

# Django框架基础

---



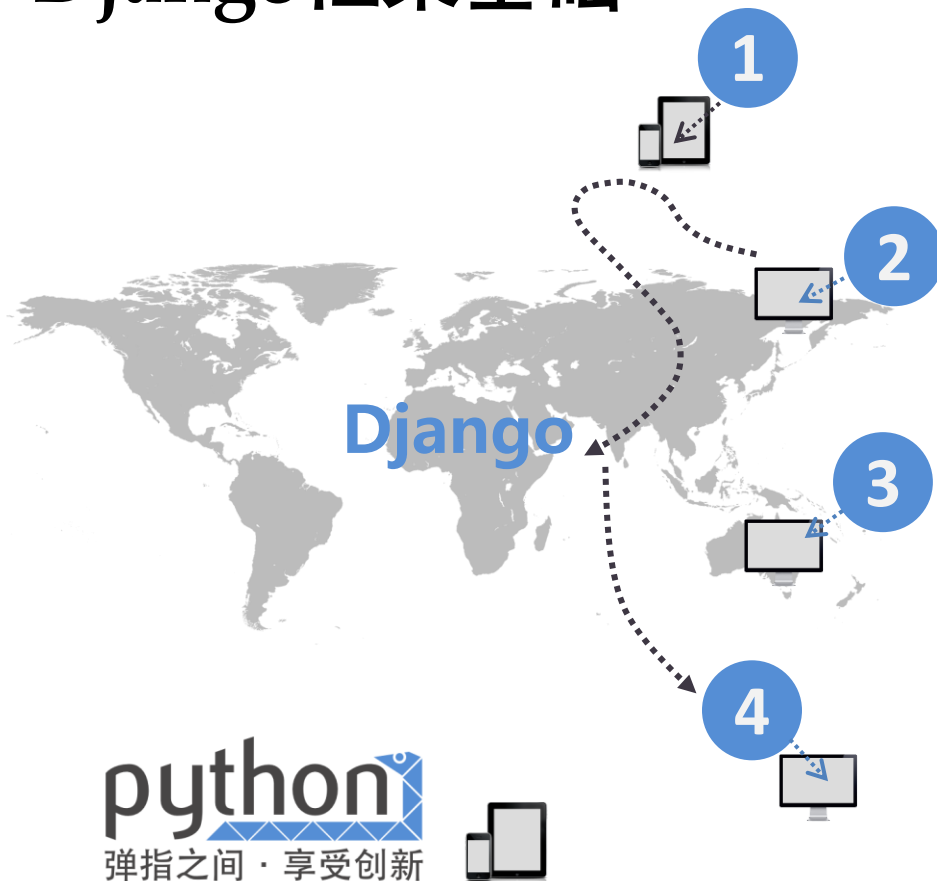
嵩 天  
北京理工大学





# 单元开篇

# Django框架基础



1 Django简介与安装

2 Django框架的最小程序

3 Django的MVT开发模式

实例1: 云端留言板之基本框架



# Django简介与安装

<https://www.djangoproject.com>

Django makes it easier to build better Web  
apps more quickly and with less code.

Get started with Django

## Meet Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Download latest release: 2.0

[DJANGO DOCUMENTATION >](#)

Support Django!



Anders Innovations Ltd donated to

# Django库的安装

Win平台: “以管理员身份运行” cmd , 执行 `pip install django`

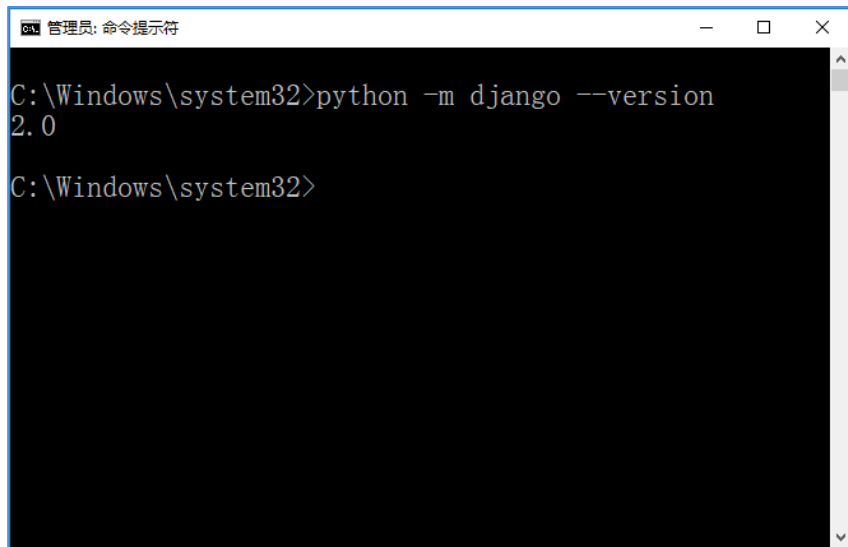
```
管理员: 命令提示符
Microsoft Windows [版本 10.0.15063]
(c) 2017 Microsoft Corporation。保留所有权利。

C:\Windows\system32>pip install django
```

```
管理员: 命令提示符 - pip install django
C:\Windows\system32>pip install django
Collecting django
  Downloading Django-2.0-py3-none-any.whl (7.1MB)
    23% |██████████| 20kB 56kB/s
```

# Django库的小测和版本

cmd命令行执行： `python -m django --version`



```
管理员: 命令提示符
C:\Windows\system32>python -m django --version
2.0
C:\Windows\system32>
```

Django 2.x版本

2017年12月2日发布

相比之前的版本有较大改动

确认Python 3.x和Django 2.x版本

# Django的理解

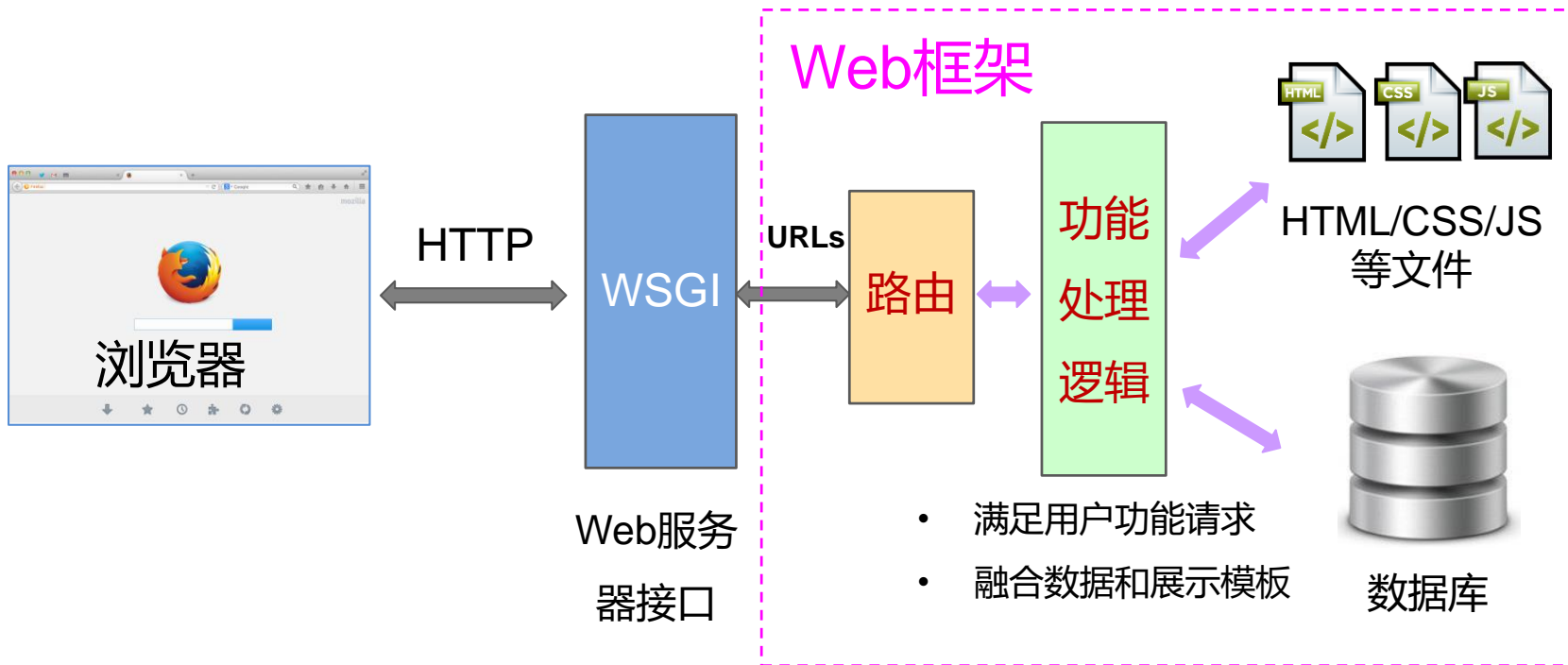
- Django是一个Web框架
- Django是一个产品级框架，支撑全球近万个网站及云端
- 采用MTV（ Model-Template-View ）模型组织
- 相比其他框架，Django充分利用Python特点，开发效率更高
- Django框架的官方文档：

<https://docs.djangoproject.com/en/2.0/>



# Web框架

Web框架是一个建设Web应用的半成品

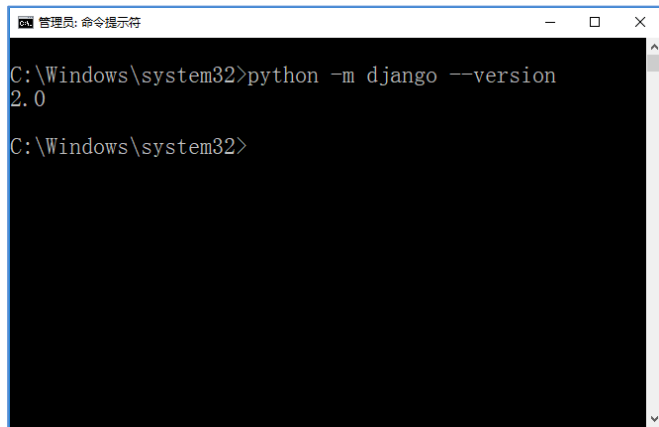
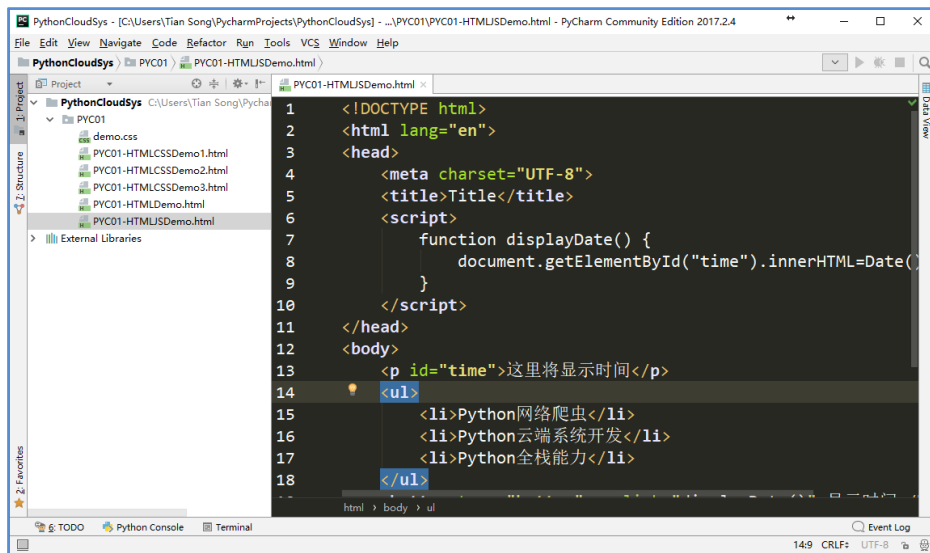




# Django框架的最小程序

# Django的开发工具

免费但够用的工具：PyCharm社区版 + Windows命令行

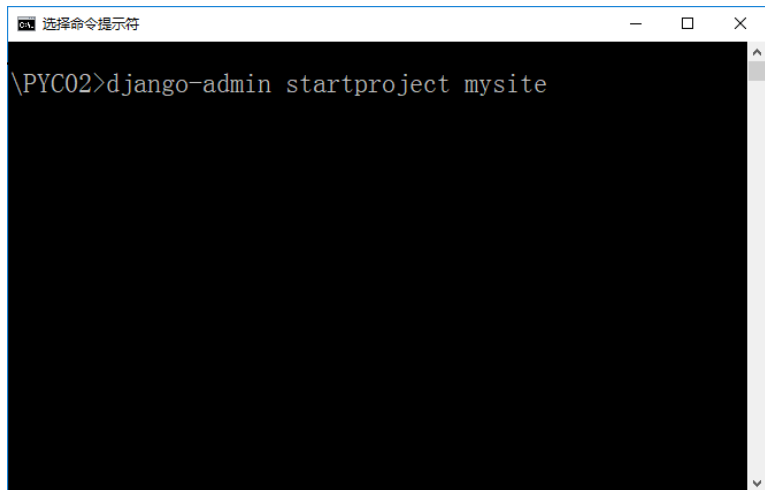


PyCharm专业版对Django及Web开发支持更好，入门开发建议使用社区版

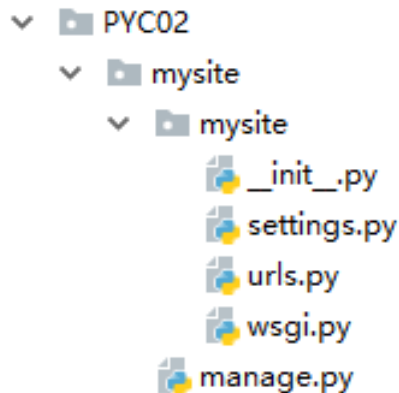
# Django框架的最小程序

**步骤1**：新建一个Web框架工程（工程：一个应用的程序员说法）

```
\>django-admin startproject mysite
```



```
选择命令提示符
[PYC02]>django-admin startproject mysite
```



# Django框架的最小程序

mysite/	————→	外层目录，名字可以更改
mysite/	————→	工程目录，保存代码和文件
__init__.py	——→	一个将mysite定义为包的空文件
settings.py	——→	部署和配置整个工程的配置文件（配置文件）
urls.py	——→	URL路由的声明文件（路由文件）
wsgi.py	——→	基于WSGI的Web服务器的配置文件
manage.py	————→	一个与Django工程进行交互的命令工具

目录结构

# Django框架的最小程序

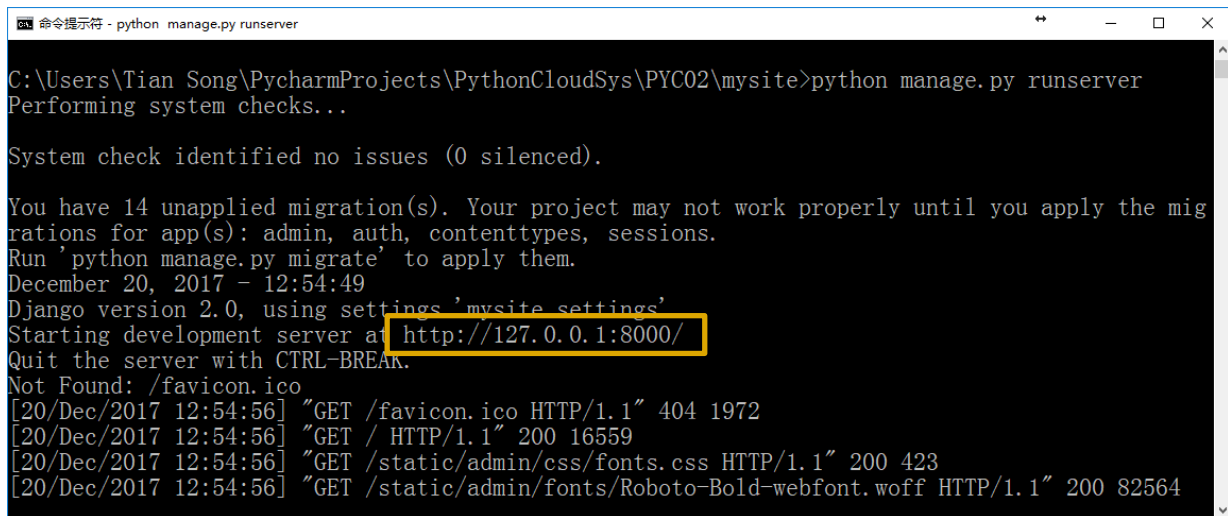
**步骤2**：修改工程，增加功能

( 此处省略5000字.... )

# Django框架的最小程序

**步骤3**：调试运行Web框架（在mysite工程目录下）

`\>python manage.py runserver`



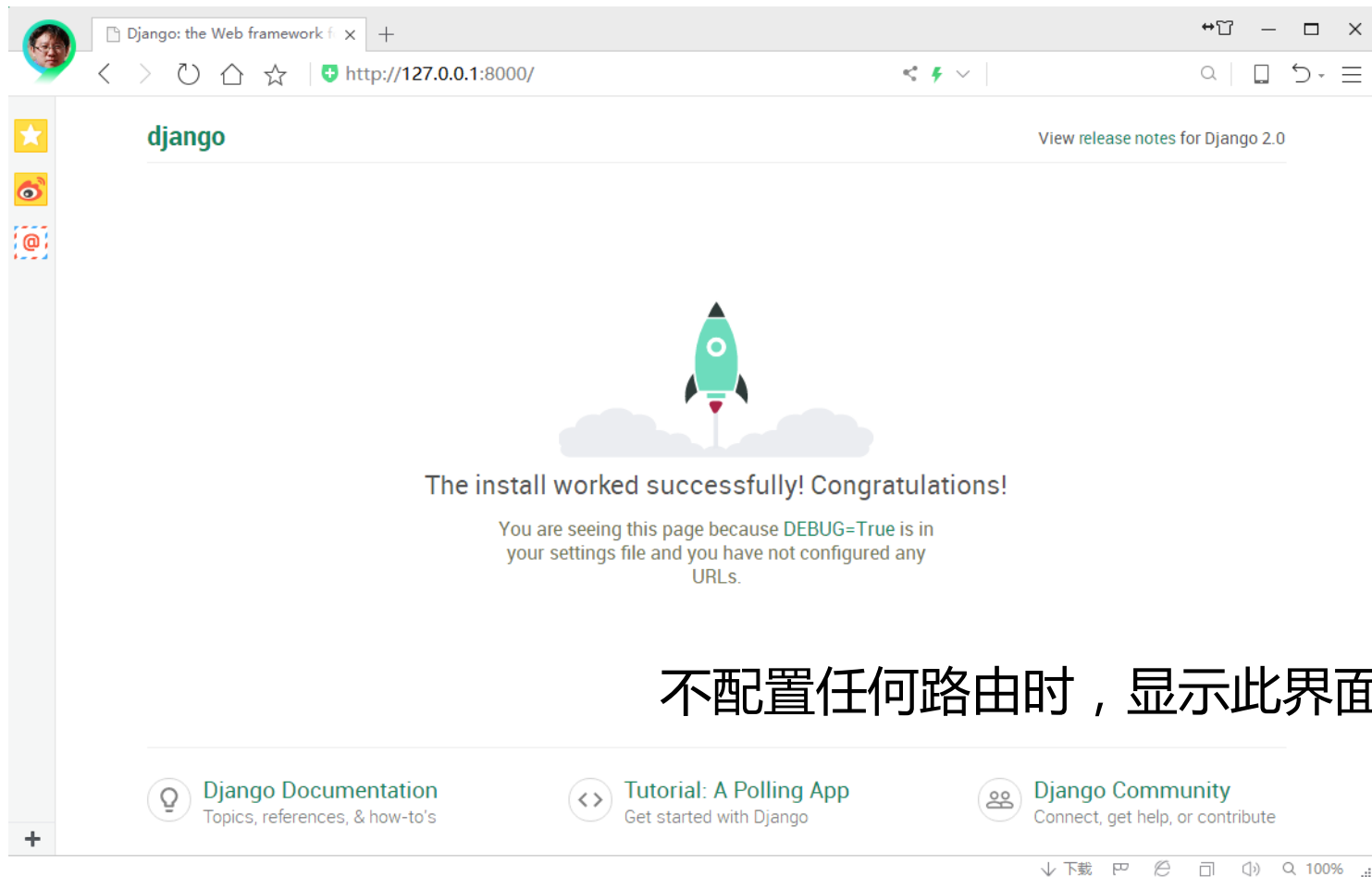
```
命令提示符 - python manage.py runserver

C:\Users\Tian Song\PycharmProjects\PythonCloudSys\PYC02\mysite>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 14 unapplied migration(s). Your project may not work properly until you apply the mig
rations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 20, 2017 - 12:54:49
Django version 2.0, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
Not Found: /favicon.ico
[20/Dec/2017 12:54:56] "GET /favicon.ico HTTP/1.1" 404 1972
[20/Dec/2017 12:54:56] "GET / HTTP/1.1" 200 16559
[20/Dec/2017 12:54:56] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[20/Dec/2017 12:54:56] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 82564
```

Django自带调试  
用Web服务器



不配置任何路由时，显示此界面



# Django框架开发流程（简）

**步骤1**：新建工程：`\>django-admin startproject mysite`

**步骤2**：修改工程：（被省略...）

**步骤3**：运行工程：`\>python manage.py runserver`

不得不说的 `django-admin`和`manage.py`

# django-admin

`\>django-admin <command> [options]`

django-admin是一个Django框架全局的管理工具：

- 建立并管理Django工程
- 建立并管理Django工程使用的数据库
- 控制调试或日志信息
- 运行并维护Django工程

更多功能：

`\>django-admin help`

# manage.py

```
\>python manage.py <command> [options]
```

与django-admin类似，但仅针对当前项目

更多功能：

```
\>python manage.py help
```

# django-admin和manage.py

几种用法：

```
\>django-admin <command> [options]
```

```
\>python manage.py <command> [options]
```

```
\>python -m django <command> [options]
```

# Django框架的最小程序

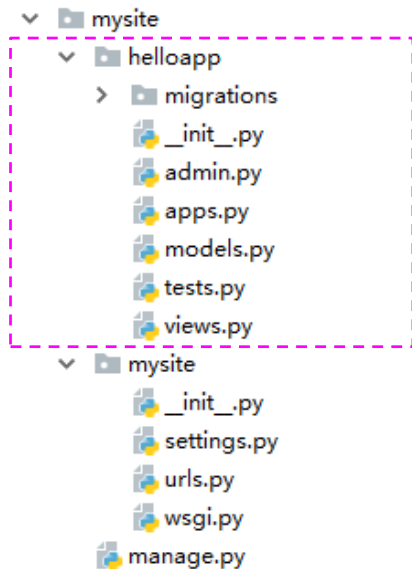
**接下来**，重点在于修改生成的工程文件

Django的Hello World程序

# Django框架的最小程序

**步骤2-1：**【修改工程】创建一个具体应用（app）

```
\>python manage.py startapp helloapp
```

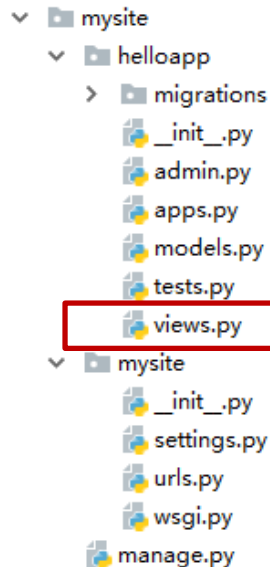


工程(project)和应用(app)什么关系呢？

- 工程对应于一个网站，是配置和应用的集合
- 应用对应于特定功能，是具体功能的载体
- 配置和功能分离是高度模块化的体现

# Django框架的最小程序

## 步骤2-2：【修改工程】修改应用的views.py

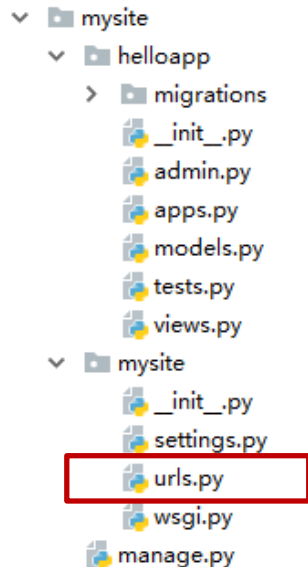


views.py中包含对某个HTTP请求(url)的响应

```
# Create your views here.  
from django.http import HttpResponse  
  
def hello(request):  
    return HttpResponse("Hello World! I am coming...")
```

# Django框架的最小程序

## 步骤2-3：【修改工程】修改URL路由



在urls.py中指定URL与处理函数之间的路径关系

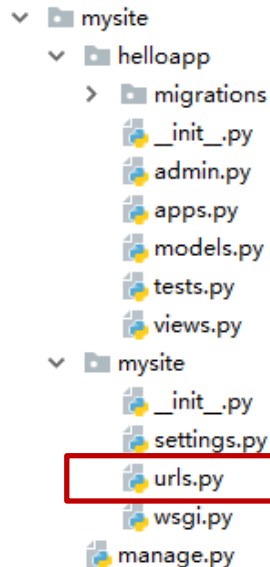
```
from django.contrib import admin
from django.urls import path
from helloapp import views

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



# Django框架的最小程序

## 步骤2-3：【修改工程】修改URL路由

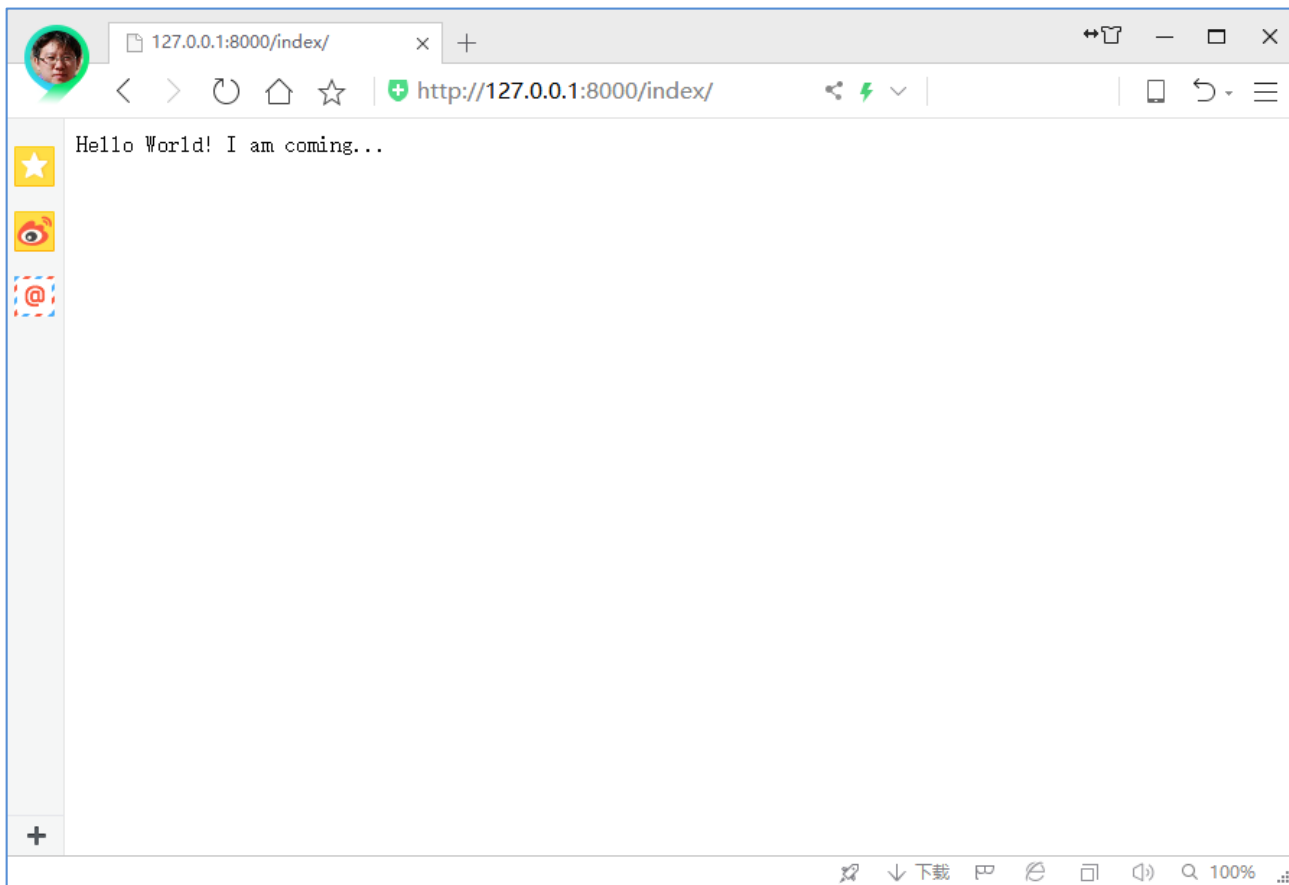


```
path('index/', views.hello)
```

某个URL

某个处理函数

路由：URL与处理函数的关联



# Django框架开发流程

**步骤1**：新建工程：`\>django-admin startproject mysite`

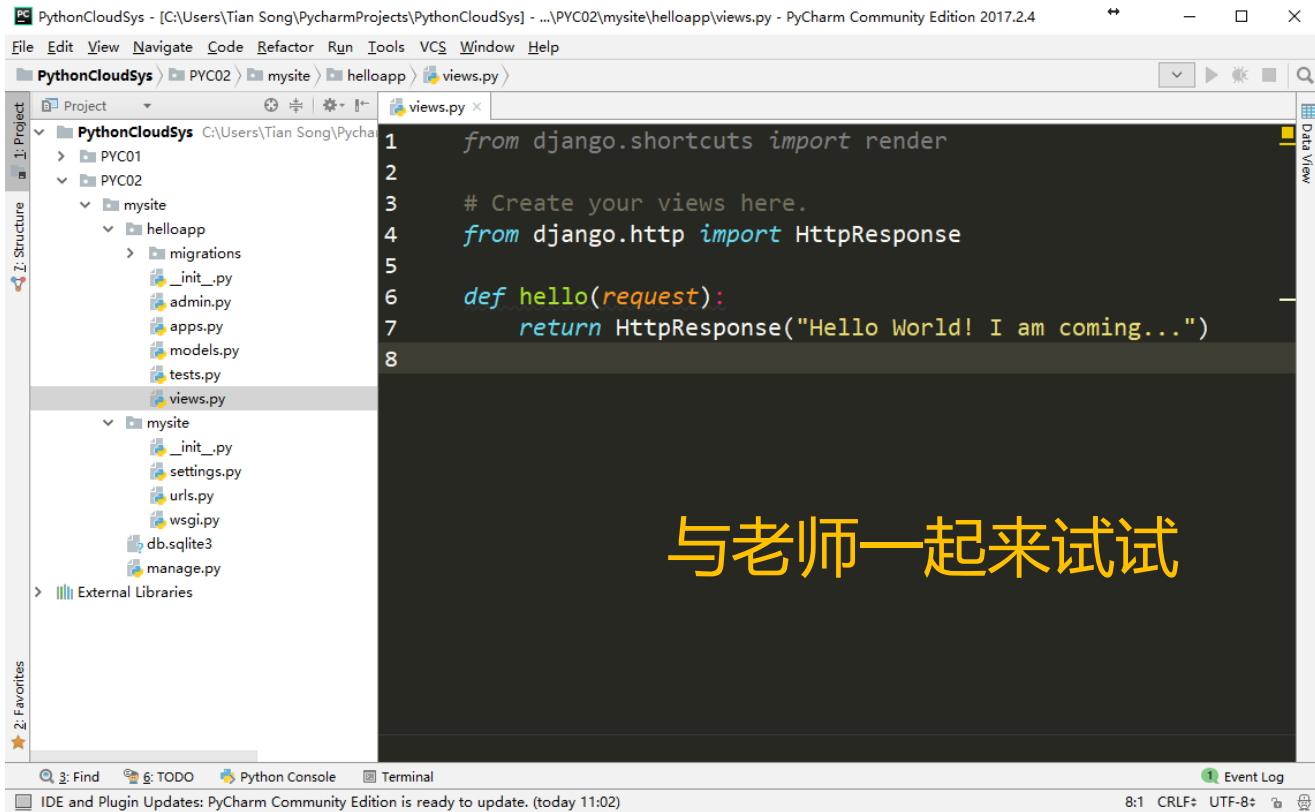
**步骤2-1**：【修改工程】创建一个具体应用（app）

**步骤2-2**：【修改工程】修改应用的views.py：对URL的具体响应功能

**步骤2-3**：【修改工程】修改URL路由：指定URL与响应之间的关系

**步骤3**：运行工程：`\>python manage.py runserver`

# 回顾：Django框架的最小程序





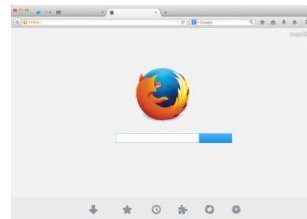
# Django的MTV开发模式

# MTV开发模式

Web云端系统的三个通用功能需求



# MTV开发模式



**M : Models 模型**

**V : Views 视图**

**T : Templates 模板**

与数据组织相关的功能

针对请求选取数据的功能

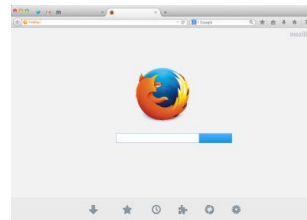
与表现相关的所有功能

组织和存储数据的方法和模式，与  
数据模型相关的操作

选择哪些数据用于展示，指定显示模  
板，每个URL对应一个回调函数

页面展示风格和方式，与具体数据  
分离，用于定义表现风格

# MTV开发模式



M : Models 模型

V : Views 视图

T : Templates 模板

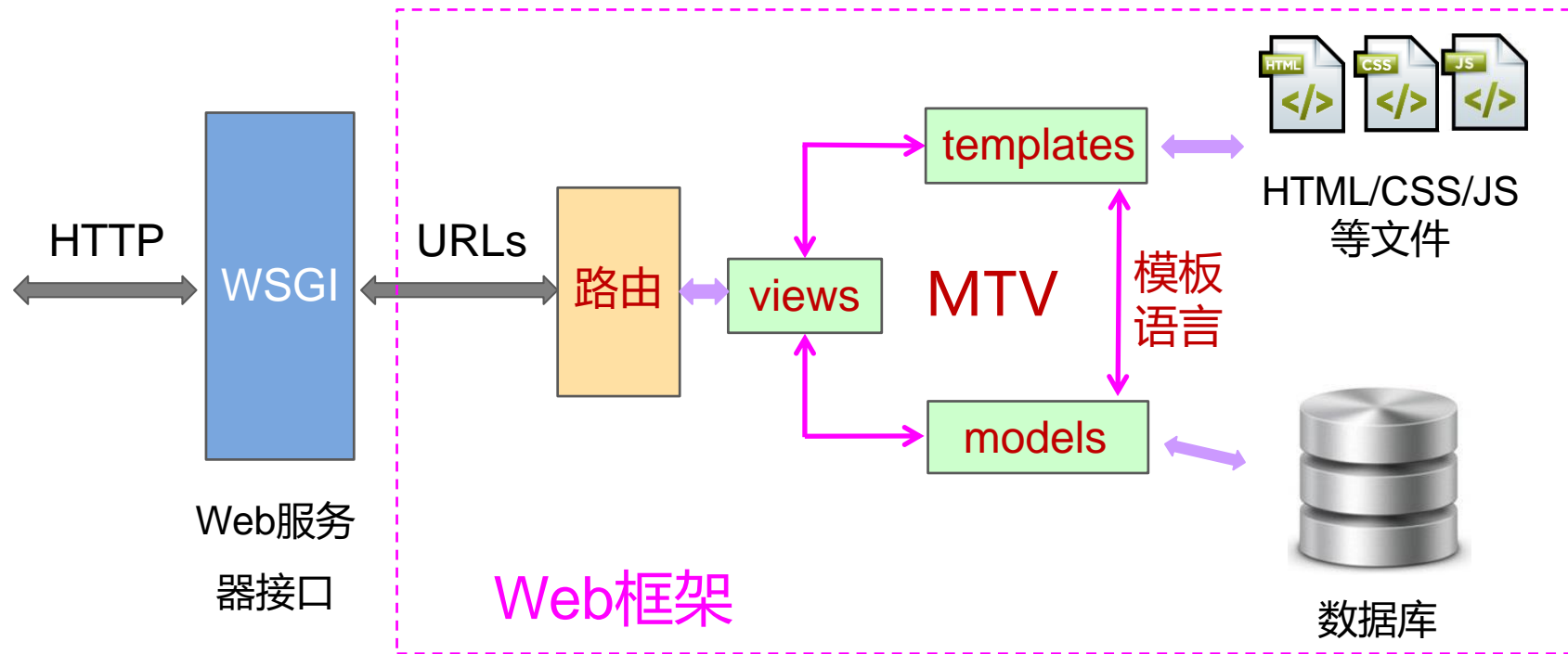
数据

处理

样式

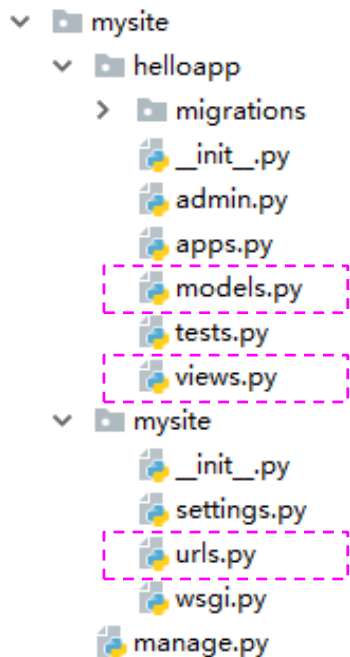


# MTV开发模式



# MTV开发模式

M : Models 模型 ↔ V : Views 视图 ↔ T : Templates 模板



模板 ( T ) 并不默认生成 , 需要手工创建目录

针对某个app的模型 ( M )

针对某个app的视图 ( V )

路由 : Web框架的一部分

# Django最小程序的改进

需求：返回一个HTML页面，而不是一个字符串

思路：建立模板（T），对应特定请求，返回模板页面

新建hello2app，通过index2来访问

# Django最小程序的改进

**步骤2-1**：新建hello2app应用

```
\>python manage.py startapp hello2app
```

**步骤2-2**：使用PYC01-HTMLJSDemo.html为返回页面，修改views.py

```
from django.shortcuts import render hello2app/views.py  
  
def hello(request):  
    return render(request, "PYC01-HTMLJSDemo.html")
```

render()是一个打包函数，第一个参数是request，第二个参数是页面

# Django最小程序的改进

**步骤2-3**：在hello2app应用中，新增urls.py文件（本地路由文件）

· 代表当前app

```
from django.urls import path
from . import views
```

*hello2app/urls.py*

```
urlpatterns = [
    path('', views.hello)
]
```

变量名固定

# Django最小程序的改进

**步骤2-4**：在全局路由文件中增加对本应用路由文件的引用

include()函数，用于引入其他路由文件

```
from django.contrib import admin
from django.urls import include, path
from helloapp import views
```

*mysite/urls.py*

```
urlpatterns = [
    path('index2/', include('hello2app.urls')),
    path('index/', views.hello),
    path('admin/', admin.site.urls),
]
```

将hello2app的局部路由增加到全局路由中

# Django最小程序的改进

**步骤2-5**：设置模板路径，让Django框架找到模板所在目录

```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': [os.path.join(BASE_DIR, 'hello2app/templates')],  
        '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',  
            ],  
        },  
    ],  
],
```

 指定templates所在路径

*mysite/settings.py*

# Django最小程序的改进

**步骤2-1**：新建hello2app应用

**步骤2-2**：使用PYC01-HTMLJSDemo.html为返回页面，修改views.py

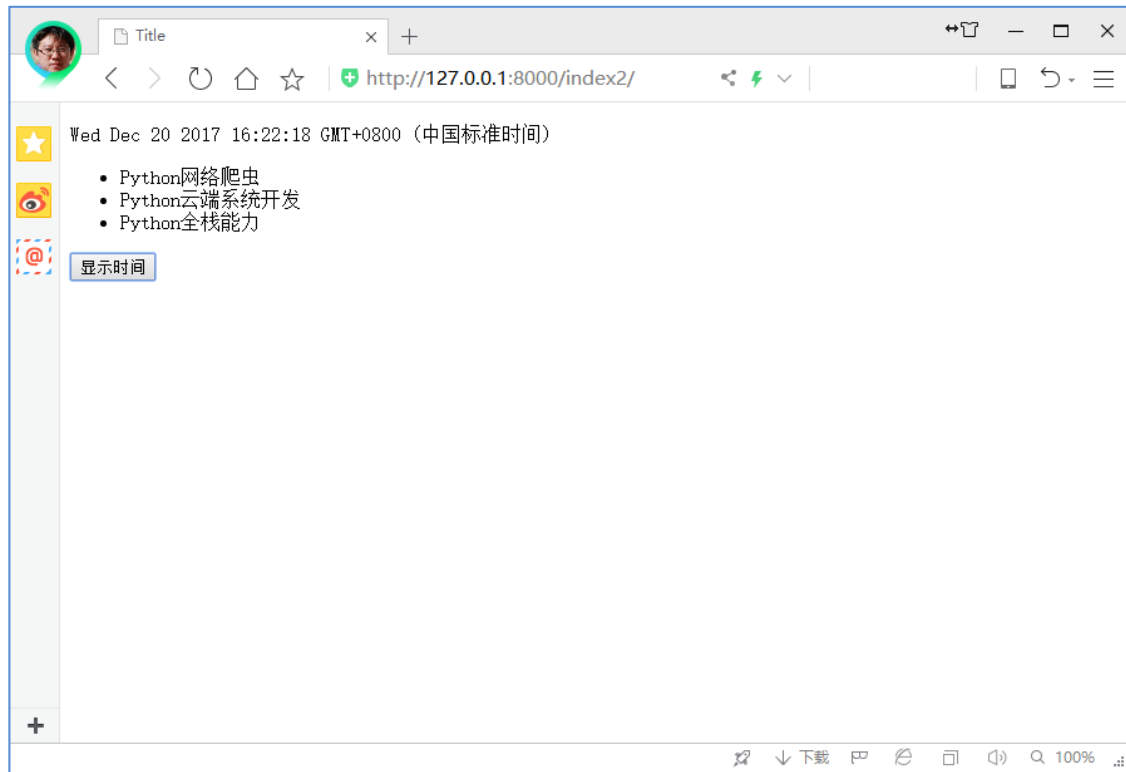
**步骤2-3**：在hello2app应用中，新增urls.py文件（本地路由文件）

**步骤2-4**：在全局路由文件中增加对本应用路由文件的引用

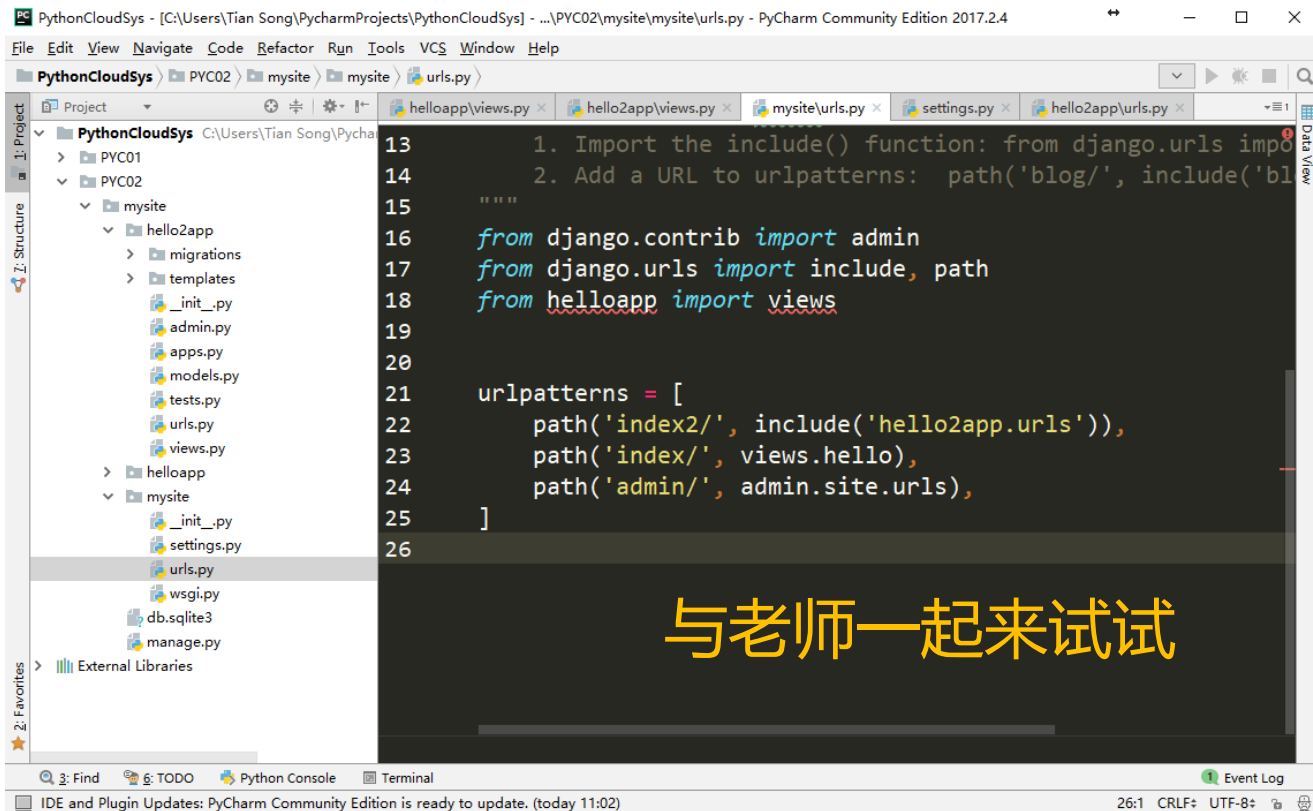
**步骤2-5**：设置模板路径，让Django框架找到模板所在目录



# Django最小程序的改进



# 回顾：Django最小程序的改进





# 实例1：云端留言版之基本框架

# 实例1：云端留言版(1)

## 基本功能定义：

- 提交留言功能：
  - † 用户设定自己的名字为A，指定任意名字B
  - † 向B留言，记为msg，留言保存在云端
- 获取留言功能：
  - † 输入名字A，云端返回10条最新留言记录

# 实例1：云端留言版(1)

## 开发要求：

- 弱化Web设计，有简单Web界面即可
- 重视云端设计，掌握Django库的使用
- 数据用文件方式存储

# 开发流程

**步骤1**：新建工程 `cloudms`

**步骤2-1**：新建应用 `msgapp`

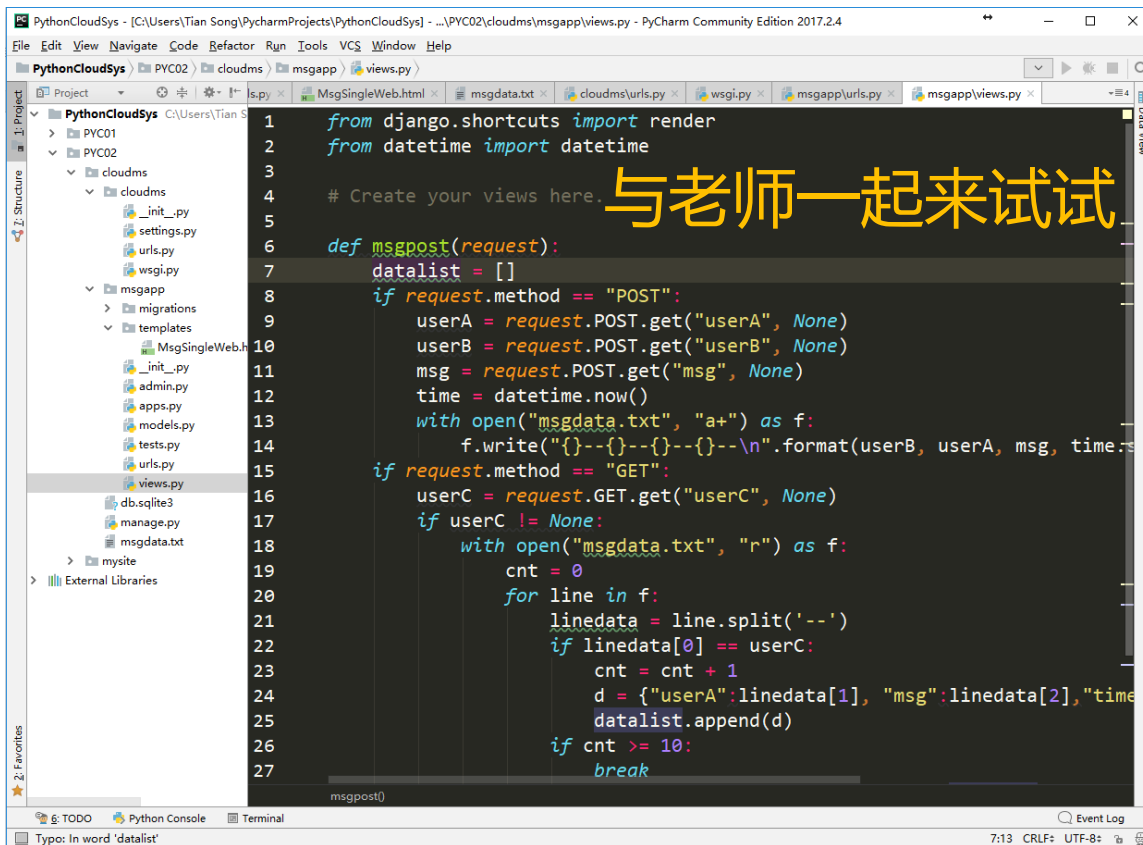
**步骤2-2**：增加模板，即显示界面的HTML/CSS/JS代码，配置路径

**步骤2-3**：设定URL路由，本地路由和全局路由

**步骤2-4**：编写交互代码

**步骤3**：运行工程

# 实例1：云端留言版(1)



# 开发流程

**步骤1**：新建工程 `cloudms`

```
\>django-admin startproject cloudms
```



# 开发流程

**步骤2-1**：新建应用 `msgapp`

```
\>python manage.py startapp msgapp
```

# 开发流程

## 步骤2-2：增加模板，配置路径

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>云端留言板(1)首页</title>
</head>
<body>
  <h1>提交留言功能区</h1>
  <form action="/msggate/" method="post">
    {% csrf_token %}
    发送方 <input type="text" name="userA" /> <br>
    接收方 <input type="text" name="userB" /> <br>
    消息文 <input type="text" name="msg" /> <br>
    <input type="submit" value="留言提交" />
  </form>
```

```
<h1>获取留言功能区</h1>
  <form action="/msggate/" method="get">
    接收方 <input type="text" name="userC" /> <br>
    <input type="submit" value="留言获取" />
  </form>
  <table border="1">
    <thead>
      <th>留言时间</th>
      <th>留言来源</th>
      <th>留言信息</th>
    </thead>
    <br>
    <tbody>
      {% for line in data %}
      <tr>
        <td>{{ line.time }}</td>
        <td align="center">{{ line.userA }}</td>
        <td>{{ line.msg }}</td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
</body>
</html> msgapp/templates/MsgSingleWeb.html
```

# 开发流程

## 步骤2-2：增加模板，配置路径

```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': [os.path.join(BASE_DIR, "msgapp/templates")],  
        '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',  
            ],  
        },  
    ],  
]
```

*cloudms/settings.py*

# 开发流程

## 步骤2-3：设定URL路由，本地路由和全局路由

```
from django.urls import path
from . import views
```

```
urlpatterns = [
    path('', views.msgproc),
]
```

*msgapp/urls.py*

```
from django.contrib import admin
from django.urls import include, path
```

```
urlpatterns = [
    path('msggate/', include('msgapp.urls')),
    path('admin/', admin.site.urls),
]
```

*cloudms/urls.py*

## 步骤2-4：编写交互代码

```
from django.shortcuts import render
from datetime import datetime
```

*msgapp/views.py*

```
def msgproc(request):
    datalist = []
    if request.method == "POST":
        userA = request.POST.get("userA", None)
        userB = request.POST.get("userB", None)
        msg = request.POST.get("msg", None)
        time = datetime.now()
        with open("msgdata.txt", "a+") as f:
            f.write("{}--{}--{}--{}\n".format(userB, userA, msg, time.strftime("%Y-%m-%d %H:%M:%S")))
    if request.method == "GET":
        userC = request.GET.get("userC", None)
        if userC != None:
            with open("msgdata.txt", "r") as f:
                cnt = 0
                for line in f:
                    linedata = line.split('--')
                    if linedata[0] == userC:
                        cnt = cnt + 1
                        d = {"userA":linedata[1], "msg":linedata[2], "time":linedata[3]}
                        datalist.append(d)
                    if cnt >= 10:
                        break
    return render(request, "MsgSingleWeb.html", {"data":datalist})
```

# 实例1：云端留言版(1)

云端留言板(1)首页

127.0.0.1:8000/msggate/

提交留言功能区

发送方 郭靖

接收方 黄蓉

消息文 相约桃花岛

留言提交

获取留言功能区

接收方

留言获取

留言时间 留言来源 留言信息

提交留言

云端留言板(1)首页

127.0.0.1:8000/msggate/?userC

提交留言功能区

发送方

接收方

消息文

留言提交

获取留言功能区

接收方 黄蓉

留言获取

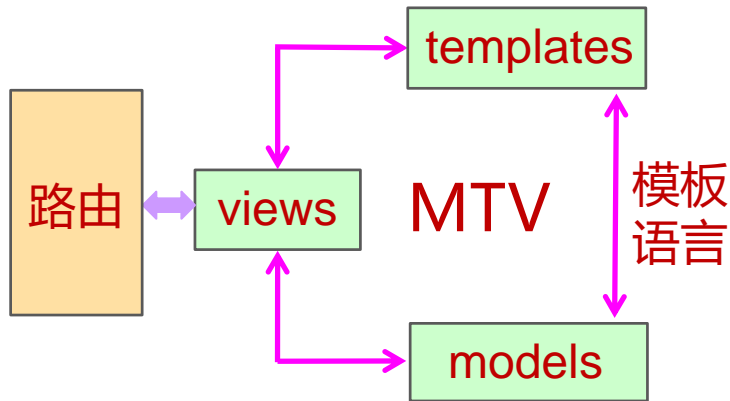
留言时间	留言来源	留言信息
2017-12-21 08:18:17	郭靖	相约桃花岛
2017-12-21 08:19:02	梅超风	小师妹
2017-12-21 08:20:08	黄药师	回家吃饭

获取留言



# 单元小结

# Django框架基础



Django简介与Web框架

Django框架的最小程序

Django的MTV开发模式

实例1：云端留言版之基本框架