

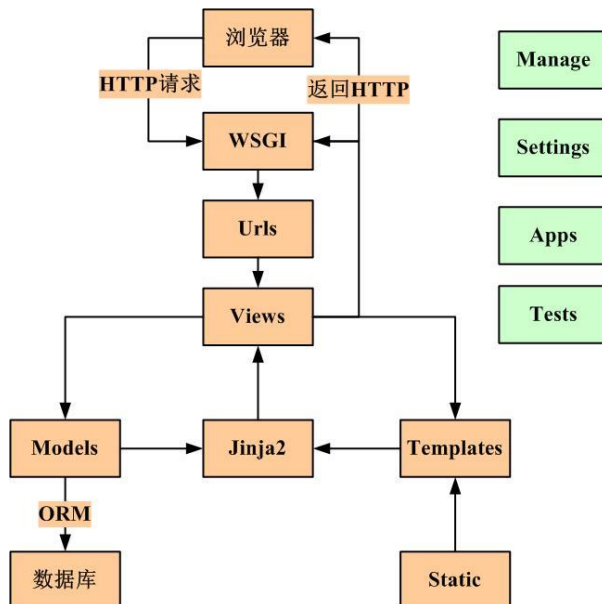
基于 Python 的 Django 架构学习报告

16302010050 马兵

Django 是一个开放源代码的 Web 应用框架，由 Python 写成，根据比利时的爵士音乐家 Django Reinhardt 命名。采用的是 MVT 的框架模式，即模型 M，视图 V 和模版 T，和以前学习过的 MVC 模式本质上是一样。

1. Django 的 MTV 模型组织

学习 Django，首先要了解一下它的组件，目录及其耦合关系，如下图：



工作流程：

1. 用 `manage .py runserver` 启动 Django 服务器时就载入了在同一目录下的 `settings .py`。该文件包含了项目中的配置信息，如前面讲的 `URLCONF` 等，其中最重要的配置是 `ROOT_URLCONF`，它告诉 Django 哪个 Python 模块应该用作本站的 `URLCONF`，默认的是 `urls .py`;
2. 当访问 url 的时候，Django 会根据 `ROOT_URLCONF` 的设置来装载 `URLCONF`;

3.然后按顺序逐个匹配 URLConf 里的 URLpatterns。如果找到则会调用相关联的视图

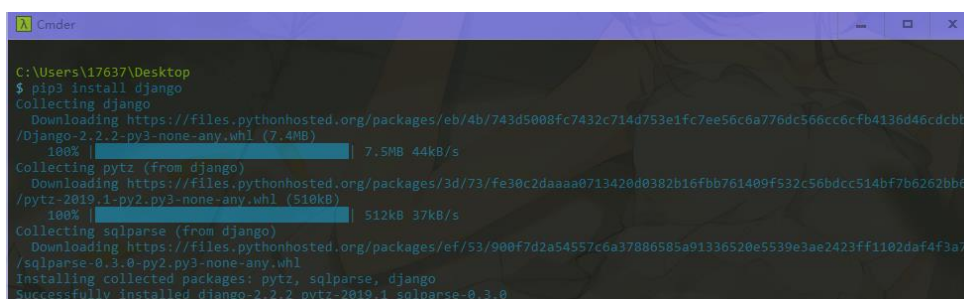
函数即 views，并把 HttpRequest 对象作为第一个参数(通常是 request)；

4.最后该 view 函数负责返回一个 HttpResponse 对象或者指向一个 html 页面。

2. Django 开发环境的配置

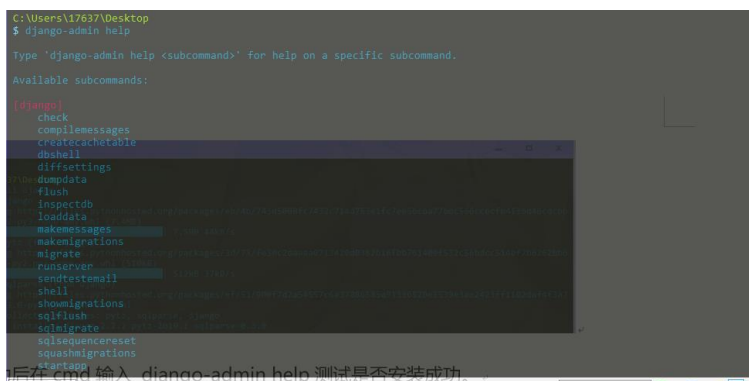
2.1 Python 环境和 Pycharm 安装比较简单不赘述；

2.2 pip3 安装，在 cmd 直接输入 pip3 install Django 即可安装最新版



```
C:\Users\17637\Desktop
$ pip3 install django
Collecting django
  Downloading https://files.pythonhosted.org/packages/eb/4b/743d5008fc7432c714d753e1fc7ee56c6a776dc566cc6cfb4136d46dcdbb/Django-2.2.2-py3-none-any.whl (7.4MB)
    100% |#####| 7.5MB 44kB/s
Collecting pytz (from django)
  Downloading https://files.pythonhosted.org/packages/3d/73/fe30c2daaaa0713420d0382b16fbb761409f532c56bdcc514bf7b6262bb6/pytz-2019.1-py2.py3-none-any.whl (510kB)
    100% |#####| 512kB 37kB/s
Collecting sqlparse (from django)
  Downloading https://files.pythonhosted.org/packages/ef/53/900f7d2a54557c6a37886585a91336520e5539e3ae2423ff1102daf4f3a7/sqlparse-0.3.0-py2.py3-none-any.whl
Installing collected packages: pytz, sqlparse, django
Successfully installed django-2.2.2 pytz-2019.1 sqlparse-0.3.0
```

安装成功后在 cmd 输入 django-admin help 测试是否安装成功，成功应该如下：



```
C:\Users\17637\Desktop
$ django-admin help

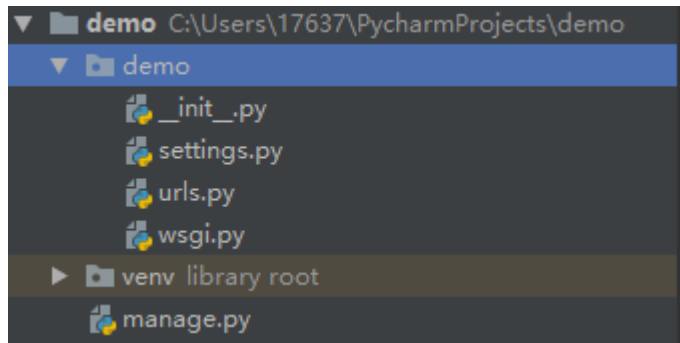
Type 'django-admin help <subcommand>' for help on a specific subcommand.

Available subcommands:

(django)
check
compilemessages
createsuperuser
dbshell
diffsettings
dumpdata
flush
inspectdb
loaddata
makemessages
migrate
migrate --help
runserver
sendtestemail
shell
showmigrations
sqlflush
sqlmigrate
sqlsequencereset
squashmigrations
startapp
```

3. Django 项目 demo 开发

3.1 在 Pycharm 创建 Django 项目，成功后目录结构应该包含如下文件：



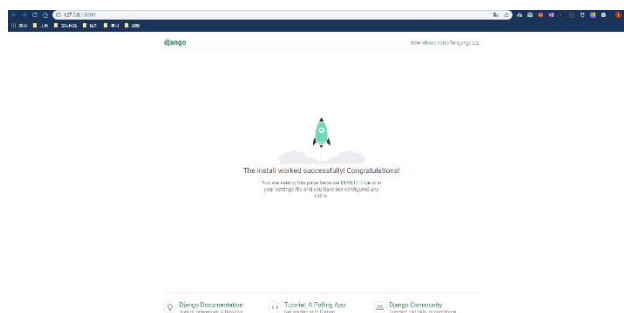
setting.py : 主要配置文件

urls.py : 路由文件

wsgi.py : 网络通信文件

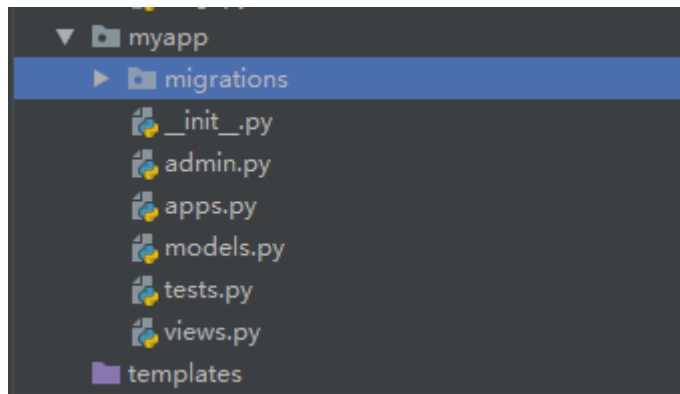
manage.py : Django 主管理程序

3.2 创建成功后我们直接尝试运行一下，运行起来后在浏览器输入 <http://127.0.0.1:8000/>:



3.3 创建自己的 app

在每个 django 项目中可以包含多个 APP，相当于一个大型项目中的分系统、子模块、功能部件等等，相互之间比较独立，但也有联系。所有的 APP 共享项目资源。在 pycharm 下方的 terminal 终端中输入命令：python manage.py startapp 加上 app 的 name 即可创建。创建成功后多了一个 myapp 文件夹



3.4 路由映射

3.4.1 首先要导入对应 app 的 view 文件，并且用 path 函数创建映射

```
from myapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('index/', views.index),
]
```

3.4.2 编写 views 的业务逻辑

导入 HttpResponse 模块，创建一个 index 函数，上一步我们已经把 urls 指向了这个函数，因此在浏览器访问这个 url 时会得到这个函数的返回值，因为 django 不允许直接返回 String，所以使用 HttpResponse 封装一下返回值。

```
from django.shortcuts import HttpResponse

# Create your views here.

def index(request):
    # request.GET
    return HttpResponse("hello world!")
```

3.4.3 这样一个简单的 demo 已经完成, 尝试运行后浏览器输入 <http://127.0.0.1:8000/index>



3.5 完成一个较完整的 web 项目

一个较为完整的 web 项目当然不是返回一个 String, 而是 html 文件, 在 django 项目想的 templates 存放的就是和 html 界面相关的文件。

3.5.1 在该目录下新建一个 index.html, 我们在 html 中添加一个简单的表单用于交互即可, 在此之前我们还可以创建一个 static 文件夹保存我们的前端其余两大文件 js 和 css。

在 index.html 中还有一个表格用于展示注册用户, 其中 django 采用 jinja2 语言编写动态模板, jinja2 会根据提供的数据, 替换掉 html 中的相应部分, 这里使用了 for 循环的

语句展示所有用户, 而关键参数 data 则是从 views 传过来的数据参数

```
<body>
<h2>用户注册</h2>
<form action="/index/" method="post">
  <input type="text" name="username">
  <input type="text" name="password">
  <button type="submit">提交</button>
</form>
<hr>
<h2>所有用户</h2>
<table border="1">
  <thead>
    <th>用户名</th>
    <th>密码</th>
  </thead>
  <tbody>
    {% for line in data %}
      <tr>
        <td>{{ line.username }}</td>
        <td>{{ line.password }}</td>
      </tr>
    {% endfor %}
  </tbody>
</table>
</body>
```

3.5.2 修改 views.py

修改 index 函数, 把 post 的数据存储到一个 list 里面, 并且以 data 参数发送到 html 页面

```

# Create your views here.

user_list = [
    {"username": "admin", "password": "123456"}
]

def index(request):
    # request.POST
    # return HttpResponse("hello world!")
    if request.method == "POST":
        username = request.POST.get("username", None)
        password = request.POST.get("password", None)
        temp = {"username": username, "password": password}
        print(username, password)
        user_list.append(temp)
    return render(request, "index.html", {"data": user_list})

```

最后初步运行结果如下：



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/index/". Below the address bar is a navigation bar with icons for application, tools, website, entertainment, learning, and login. The main content area has a heading "用户注册" (User Registration) followed by a form with two input fields and a "提交" (Submit) button. Below the form is a heading "所有用户" (All Users) followed by a table listing users.

用户名	密码
admin	123456
anxin	123456
ma	123456

3.6 使用数据库

Django 框架的 MTV 模式到这里只剩下 model 相关的数据库了，django 通过自带的 ORM 框架操作数据库，并且自带轻量级的 sqlite3 数据库，接下来为 demo 加入数据库以持久化保存数据。

3.6.1 注册 app

在 setting.py 中的 INSTALLED_APPS 中注册我们创建的 myapp，以便于数据库找到我们的 app

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'myapp',  
]
```

3.6.2 在 setting.py 中还可以找到 django 自带的轻量级数据库 sqlite3，做 demo 的话已经够用，所有这部分不用修改。

```
# Database  
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases  
  
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),  
    }  
}
```

3.6.3 修改 model.py

创建一个 UserInfo 类继承 models.Model，这个 UserInfo 相当于一张表，然后在该类下创建两个字段 username 和 password

```
class UserInfo(models.Model):  
    username = models.CharField(max_length=32)  
    password = models.CharField(max_length=32)
```

然后在 terminal 依次运行两条指令既可创建表：

1. python manage.py makemigrations
2. python manage.py migrate

3.6.4 修改 views.py

新增数据库后，我们就不用 user_list 保存数据，直接使用数据库操作得到数据，重新运行后注册过的用户也还存在。而且由于 django 自带的 ORM 框架操作数据库，我们并不需要

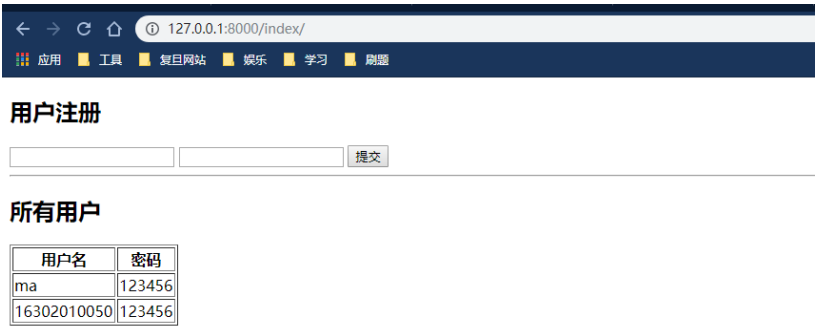
自己去写 sql 语句。

```
from django.shortcuts import render
from myapp import models

# Create your views here.

def index(request):
    # request.POSTd
    # return HttpResponse("hello world!")
    global user_list
    if request.method == "POST":
        username = request.POST.get("username", None)
        password = request.POST.get("password", None)
        # 插入一条数据
        models.UserInfo.objects.create(username=username, password=password)
        print(username, password)
        # 从数据库读取所有数据
        user_list = models.UserInfo.objects.all()
    return render(request, "index.html", {"data": user_list})
```

3.6.5 最终项目 demo 的运行效果如下：



总结：django 是一个功能强大，内容全面的 python web 框架，感觉和 spring boot 2.0 比较相似，并且基于 Python 语言，开发效率和可读性高，项目目录结构比较清晰，各模块分工明确，比较适合 web 框架入门学习。

层次	职责
模型（Model），即数据存取层	处理与数据相关的所有事务： 如何存取、如何验证有效性、包含哪些行为以及数据之间的关系等。
模板(Template)，即表现层	处理与表现相关的决定： 如何在页面或其他类型文档中进行显示。

视图（View），即业务逻辑层	存取模型及调取恰当模板的相关逻辑。模型与模板的桥梁。
-----------------	----------------------------