本文主要来自于mooc公开课Django课程

首先创建一个Django的虚拟环境：

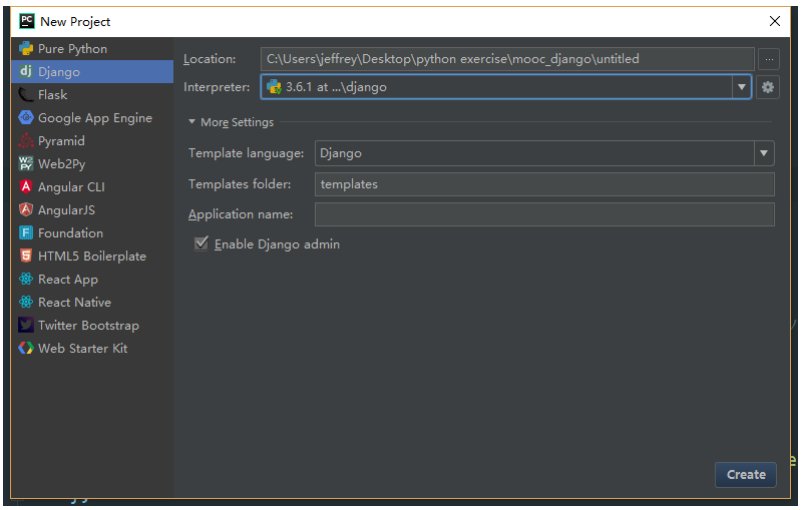
`pip install virtualenv`，然后使用`virtualenv 虚拟环境名`建立一个虚拟环境，在windows下面的激活方法是

```
cd django_virtual
cd Script
activate
```

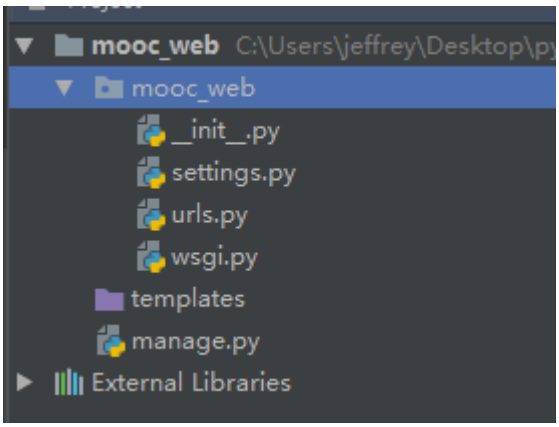
在linux下只需要将最后一步换位source activate

建立Django项目

在PyCharm建立程序时，直接选择Django程序，选择解释器为virtualenv建立的django环境，即可建立django项目



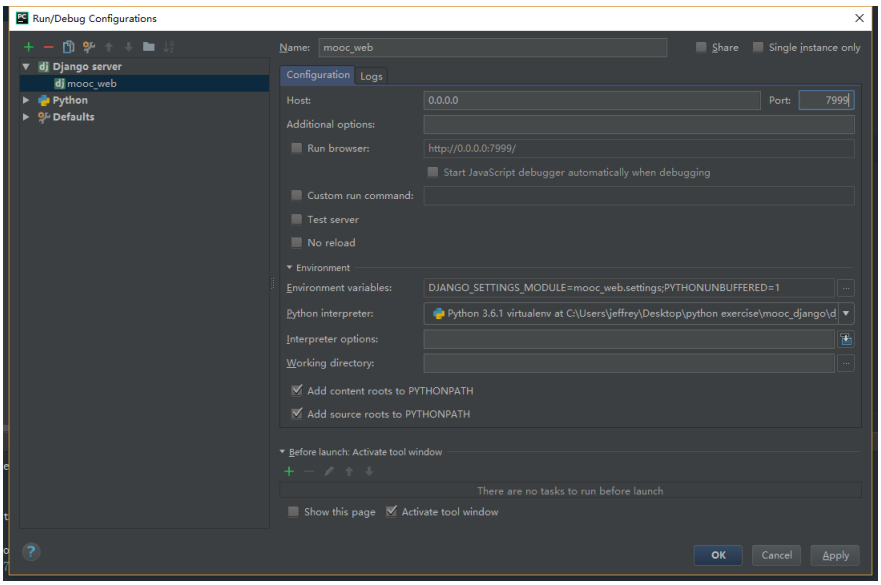
项目建立之后，我们得到了如下的目录结构，紫色的被标记为了templet目录，这样就能智能提示，如果你想要在import的时候不需要加入绝对路径，那么就可以`mark as source root`



接下来我们先尝试运行该项目，点击run，运行这个Django项目，可以看到运行在了本机的8000端口，点击该地址即可访问：

```
You have 13 unapplied migration(s). Your project may not work properly
Run 'python manage.py migrate' to apply them.
December 13, 2017 - 19:52:32
Django version 1.11.7, using settings 'mooc_web.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

如果要配置监听所有ip，那么我们需要通过`run-Edit configurations`，将监听端口改为0.0.0.0



Navicat使用

安装好Navicat之后，打开软件，建立连接，之后建立表，设计表的字段，设计完之后按`ctrl+s`保存，之后就可以插入数据条目，这就是简单的使用方法，具体使用会在之后用到的时候详细介绍。

Django目录结构

首先看看之前提到的建立django之后的目录结构，windows使用`tree \F .`得到如下结构，linux中使用`apt-get install tree`，然后`tree .即可得到`：

```
├mooc_web
  │  manage.py
  ├─mooc_web
  │  │  settings.py
  │  │  urls.py
  │  │  wsgi.py
  │  │  __init__.py
  └─templates
```

- templetes文件夹主要放html文件
- 主文件夹下面有settings.py，以及manage.py
- 我们需要建立static文件夹，用于存放css和js文件，以及主要的图片文件
- 建立log文件夹，用于存放日志
- 建立media文件夹，用于存放用户上传的文件
- 建立apps文件夹，用于存放各个app

建立app

通过pycharm当中的`tools-run manage.py task`，此时可以在shell中执行Django命令：

```
startapp message
```

此时就有了一个名为message的文件夹，将其拖入apps文件夹，自动生成了一个`__init__.py`文件，这样就让apps变成一个可以导入的包，为了方便导入，那么我们需要将apps文件夹remark成source root，这样每次引用的时候就不需要import apps.message.view 而是可以直接 import message.view

这样在pycharm中引用变得方便了，但是在使用命令行运行的时候就会出错，此时我们需要在settings.py中将apps加入到**根搜索路径**（BASE_DIR）中

```
├── apps
│   ├── __init__.py
│   └── message
│       ├── admin.py
│       ├── apps.py
│       ├── __init__.py
│       ├── migrations
│       │   └── __init__.py
│       ├── models.py
│       ├── tests.py
│       └── views.py
├── db.sqlite3
├── log
├── manage.py
├── media
├── mooc_web
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── static
└── templates
```

使用模板

将html模板放入template文件夹，在static文件夹中建立css文件夹，并放入style.css文件

settings配置

更改数据库配置

因为Django默认用的是sqlite数据库，我们要改成mysql数据库，打开项目的settings.py，找到DATABASES，将其中的：

- ENGINE改为`django.db.backends.mysql`
- NAME改为Navicat中看到的`testDjango`
- USER改为数据库的user名，我这里是root
- PASSWORD改为数据库的password，我这里也是root

- HOST改为localhost，也就是127.0.0.1

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'testDjango',
        'USER': 'root',
        'PASSWORD': 'root',
        'HOST': '127.0.0.1'
    }
}
```

migration生成数据表

通过`tools-Run manage.py Task`来连接数据库，首先需要安装mysqlclient，用`pip install mysqlclient`安装：

```
> makemigrations   # 用于生成一个基于你对模型改变的迁移，在这里就是数据库类型从sqlite迁移到mysql
> migrate          # 用于应用改变
```

此时Django自动生成了一大堆数据库表，在Navicat中可以看到表的名称：

此时可以点击run运行整个系统，可以在127.0.0.1:8000上访问该网址

配置static文件夹地址

但是此时的style.css无法被找到，我们要在settings.py中配置一下static文件夹的地址，因为`STATICFILES_DIRS`可能不止一个，所以用list的形式进行赋值，其中BASE_DIR是项目文件的根目录：

```
STATICFILES_DIRS = [
    os.path.join(BASE_DIR,'static')
]
```

urls.py配置和views.py函数

在urls.py中添加新的页面，页面的处理函数要在`apps.message.views.py`中新增，首先我们在`view.py`中构造如下函数，直接return render的模板，其中request参数是Django的请求，每个views函数都得有：

```
def getform(request):
    return render(request, template_name='course-comment.html')
```