, contenttypes, sites, auth, sessions

Synchronizing apps without migrations:

Creating tables...

Running deferred SQL...

Installing custom SQL...

Running migrations:

Rendering model states... DONE

Applying admin.0001\_initial... OK

Admin这个backend application号称django这个皇冠上的宝石。

对于django，admin这个是可选的，所以如果要使用这个功能，我们需要像其他app一样在setting文件里的INSTALLED\_APPS配置。

每加入一个app，都要执行syncdb/migrate以确保对应的table已经在database里创建。

## enable admin

去掉setting和url中的一些注释语句

setting.py

INSTALLED\_APPS **=** **(**

# Uncomment the next line to enable the admin:

'django.contrib.admin'**,**

# Uncomment the next line to enable admin documentation:

'django.contrib.admindocs'**,**

**)**

app配置好了之后，还需要再把它配置到URL，这样我们能够访问该app. app创建好了之后，系统默认加了这些url，但是注释掉了，把#去掉就可以了。

urls.py

# Uncomment the next two lines to enable the admin:

**from** django**.**contrib **import** admin

admin**.**autodiscover**()**

urlpatterns **=** patterns**(**''**,**

# Uncomment the admin/doc line below to enable admin documentation:

url(r'^admin/doc/', include('django.contrib.admindocs.urls')),

# Uncomment the next line to enable the admin:

url**(**r'^admin/'**,** include**(**admin**.**site**.**urls**)),**

**)**

# Apps

我们已经有了项目TryDjango18，我们可以在project下面创建application。

创建app我们会再次使用manage.py

(trydjango18) D:\virtualdir\trydjango18\src>python manage.py startapp newsletter

这样我们就在项目项目创建了app，它会创建下面文件

newsletter/

     \_\_init\_\_.py

     models.py

     tests.py

     views.py

和project一样，app也是package。

刚创建好时，models.py和views.py没有真正的代码，仅有placeholder。

要让project知道这个app，需要修改setting文件

在“INSTALLED\_APPS”下面添加“newsletter”

INSTALLED\_APPS **=** **(**

'django.contrib.admin'**,**

'django.contrib.auth'**,**

'django.contrib.contenttypes'**,**

'django.contrib.sessions'**,**

'django.contrib.messages'**,**

'django.contrib.staticfiles'**,**

'newsletter'

**)**

我们不能创建两个名字完全一样的application，但是可以创建于类似“admin” 这种build-in site-packages的application

# First View and URL Routing

添加最基本的view功能

* 首先添加url

trydjango18\urls.py

urlpatterns **=** patterns**(**''**,**

# Examples:

url**(**r'^$'**,** 'newsletter.views.home'**,** name**=**'home'**),**

**)**

|  |  |
| --- | --- |
| **Symbol / Expression** | **Matched String** |
| . (Dot) | Any character. |
| ^ (Caret) | Start of string. |
| $ | End of string. |
| \* | 0 or more repetitions. |
| + | 1 or more repetitions. |
| ? | 0 or 1 repetitions. |
| | | A | B means A or B. |
| [a-z] | Any lowercase character. |
| \w | Any alphanumeric character or \_. |
| \d | Any digit. |

这儿的name后面可以用在template里面的url引用，例如

templates\navbar.html

<li class=**"active"**><a href=**"{% url 'home' %}"**>**Home**</a></li>

* 添加view里面的实现

newsletter\view.py

**from** django**.**shortcuts **import** render

# Create your views here.

**def** home**(**request**):**

context **=** **{}**

**return** render**(**request**,** "home.html"**,** context**)**

# Django Setting Overview

(trydjango18) D:\virtualdir\trydjango18\src>tree /F

D:.

│ db.sqlite3

│ manage.py

│

├─newsletter

│ │ admin.py

│ │ models.py

│ │ tests.py

│ │ views.py

│ │ \_\_init\_\_.py

│ │

│ └─migrations

│ \_\_init\_\_.py

│

└─trydjango18

settings.py

settings.pyc

urls.py

urls.pyc

wsgi.py

wsgi.pyc

\_\_init\_\_.py

\_\_init\_\_.pyc

"""

Django settings for trydjango18 project.

Generated by 'django-admin startproject' using Django 1.8.

For more information on this file, see

https://docs.djangoproject.com/en/1.8/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/1.8/ref/settings/

"""

**BASE\_DIR**

import os.path

os.path.dirname(\_\_file\_\_)

返回当前路径

# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)

**import** os

BASE\_DIR **=** os**.**path**.**dirname**(**os**.**path**.**dirname**(**os**.**path**.**abspath**(**\_\_file\_\_**)))**

#root of project

ROOT\_URLCONF = 'trydjango18.urls'

MIDDLEWARE\_CLASSES

介于request和response之间

**TEMPLATES**

A list containing the settings for all template engines to be used with Django. Each item of the list is a dictionary containing the options for an individual engine.

Here’s a simple setup that tells the Django template engine to load templates from the templates subdirectory inside each installed application:

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

**TEMPLATE\_DIRS**

Default: () (Empty tuple)

Deprecated since version 1.8: Set the [DIRS](https://docs.djangoproject.com/en/1.8/ref/settings/#std:setting-TEMPLATES-DIRS) option of a DjangoTemplates backend instead.

List of locations of the template source files searched by [django.template.loaders.filesystem.Loader](https://docs.djangoproject.com/en/1.8/ref/templates/api/#django.template.loaders.filesystem.Loader), in search order.

Note that these paths should use Unix-style forward slashes, even on Windows.

**TEMPLATE\_CONTEXT\_PROCESSORS**

Default:

(

"django.contrib.auth.context\_processors.auth",

"django.template.context\_processors.debug",

"django.template.context\_processors.i18n",

"django.template.context\_processors.media",

"django.template.context\_processors.static",

"django.template.context\_processors.tz",

"django.contrib.messages.context\_processors.messages"

)

Deprecated since version 1.8: Set the 'context\_processors' option in the [OPTIONS](https://docs.djangoproject.com/en/1.8/ref/settings/#std:setting-TEMPLATES-OPTIONS) of a DjangoTemplates backend instead.

A tuple of callables that are used to populate the context in RequestContext. These callables take a request object as their argument and return a dictionary of items to be merged into the context.

Changed in Django 1.8:

Built-in template context processors were moved from django.core.context\_processors to django.template.context\_processors in Django 1.8.

**TEMPLATE\_LOADERS**

Default:

(

'django.template.loaders.filesystem.Loader',

'django.template.loaders.app\_directories.Loader'

)

Deprecated since version 1.8: Set the 'loaders' option in the [OPTIONS](https://docs.djangoproject.com/en/1.8/ref/settings/#std:setting-TEMPLATES-OPTIONS) of a DjangoTemplates backend instead.

A tuple of template loader classes, specified as strings. Each Loader class knows how to import templates from a particular source. Optionally, a tuple can be used instead of a string. The first item in the tuple should be the Loader’s module, subsequent items are passed to the Loader during initialization. See [The Django template language: for Python programmers](https://docs.djangoproject.com/en/1.8/ref/templates/api/).

# Template Configuration

# Create your views here.

**def** home**(**request**):**

context **=** **{}**

**return** render**(**request**,** "home.html"**,** context**)**

如果不创建template的话, <http://127.0.0.1:8000/> 将会报告下面的错误调试信息

TemplateDoesNotExist

Template-loader postmortem

Django tried loading these templates, in this order:

* Using loader django.template.loaders.filesystem.Loader:
* Using loader django.template.loaders.app\_directories.Loader:
  + D:\virtualdir\trydjango18\lib\site-packages\django\contrib\auth\templates\home.html (File does not exist)
  + D:\virtualdir\trydjango18\lib\site-packages\django\contrib\admin\templates\home.html (File does not exist)
  + D:\virtualdir\trydjango18\lib\site-packages\django\contrib\admindocs\templates\home.html (File does not exist)

在newsletter下面创建template目录，并且在“INSTALLED\_APPS”下面添加“newsletter”，django将会搜索该template目录

INSTALLED\_APPS **=** **(**

'django.contrib.admin'**,**

'django.contrib.auth'**,**

'django.contrib.contenttypes'**,**

'django.contrib.sessions'**,**

'django.contrib.messages'**,**

'django.contrib.staticfiles'**,**

'newsletter'

**)**

如果将newsletter注释掉，django会不找不到模板文件

Template-loader postmortem

Django tried loading these templates, in this order:

* Using loader django.template.loaders.filesystem.Loader:
* Using loader django.template.loaders.app\_directories.Loader:
  + D:\virtualdir\trydjango18\lib\site-packages\django\contrib\auth\templates\home.html (File does not exist)
  + D:\virtualdir\trydjango18\lib\site-packages\django\contrib\admin\templates\home.html (File does not exist)
  + D:\virtualdir\trydjango18\lib\site-packages\django\contrib\admindocs\templates\home.html (File does not exist)
  + D:\virtualdir\trydjango18\src\newsletter\templates\home.html (File does not exist)

在“newsletter”下面创建文件“home.html”, it works

本例子中，我们把templates从application目录中移到root文件夹，在src目录创建templates文件夹

修改settings.py

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [os.path.join(BASE\_DIR, "templates")],

'APP\_DIRS': True, # Whether the engine should look for template source files inside installed applications.

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

如果home.html没有创建，django会抛出下面异常，否则, it works.

Template-loader postmortem

Django tried loading these templates, in this order:

* Using loader django.template.loaders.filesystem.Loader:
  + D:\virtualdir\trydjango18\src\templates\home.html (File does not exist)
* Using loader django.template.loaders.app\_directories.Loader:
  + D:\virtualdir\trydjango18\lib\site-packages\django\contrib\auth\templates\home.html (File does not exist)
  + D:\virtualdir\trydjango18\lib\site-packages\django\contrib\admin\templates\home.html (File does not exist)
  + D:\virtualdir\trydjango18\lib\site-packages\django\contrib\admindocs\templates\home.html (File does not exist)

# Models

<https://docs.djangoproject.com/en/1.8/ref/models/>

<https://docs.djangoproject.com/en/1.8/ref/models/fields/>

## 定义Model

Django的数据库模型层用了很多ORM (Object-Relational Mapper)

#### Rich Field Types

|  |  |
| --- | --- |
| CharField和TextField | 用的最多的field，它们有很多的相似处，都用于储存文本。  CharField有限定的长度， TextField的长度本质上是不受限制的，依赖于你的需求和数据库的效率 |
| EmailField, URLField, 和IPAddressField | 本质上是CharField，增加了额外的验证。 |
| BooleanField 和 NullBooleanField | BooleanField用的比较多，用于存储True和False，但有的时候你可能在存储时还不有知道它的值，这时你可能会考虑用Empty或者Null来代替，NullBooleanField正是用于这种需求。 |
| FileField | FileField是最复杂的Field之一，因为它的大部分工作都根本不在数据库。  跟FilePathField类似，FileField只存储文件路径，但是它还有上传文件和存储到服务器上某个位置的功能。 |

**from** django**.**db **import** models

# Create your models here.

**class** **SignUp(**models**.**Model**):**

email **=** models**.**EmailField**()**

full\_name **=** models**.**CharField**(**max\_length**=**120**,** blank**=True,** null**=True)**

timestamp **=** models**.**DateTimeField**(**auto\_now\_add**=True,** auto\_now**=False)**

updated **=** models**.**DateTimeField**(**auto\_now\_add**=False,** auto\_now**=True)**

**def** \_\_unicode\_\_**(**self**):** #Python 3.3 is \_\_str\_\_

**return** self**.**email

SignUp从django.db.models.Model继承。这类是Model的标准基类，是django强大的ORM系统的核心。

## 创建表 - migrate database

makemigrations : 初始化migrations

migrates : 实际运行migrations并且存储到database

(trydjango18) D:\virtualdir\trydjango18\src>python manage.py makemigrations

Migrations for 'newsletter':

0001\_initial.py:

- Create model SignUp

调用该命令后，django会轮询INSTALLED\_APPS里面的app的models.py文件，对于它找到每一个model都会创建一个数据库表（有一些例外：There are

exceptions to this later when we get into fancy stuff such as many-to-many relations, but it’s true for this example. If you are using SQLite, you also notice the django.db database file is created exactly where you specified.）

(trydjango18) D:\virtualdir\trydjango18\src>python manage.py migrate

Operations to perform:

Synchronize unmigrated apps: staticfiles, admindocs, messages

Apply all migrations: sessions, admin, sites, auth, contenttypes, newsletter

Synchronizing apps without migrations:

Creating tables...

Running deferred SQL...

Installing custom SQL...

Running migrations:

Rendering model states... DONE

Applying newsletter.0001\_initial... OK

如果不做修改重新执行该命令，将显示No migrations to apply

(trydjango18) D:\virtualdir\trydjango18\src>python manage.py migrate

Operations to perform:

Synchronize unmigrated apps: staticfiles, admindocs, messages

Apply all migrations: sessions, admin, sites, auth, contenttypes, newsletter

Synchronizing apps without migrations:

Creating tables...

Running deferred SQL...

Installing custom SQL...

Running migrations:

No migrations to apply.

# Admin

将自己的模块加入管理工具中，这样我们就能够通过这个漂亮的界面添加、修改和删除数据库中的对象了。

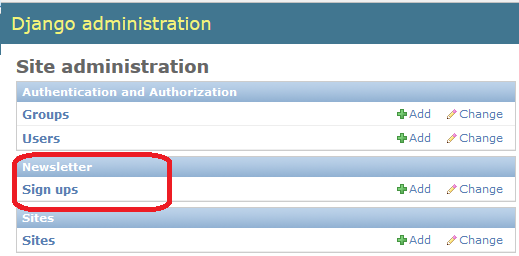
在Admin中注册SignUp

**from** django**.**contrib **import** admin

**from** **.**models **import** SignUp

admin**.**site**.**register**(**SignUp**,** SignUpAdmin**)**

model将会显示在admin中





**Customize admin**

**from** django**.**contrib **import** admin

# Register your models here.

**from** **.**models **import** SignUp

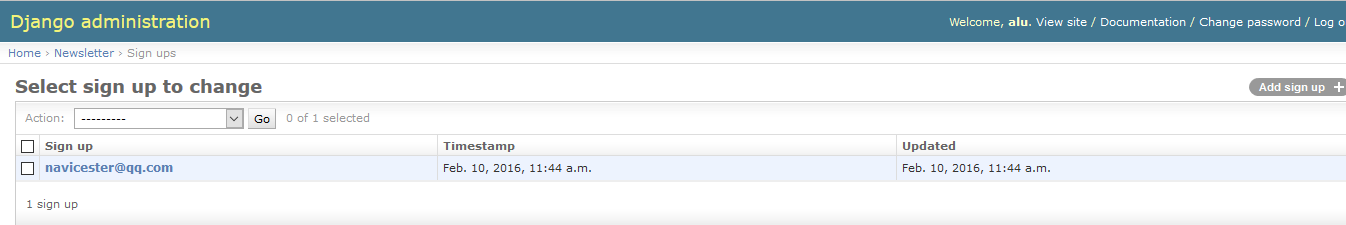
**class** **SignUpAdmin(**admin**.**ModelAdmin**):**

list\_display **=** **[**"\_\_unicode\_\_"**,** "timestamp"**,** "updated"**]**

class Meta:

model = SignUp

admin**.**site**.**register**(**SignUp**,** SignUpAdmin**)**



# Model Form

Newsletter下面添加文件forms.py

**from** django **import** forms

**from** **.**models **import** SignUp

**class** **SignUpForm(**forms**.**ModelForm**):**

**class** **Meta:**

model **=** SignUp

fields **=** **[**'full\_name'**,** 'email'**]**

### exclude = ['full\_name']

Modify admin.py

**from** django**.**contrib **import** admin

# Register your models here.

**from** **.**forms **import** SignUpForm

**from** **.**models **import** SignUp

**class** **SignUpAdmin(**admin**.**ModelAdmin**):**

list\_display **=** **[**"\_\_unicode\_\_"**,** "timestamp"**,** "updated"**]**

form **=** SignUpForm

# class Meta:

# model = SignUp

admin**.**site**.**register**(**SignUp**,** SignUpAdmin**)**

# Form Validation

给 “email” & “full\_name”添加validation函数

**class** **SignUpForm(**forms**.**ModelForm**):**

**class** **Meta:**

model **=** SignUp

fields **=** **[**'full\_name'**,** 'email'**]**

### exclude = ['full\_name']

**def** clean\_email**(**self**):**

email **=** self**.**cleaned\_data**.**get**(**'email'**)**

email\_base**,** provider **=** email**.**split**(**"@"**)**

domain**,** extension **=** provider**.**split**(**'.'**)**

# if not domain == 'USC':

# raise forms.ValidationError("Please make sure you use your USC email.")

**if** **not** extension **==** "edu"**:**

**raise** forms**.**ValidationError**(**"Please use a valid .EDU email address"**)**

**return** email

**def** clean\_full\_name**(**self**):**

full\_name **=** self**.**cleaned\_data**.**get**(**'full\_name'**)**

#write validation code.

**return** full\_name

# View and Template Context

**from** django**.**shortcuts **import** render

# Create your views here.

**def** home**(**request**):**

title **=** 'Welcome'

**if** request**.**user**.**is\_authenticated**():**

title = "My title is %s" % (request.user)

context **=** **{**

"title"**:** title**,**

**}**

**return** render**(**request**,** "home.html"**,** context**)**

home.html

<h1>{{title}}</h1>

{{user}}

{{request.user}}

这两个user分别是由下面的middleware传进来的

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

# Form in a view

<https://docs.djangoproject.com/en/1.8/ref/forms/>

**from** **.**forms **import** SignUpForm

**from** **.**models **import** SignUp

# Create your views here.

**def** home**(**request**):**

title **=** 'Welcome'

form **=** SignUpForm**(**request**.**POST **or** **None)**

context **=** **{**

"title"**:** title**,**

"form"**:** form

**}**

**if** form**.**is\_valid**():**

#form.save()

#print request.POST['email'] #not recommended, raw data without validation

instance **=** form**.**save**(**commit**=False)**

full\_name **=** form**.**cleaned\_data**.**get**(**"full\_name"**)**

**if** **not** full\_name**:**

full\_name **=** "New full name"

instance**.**full\_name **=** full\_name

# if not instance.full\_name:

# instance.full\_name = "Justin"

instance**.**save**()**

context **=** **{**

"title"**:** "Thank you"

**}**

**return** render**(**request**,** "home.html"**,** context**)**

form = SignUpForm(request.POST or None)这句话中记得加 or None，否则的话SignUpForm会一直执行Validation

即使只是网址GET访问，也会出现下列validation错误



调用form**.**is\_valid，form会执行form类里面的那些validation函数

form**.**save**(**commit**=False)**并不会真正的保存数据 ，instance**.**save**()**才会真正保存

This save() method accepts an optional commit keyword argument, which accepts either True or False. If you

call save() with commit=False, then it will return an object that hasn’t yet been saved to the database. In this

case, it’s up to you to call save() on the resulting model instance.

home.html

<h1>**{{title}}**</h1>

**{{user}}**

**{{request.user}}**

<form method=**"POST"** action=**''**> **{% csrf\_token%}**

**{{form.as\_p}}**

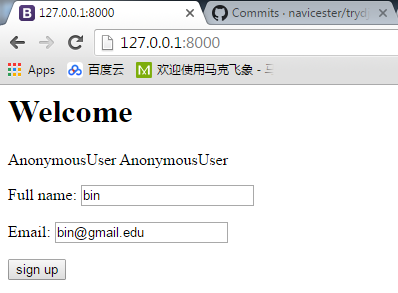
<input type=**"submit"** value=**"sign up"**>

</form>

action指定了提交之后的重定向地址，可以用”.”

as\_p = as paragraph

GET vs POST



Home函数里添加下列打印

print request

print request.POST

在shell里面查看打印输出

如果只是执行网址访问<http://127.0.0.1:8000/>

<WSGIRequest: GET '/'>

<QueryDict: {}>

如果按”sign up”提交

<WSGIRequest: POST '/'>

<QueryDict: {u'csrfmiddlewaretoken': [u'xcdCoiISxk5yS4GSbVHENmjWwnhvj7kk'], u'email': [u'bin@gmail.edu'], u'full\_name': [u'bin']}>

# Custom form in a view (NON ModelForm)

在forms.py中添加contactForm

**class** **ContactForm(**forms**.**Form**):**

full\_name **=** forms**.**CharField**(**required**=False)**

email **=** forms**.**EmailField**()**

message **=** forms**.**CharField**()**

在views.py中添加Contact函数

**def** contact**(**request**):**

form **=** ContactForm**(**request**.**POST **or** **None)**

if form.is\_valid():

# for key, value in form.cleaned\_data.iteritems():

# print key, value

# #print form.cleaned\_data.get(key)

context **=** **{**

"form"**:** form**,**

**}**

**return** render**(**request**,** "forms.html"**,** context**)**

在urls.py中添加入口

urlpatterns **=** patterns**(**''**,**

url**(**r'^$'**,** 'newsletter.views.home'**,** name**=**'home'**),**

url**(**r'^contact/$'**,** 'newsletter.views.contact'**,** name**=**'contact'**),**

**)**

创建forms.html文件

<form method=**'POST'** action=**''**>**{% csrf\_token %}**

**{{ form.as\_p }}**

<input type=**'submit'** value=**'Submit'** />

</form>

# Setup Email

<https://docs.djangoproject.com/en/1.8/topics/email/>

In settings.py

EMAIL\_HOST = 'smtp.sina.com'

EMAIL\_HOST\_USER = 'hebinn2004@sina.com'

EMAIL\_HOST\_PASSWORD = ''

EMAIL\_PORT = 25

EMAIL\_USE\_TLS = True

views.py

**from** django**.**conf **import** settings

**from** django**.**core**.**mail **import** send\_mail

**def** contact**(**request**):**

title **=** 'Contact Us'

title\_align\_center **=** **True**

form **=** ContactForm**(**request**.**POST **or** **None)**

**if** form**.**is\_valid**():**

# for key, value in form.cleaned\_data.iteritems():

# print key, value

# #print form.cleaned\_data.get(key)

form\_email **=** form**.**cleaned\_data**.**get**(**"email"**)**

form\_message **=** form**.**cleaned\_data**.**get**(**"message"**)**

form\_full\_name **=** form**.**cleaned\_data**.**get**(**"full\_name"**)**

# print email, message, full\_name

subject **=** 'Site contact form'

from\_email **=** settings**.**EMAIL\_HOST\_USER

to\_email **=** **[**from\_email**,** 'youotheremail@email.com'**]**

contact\_message **=** "%s: %s via %s"**%(**

form\_full\_name**,**

form\_message**,**

form\_email**)**

some\_html\_message **=** """

<h1>hello</h1>

"""

send\_mail**(**subject**,**

contact\_message**,**

from\_email**,**

to\_email**,**

html\_message**=**some\_html\_message**,**

fail\_silently**=True)**

context **=** **{**

"form"**:** form**,**

"title"**:** title**,**

"title\_align\_center"**:** title\_align\_center**,**

**}**

**return** render**(**request**,** "forms.html"**,** context**)**

# static file in django1.8

Static files (CSS, JavaScript, Images)

<https://docs.djangoproject.com/en/1.8/howto/static-files/>

STATIC\_URL **=** '/static/'

STATIC\_ROOT **=** os**.**path**.**join**(**os**.**path**.**dirname**(**BASE\_DIR**),** "static\_in\_env"**,** "static\_root"**)**

STATICFILES\_DIRS **=** **(**

os**.**path**.**join**(**BASE\_DIR**,** "static\_in\_pro"**,** "our\_static"**),**

#os.path.join(BASE\_DIR, "static\_in\_env"),

#'/var/www/static/',

**)**

MEDIA\_URL **=** '/media/'

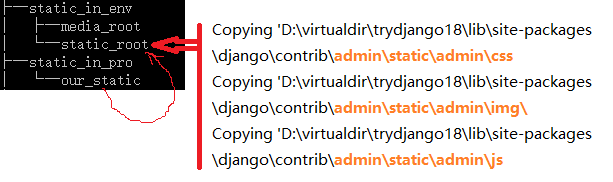
MEDIA\_ROOT **=** os**.**path**.**join**(**os**.**path**.**dirname**(**BASE\_DIR**),** "static\_in\_env"**,** "media\_root"**)**

STATIC\_URL : 服务器相对地址

MEDIA\_ROOT : 文件系统绝对路径，保存用户上传文件.

MEDIA\_URL: URL that handles the media served from MEDIA\_ROOT. Make sure to use a trailing slash.

执行 **python manage.py collectstatic**，STATICFILES\_DIRS中的文件将会拷贝到STATIC\_ROOT中



# Serving static files

**from** django**.**conf **import** settings

**from** django**.**conf**.**urls**.**static **import** static

**if** settings**.**DEBUG**:**

urlpatterns **+=** static**(**settings**.**STATIC\_URL**,** document\_root**=**settings**.**STATIC\_ROOT**)**

urlpatterns **+=** static**(**settings**.**MEDIA\_URL**,** document\_root**=**settings**.**MEDIA\_ROOT**)**

# adding bootstrap to django

Bootstrap CDN

CDN = content of network

The folks over at [MaxCDN](https://www.maxcdn.com/) graciously provide CDN support for Bootstrap's CSS and JavaScript. Just use these [Bootstrap CDN](https://www.bootstrapcdn.com/) links.

<!-- Latest compiled and minified CSS -->

<link rel=**"stylesheet"** href=**"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"** integrity=**"sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"** crossorigin=**"anonymous"**>

<!-- Optional theme -->

<link rel=**"stylesheet"** href=**"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css"** integrity=**"sha384-fLW2N01lMqjakBkx3l/M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV+0En5r"** crossorigin=**"anonymous"**>

<!-- Latest compiled and minified JavaScript -->

<script src=**"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"** integrity=**"sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS"** crossorigin=**"anonymous"**></script>

创建 base.html

拷贝该例子的源代码 <http://getbootstrap.com/examples/navbar-static-top/>

在文件头添加 CDN或者使用文件

将css/js/img文件下载到本地保存到static\_in\_pro

* bootstrap.min.css
* navbar-static-top.css
* bootstrap.min.js
* ie10-viewport-bug-workaround.js

将{% load staticfiles %} 添加到base.html开头，改变css/js/img的链接

**{% load staticfiles %}**

<!-- <link href="../../dist/css/bootstrap.min.css" rel="stylesheet"> -->

<link href=**"{% static 'css/bootstrap.min.css' %}"** rel=**"stylesheet"**>

<!-- <link href="navbar-static-top.css" rel="stylesheet"> -->

<link href=**"{% static 'css/navbar-static-top.css' %}"** rel=**"stylesheet"**>

**{% load staticfiles %}**

<script src=**"https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"**></script>

<script src=**"{% static 'js/bootstrap.min.js' %}"**></script>

<!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

<script src=**"{% static 'js/ie10-viewport-bug-workaround.js' %}"**></script>

执行 **python manage.py collectstatic**

# Django template Include, Inheritance, Blocks

**include**

将base.html里面<nav>…</nav>的代码块移到navbar.html里面

In base.html, move the navbar code <nav>…</nav> to navbar.html

在base.html里面添加include

And include it in base.html

**{% include 'navbar.html' %}**

verbatim可以保持原来的格式

{% include 'navbar.html' %}

{% verbatim %}

同样的方法，可以添加head\_css.html和javascript.html，别忘了在文件开始添加 load staticfile

**extents**

在 home.html中, 添加

**{% extends "base.html" %}**

**block**

将div jumbotron移到home.html，用block jumbotron标记

<http://getbootstrap.com/examples/jumbotron/>

**{% block "jumbotron" %}**

<!-- Main component for a primary marketing message or call to action -->

<div class=**"jumbotron"**>

<h1>**Navbar example**</h1>

<p>**This example is a quick exercise to illustrate how the default, static and fixed to top navbar work. It includes the responsive CSS and HTML, so it also adapts to your viewport and device.**</p>

<p>**To see the difference between static and fixed top navbars, just scroll.**</p>

<p>

<a class=**"btn btn-lg btn-primary"** href=**"../../components/#navbar"** role=**"button"**>**View navbar docs** &raquo;</a>

</p>

</div>

**{% endblock%}**

并且在 base.html中添加block jumbotron

**{% block "jumbotron" %}**

**{% endblock%}**

{{block.super}} 会显示super内容.

In base.html

**<title>{% block head\_title %}MVP Landing{% endblock %}</title>**

In home.html

**{% block head\_title %}Welcome | {{ block.super }}{% endblock %}**

# Third Party Library Crispy Form

<http://django-crispy-forms.readthedocs.org/en/latest/>

在当前python环境下安装最新稳定版本

**pip install --upgrade django-crispy-forms**

安装完成之后将安装信息更新到requirements.txt

**pip freeze > requirements.txt**

settings.py中，将crispy\_forms添加到INSTALLED\_APPS:

**INSTALLED\_APPS = (**

**...**

**'crispy\_forms',**

**)**

运行 **python manage.py makemigrations** 看看是否需要migrations

settings.py添加一些对应的配置信息

**#Crispy FORM TAGs SETTINGS**

**CRISPY\_TEMPLATE\_PACK = 'bootstrap3'**

在html文件修改，home.html

**{% load crispy\_forms\_tags %}**

**{{ form|crispy }}**

# Bootstrap Grid System

<http://getbootstrap.com/css/#grid>

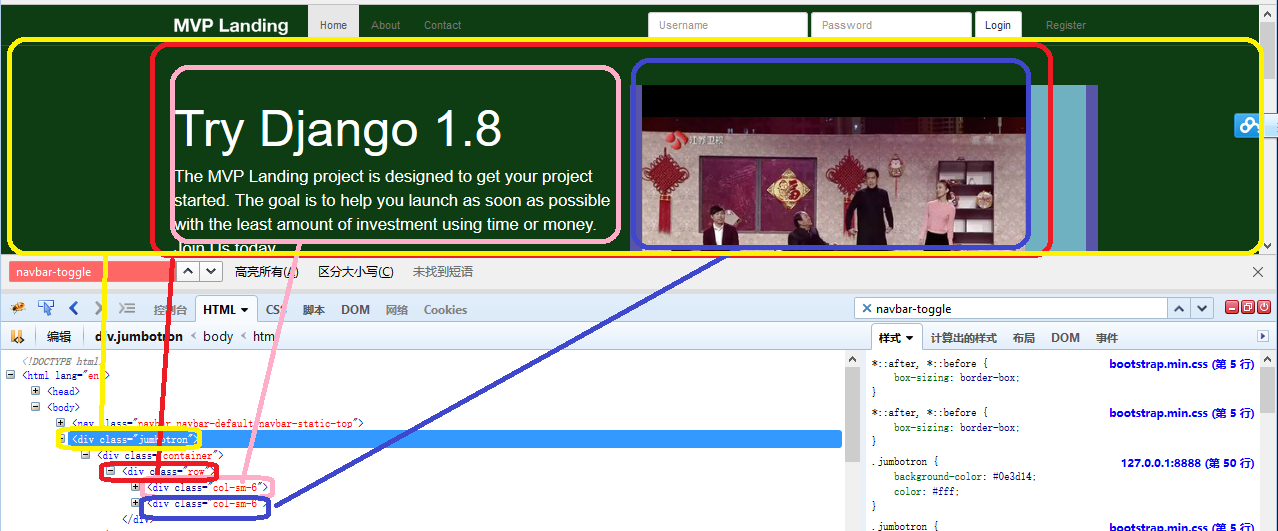
Class “container-fluid” 全屏

<div class="container"> 这个只显示中间部分，宽度不会全屏

Introduction

Grid systems are used for creating page layouts through a series of rows and columns that house your content. Here's how the Bootstrap grid system works:

* Rows must be placed within a .container (fixed-width) or .container-fluid (full-width) for proper alignment and padding.
* Use rows to create horizontal groups of columns.
* Content should be placed within columns, and only columns may be immediate children of rows.
* Predefined grid classes like .row and .col-xs-4 are available for quickly making grid layouts. Less mixins can also be used for more semantic layouts.
* Columns create gutters (gaps between column content) via padding. That padding is offset in rows for the first and last column via negative margin on .rows.
* The negative margin is why the examples below are outdented. It's so that content within grid columns is lined up with non-grid content.
* Grid columns are created by specifying the number of twelve available columns you wish to span. For example, three equal columns would use three .col-xs-4.
* If more than 12 columns are placed within a single row, each group of extra columns will, as one unit, wrap onto a new line.
* Grid classes apply to devices with screen widths greater than or equal to the breakpoint sizes, and override grid classes targeted at smaller devices. Therefore, e.g. applying any .col-md-\* class to an element will not only affect its styling on medium devices but also on large devices if a .col-lg-\* class is not present.





Tips

如果想要固定宽度fixed, 使用 col-xs-???, 否则当屏幕变得更小级别时，它会变成全屏.

例如，如果你使用col-sm-3, 在smal size或者更大宽度显示时，它会显示1/4（3/12）屏幕宽度, 但是如果转到xsmall屏幕时，它将会100%显示.

# Styling MVP Landing Part1

**{% block jumbotron %}**

<div class=**"jumbotron"**>

<div class=**"container"**>

<div class=**"row"**>

<div class=**'col-sm-6'**>

<h1>**Try Django 1.8**</h1>

<p>**The MVP Landing project is designed to get your project started. The goal is to help you launch as soon as possible with the least amount of investment using time or money. Join Us today.**</p>

<p>

<a class=**"btn btn-lg btn-primary"** href=**""** role=**"button"**>**Join Us** &raquo;</a>

</p>

</div>

**<div class='col-sm-6 style=”background-color:black;heigh:300px;”>**

**</div>**

</div>

</div>

</div>

**{% endblock %}**

**{% block content %}**

<div class=**"row"**>

<div class=**"col-xs-3 pull-right"**>

<h1>**{{title}}**</h1>

**{{user}}**

**{{request.user}}**

<form method=**"POST"** action=**''**> **{% csrf\_token%}**

**{{form|crispy}}**

<input type=**"submit"** value=**"sign up"**>

</form>

</div>

</div>

**{% endblock %}**

**{% block content %}**

<div class=**"row"**>

<div class=**" col-sm-3 pull-right"**>

<p class=**'lead text-align-center'**>**{{ title }}**</p>

<form method=**"POST"** action=**''**> **{% csrf\_token%}**

**{{ form|crispy }}**

<input type=**"submit"** value=**"sign up"**>

</form>

</div>

<div class=**'col-sm-3'**>

<p class=**'lead text-align-center'**>**Django** &amp; **Bootstrap**</p>

</div>

<div class=**'col-sm-3'**>

<p class=**'lead text-align-center'**>**Created for Starters**

<br/>

</p>

</div>

<div class=**'col-sm-3'**>

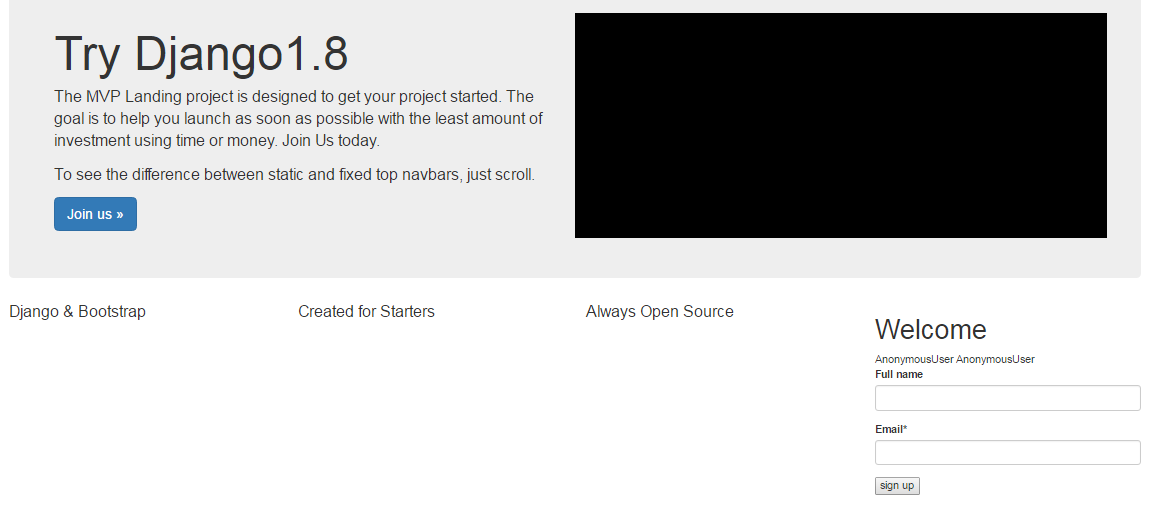
<p class=**'lead text-align-center'**>**Always Open Source** <br/><br/>

</p>

</div>

</div>

**{% endblock%}**



<div class="container-fluid">会全屏平铺显示

# CSS with Blocks

**Style**

In base.html

<style>

{% block style %}{% endblock %}

</style>

Extend in home.html

<style>

**{% block style %}**

**.text-align-center {**

**text-align: center;**

**}**

**.navbar-static-top {**

**margin-bottom: 0px !important;**

**}**

**{% endblock %}**

</style>

**text-align-center** can also be moved to css\custom.css

同时在head\_css.html添加

<link href="{% static 'css/custom.css' %}" rel="stylesheet">

**{% block content %}**

<div class=**"row"**>

<div class=**" col-sm-3 pull-right"**>

<p class=**'lead text-align-center'**>**{{ title }}**</p>

<form method=**"POST"** action=**''**> **{% csrf\_token%}**

**{{ form|crispy }}**

<input type=**"submit"** value=**"sign up"**>

</form>

</div>

<div class=**'col-sm-3'**>

<p class=**'lead text-align-center'**>**Django** &amp; **Bootstrap**</p>

</div>

<div class=**'col-sm-3'**>

<p class=**'lead text-align-center'**>**Created for Starters**

<br/>

</p>

</div>

<div class=**'col-sm-3'**>

<p class=**'lead text-align-center'**>**Always Open Source** <br/><br/>

</p>

</div>

</div>

**{% endblock%}**

# URL Names as Links

在trydjango18目录下面创建文件views.py

添加about函数

**from** django**.**shortcuts **import** render

**def** about**(**request**):**

**return** render**(**request**,** "about.html"**,** **{})**

在urls.py中添加url

urlpatterns **=** patterns**(**''**,**

url**(**r'^about/$'**,** 'trydjango18.views.about'**,** name**=**'about'**),**

**)**

在navbar.html添加about和contact相关的内容

<div id=**"navbar"** class=**"navbar-collapse collapse"**>

<ul class=**"nav navbar-nav"**>

<!--

<li class="active"><a href="#">Home</a></li>

<li><a href="/about">About</a></li>

<li><a href="/contact">Contact</a></li>

-->

<li class=**"active"**><a href=**"{% url 'home' %}"**>**Home**</a></li>

<li><a href=**"{% url 'about' %}"**>**About**</a></li>

<li><a href=**"{% url 'contact' %}"**>**Contact**</a></li>

可以用**{% url '???' %}**来代替显式的url

# Styling MVP Landing Part2

forms.html

* extend base.html
* crispy，并修改submit按钮class
* 添加div，调整宽度和增加title

{% extends "base.html" %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="row">

<div class ="col-sm-6 col-sm-offset-3">

{% if title %}

<h1 class = "{% if title\_align\_center %}text-align-center{% endif %}">{{title}}</h1>

{% endif %}

<form method='POST' action=''>{% csrf\_token %}

{{ form|crispy }}

<input class = "btn btn-primary" type='submit' value='Submit' />

</form>

</div>

</div>

{% endblock %}

newsletter\views.py

添加title和对齐的变量

**def** contact**(**request**):**

title **=** 'Contact Us'

title\_align\_center **=** **True**

……

context **=** **{**

"form"**:** form**,**

"title"**:** title**,**

"title\_align\_center"**:** title\_align\_center**,**

**}**

**return** render**(**request**,** "forms.html"**,** context**)**

# Django Registration Redux

<http://django-registration-redux.readthedocs.org/en/latest/quickstart.html>

**安装**

pip install django-registration-redux

**settings.py**

INSTALLED\_APPS **=** **(**

#third party apps

'registration'**,**

**)**

#DJANGO REGISTRATION REDUX SETTINGS

ACCOUNT\_ACTIVATION\_DAYS = 7

REGISTRATION\_AUTO\_LOGIN = True

SITE\_ID = 1

**urls.py**

urlpatterns **=** **[**

url**(**r'^accounts/'**,** include**(**'registration.backends.default.urls'**)),**

**]**

urlpatterns **=** **[**

url**(**r'^login/$'**,**

auth\_views**.**login**,**

**{**'template\_name'**:** 'registration/login.html'**},**

name**=**'auth\_login'**),**

url**(**r'^logout/$'**,**

auth\_views**.**logout**,**

**{**'template\_name'**:** 'registration/logout.html'**},**

name**=**'auth\_logout'**),**

url**(**r'^password/change/$'**,**

auth\_views**.**password\_change**,**

**{**'post\_change\_redirect'**:** reverse\_lazy**(**'auth\_password\_change\_done'**)},**

name**=**'auth\_password\_change'**),**

url**(**r'^password/change/done/$'**,**

auth\_views**.**password\_change\_done**,**

name**=**'auth\_password\_change\_done'**),**

url**(**r'^password/reset/$'**,**

auth\_views**.**password\_reset**,**

**{**'post\_reset\_redirect'**:** reverse\_lazy**(**'auth\_password\_reset\_done'**)},**

name**=**'auth\_password\_reset'**),**

url**(**r'^password/reset/complete/$'**,**

auth\_views**.**password\_reset\_complete**,**

name**=**'auth\_password\_reset\_complete'**),**

url**(**r'^password/reset/done/$'**,**

auth\_views**.**password\_reset\_done**,**

name**=**'auth\_password\_reset\_done'**),**

**]**

**migrations**

python manage.py makemigrations

python manage.py migrate

注册

http://127.0.0.1:8000/accounts/register/

**Templates**

从**django-registration-redux**拷贝 “registration” 目录到template

**Decorate** registration templates

registration\_form.html

**{% extends "base.html" %}**

**{% load i18n %}**

**{% load crispy\_forms\_tags %}**

**{% block content %}**

<div class=**'row'**>

<div class=**'col-sm-6 col-sm-offset-3'**>

<h1>**Register for free!**</h1>

<form method=**"post"** action=**"."**>

**{% csrf\_token %}**

**{{ form****|crispy }}**

<input class=**'btn btn-block btn-primary'** type=**"submit"** value=**"{% trans 'Join' %}"** />

</form>

</div>

</div>

<hr/>

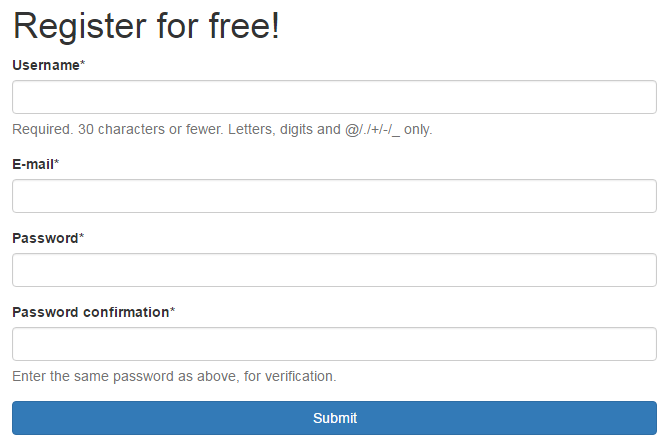
<div class=**'row'**>

<div class=**'col-sm-6 col-sm-offset-3 text-align-center'**>

<p>**Need to** <a href=**"{% url 'auth\_login' %}"**>**Login**</a>**?**</p>

</div>

</div>



注册之后在admin里面registration profiles里生成激活码



激活

activation\_email.html

下面这句话可以激活用户，activation\_key是参数

http://{{site.domain}}{% url 'registration\_activate' activation\_key %}

从registration.backends.default.urls中可以看出，registration\_activate对应的是

urlpatterns **=** **[**

url**(**r'^activate/complete/$'**,**

TemplateView**.**as\_view**(**template\_name**=**'registration/activation\_complete.html'**),**

name**=**'registration\_activation\_complete'**),**

# Activation keys get matched by \w+ instead of the more specific

# [a-fA-F0-9]{40} because a bad activation key should still get to the view;

# that way it can return a sensible "invalid key" message instead of a

# confusing 404.

url**(**r'^activate/(?P<activation\_key>\w+)/$'**,**

ActivationView**.**as\_view**(),**

name**=**'registration\_activate'**),**

手动运行http://127.0.0.1:8000/accounts/activate/688d2cb0b8c765b995c9025ff69924d1d474c96f即可激活用户

实际系统里，127.0.0.1:8000用换成site.domain指定的域名

Mail注册配置

DEFAULT\_FROM\_EMAIL = *'hebinn2004@sina.com'* 这个必须要有否则会报错from\_addr出错为webmaster@localhost

EMAIL\_HOST = *'smtp.sina.com'*

EMAIL\_HOST\_USER = DEFAULT\_FROM\_EMAIL

EMAIL\_HOST\_PASSWORD = *'\*\*\*\*\*\*'*

EMAIL\_PORT = 25

ACCOUNT\_ACTIVATION\_DAYS = 7

REGISTRATION\_AUTO\_LOGIN = True

SITE\_ID = 1

邮箱地址目前可以重复，后面得修改变成不能重复 （tag ?）

# Update django login URL to Custom URL

**settings.py**

LOGIN\_REDIRECT\_URL = *'/'*

REGISTRATION\_EMAIL\_SUBJECT\_PREFIX = *'[Django Registration trydjango18]'*

EMAIL\_BACKEND = *'django.core.mail.backends.smtp.EmailBackend'*

SEND\_ACTIVATION\_EMAIL = True

# Authentication Links in Navbar

navbar.html

<ul class="nav navbar-nav navbar-right">

{% if request.user.is\_authenticated %}

<li><a href="{% url 'auth\_logout' %}">Logout</a></li>

{% else %}

<li><a href="{% url 'registration\_register' %}">Register</a></li>

<li><a href="{% url ' auth\_login' %}">Login</a></li>

{% endif %}

</ul>

**registration\_register** is url name of register (**django-registration-redux**)

# Login Form in bootstrap Navbar

navbar.html

**{% if not request.user.is\_authenticated and not "/accounts/login" in request.get\_full\_path %}**

<form class=**'navbar-form navbar-right'** method=**'POST'** action=**'{% url "auth\_login" %}'**>**{% csrf\_token %}**

<div class=**'form-group'**>

<input type=**'text'** class=**'form-control'** name=**'username'** placeholder=**'Username'** />

</div>

<div class=**'form-group'**>

<input type=**'password'** class=**'form-control'** name=**'password'** placeholder=**'Password'** />

</div>

<button type=**'submit'** class=**'btn btn-default'**>**Login**</button>

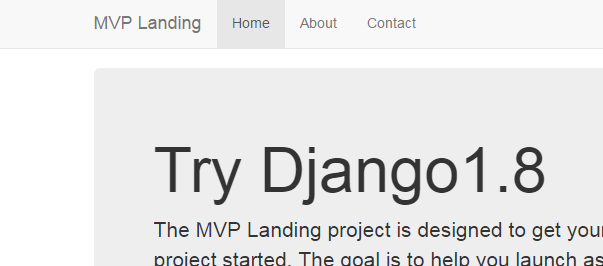
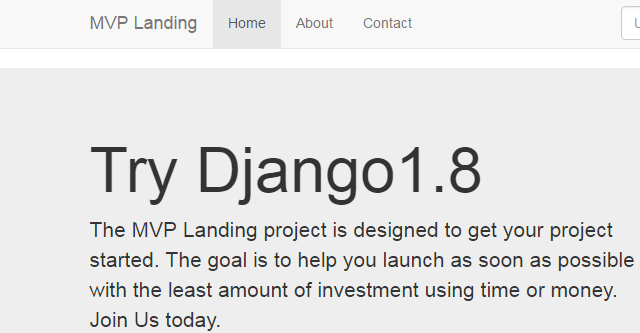
</form>

**{% endif %}**

这人action的link action=**'{% url "auth\_login" %}'**就是前面的Login，这儿的用户名和密码的name也是一致的

# Styling MVP Landing Part3

**效果1**

🡺

base.html,将jumbotron移到class container外面

<body>

**{% include 'navbar.html' %}**

**{% block container %}**

<div class=**"container"**>

**{% block jumbotron %}**

**{% endblock%}**

**{% block content %}**

**{% endblock%}**

</div><!-- /container -->

**{% endblock%}**

**{% include 'javascript.html' %}**

</body>

🡺修改为

<body>

**{% include 'navbar.html' %}**

**{% block container %}**

{% block jumbotron %}

{% endblock%}

<div class="container">

{% block content %}

{% endblock%}

</div> <!-- /container -->

{% endblock%}

{% include 'javascript.html' %}

</body>

home.html,在jumbotron里面插入container class

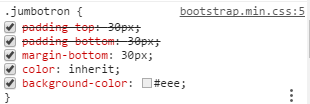
<div class="jumbotron">

<div class="container">

<div class="row">

**效果二**

修改jumbotron背景色



**.** jumbotron **{**

**background-color: #0E3D14 !important;**

**}**

**效果三**

设置navbar背景

**.**navbar-static-top **{**

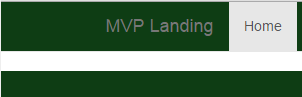
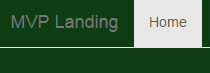
**background-color: #0E3D14;**

**}**

**效果四**

让navbar和jumbotron之间平稳连接

navbar和jumbotron之间的边框减少

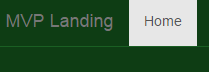
 🡺 

**.**navbar-static-top **{**

**margin-bottom: 0px;**

**}**

设置边框颜色，颜色变浅一点



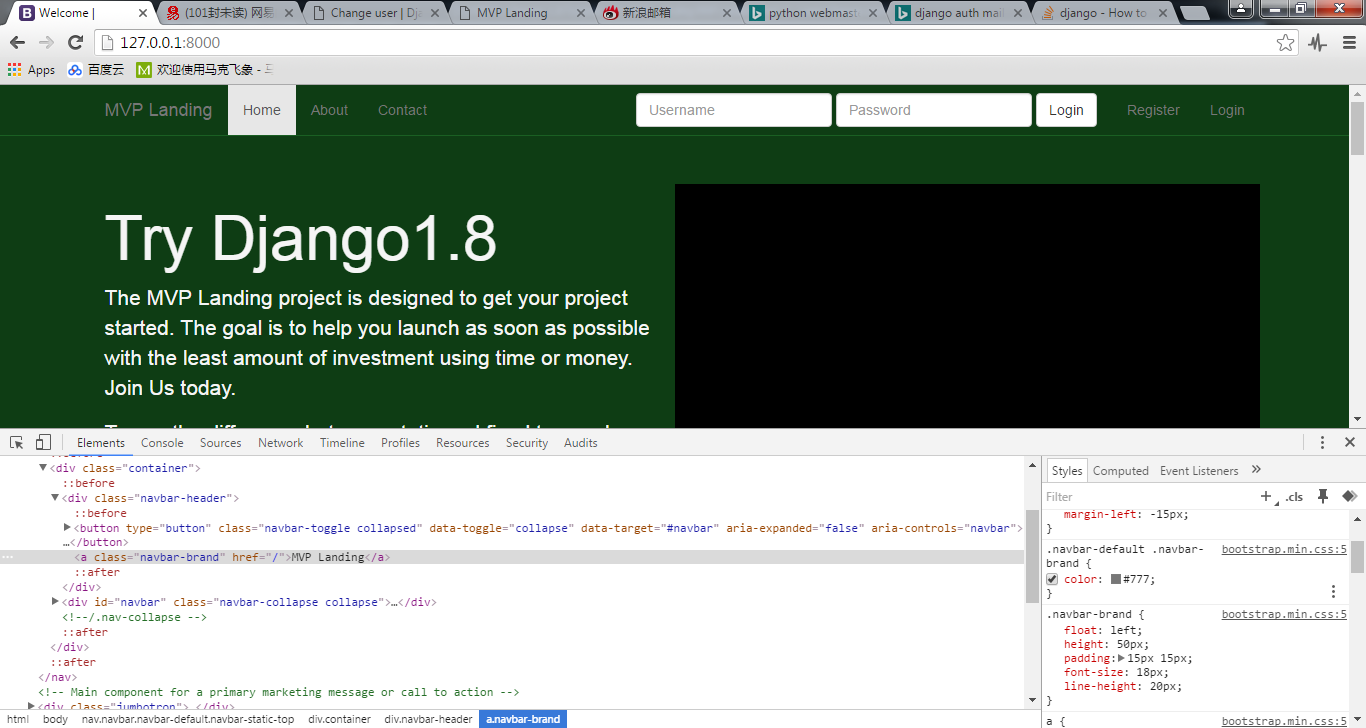
**.**navbar-static-top **{**

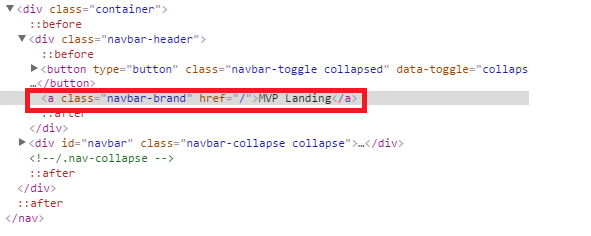
**border-color: #1A5F23;**

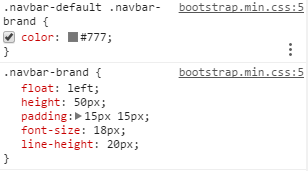
**}**

**效果五**

修改brand链接颜色，添加hover选项







**.**navbar-default **.**navbar-brand **{**

**color: #777;**

**}**

**.**navbar-default **.**navbar-brand**:hover** **{**

**color: #FFF;**

**}**

**效果六**

修改nav link的hover颜色 Home Contact About

**.**navbar-default **.**navbar-nav**>**li**>**a**:hover** **{**

**color: #999;**

**}**

**效果七**

修改toggle按钮属性

**.**navbar-default **.**navbar-toggle **{**

**border-color: transparent !important;**

**}**

**.**navbar-toggle**:hover** **{**

**background-color: #1A5F23 !important;**

**}**

**.**navbar-toggle **{**

**background-color: #0E3D14 !important;**

**}**

In navbar-static-top.css

body **{**

**min-height: 1000px;**

**}**

在navbar.html中对应的几个class的定义位置

<nav class=**"navbar navbar-default navbar-static-top"**>

<div class=**"container"**>

<div class=**"navbar-header"**>

<button type=**"button"** class=**"navbar-toggle collapsed"** data-toggle=**"collapse"** data-target=**"#navbar"** aria-expanded=**"false"** aria-controls=**"navbar"**>

<span class=**"sr-only"**>**Toggle navigation**</span>

<span class=**"icon-bar"**></span>

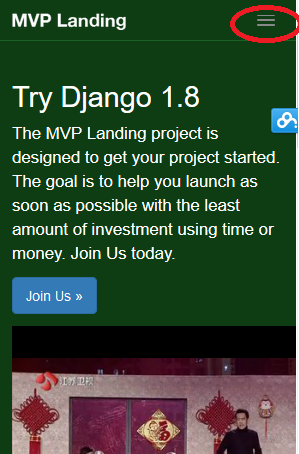
<span class=**"icon-bar"**></span>

<span class=**"icon-bar"**></span>

</button>

<a class=**"navbar-brand"** href=**"{% url 'home' %}"**><img src=**"{% static 'img/mvp\_landing\_logo.png' %}"** /></a>

Button is for small screen



# Logo image in navbar

图片高度23

**{% load staticfiles %}**

<a class=**"navbar-brand"** href=**"{% url 'home' %}"**><img src=**"{% static 'img/mvp\_landing\_logo.png' %}"** /></a>

# Promo Video and Image



home.html

添加视频

<div class='col-sm-6 video'>

<embed width="560" height="315" src= "http://player.youku.com/player.php/Type/Folder/Fid/26679028/Ob/1/sid/XMTQ2Nzg5NjM4NA==/v.swf" quality="high" width="480" height="400" align="middle" allowScriptAccess="always" allowFullScreen="true" mode="transparent" type="application/x-shockwave-flash"></embed>

</div>

添加图片

<div class='col-sm-3'>

<p class='lead text-align-center'>Django &amp; Bootstrap</p>

<img src='{% static "img/marketing1.jpg" %}' class='img-responsive' />

</div>

# Icon with Font Awesome

<http://fortawesome.github.io/Font-Awesome/>

<http://fortawesome.github.io/Font-Awesome/get-started/>

<http://fontawesome.io>

**CDN**

In head\_css.html

<link rel=**"stylesheet"** href=**"//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.css"**>

home.html

<div class=**'col-sm-3'**>

<p class=**'lead text-align-center'**>**Created for Starters** <br/>

<span class=**'fa-stack fa-lg fa-5x'**>

<i class=**'fa fa-comment-o fa-stack-2x'**></i>

<i style=**'color:yellow;'** class=**"fa fa-bolt fa-stack-1x"**></i>

</span>

</p>

</div>

<div class=**'col-sm-3'**>

<p class=**'lead text-align-center'**>**Always Open Source** <br/><br/>

<span class=**"fa-stack fa-lg fa-5x"**>

<i class=**"fa fa-circle-o-notch fa-stack-2x fa-spin "**></i>

<!-- <i class="fa fa-square-o fa-stack-2x"></i> -->

<i class=**"fa fa-github fa-stack-1x "**></i>

</span>

</p>

</div>

旋转 **fa-spin**

# Content for Authenticated Users

添加about.html

home.html

{% block jumbotron %}

{# % if not request.user.is\_authenticated % #}

<div class="jumbotron">

</div>

{# % endif % #}

{% endblock %}

{% block content %}

{% if request.user.is\_authenticated %}

<h1>You are logged in!</h1>

{% queryset %}

{% else %}

{% endif %}

view.py

if request.user.is\_authenticated() and request.user.is\_staff:

context = {

"queryset" : [123,456]

}

# QuerySetBasics

<https://docs.djangoproject.com/en/1.8/topics/db/queries/>

<https://docs.djangoproject.com/en/1.8/ref/models/querysets/>

views.py

**def** home**(**request**):**

**if** request**.**user**.**is\_authenticated**()** **and** request**.**user**.**is\_staff**:**

#print(SignUp.objects.all())

# i = 1

# for instance in SignUp.objects.all():

# print(i)

# print(instance.full\_name)

# i += 1

queryset **=** SignUp**.**objects**.**all**().**order\_by**(**'-timestamp'**)** #.filter(full\_name\_\_iexact="Justin")

#print(SignUp.objects.all().order\_by('-timestamp').filter(full\_name\_\_iexact="Justin").count())

context **=** **{**

"queryset"**:** queryset

**}**

home.html

{% if queryset %}

<h2>Welcome Staff</h2>

<table class='table'>

{% for instance in queryset %}

<tr>

<td>{{ forloop.counter }}</td>

<td>{{ instance.email }}</td>

<td>{{ instance.full\_name }}</td>

<td>{{ instance.timestamp|timesince }} ago</td>

</tr>

{% if instance.email == "abc@gmail.edu" %}

<tr><td>is equal</td></tr>

{% endif %}

{% endfor %}

</table>

{% endif %}

# Setting for different Environment

<https://www.webfaction.com/>

remove settings.py

add folder “settings”, and file \_\_init\_\_.py, base.py, local.py, product.py

in \_\_init\_\_.py

**from** **.**base **import** **\***

**try:**

**from** **.**local **import** **\***

**except:**

**pass**

**try:**

**from** **.**production **import** **\***

**except:**

**pass**

**try:**

**from** **.**imac **import** **\***

**except:**

**pass**

**try:**

**from** **.**macbookpro **import** **\***

**except:**

**pass**

# Using SSH with WebFaction Secure Shell

# Using SSH with WebFaction Secure Shell

# FTP for Production Settings

# Install PIP+ Migrate + Static Settings

# Fix Static Settings

# Final One