# **一、**Django简介

## **1.Django的优点**

功能完善，容易上手，开发速度快，安全性高；

有完善的在线文档；

其模型自带数据库ORM(Object-Relational-Mapping，对象关系映射)组件，使得开发者无须学习其他数据库访问技术(如SQLAlchemy等)；

可以使用正则表达式管理路由映射，方便灵活。

## **MVC和MTV模式**

**MVC模式：**

目前的主流web框架，基本上都使用mvc模式开发web应用，其最大的优点是可以降低系统各模块间的耦合度。

Modle：模型，负责处理各个功能的实现，如增加，修改和删除，其中包含模型实体类和业务处理类；

View：视图，负责页面的显示和用户的交互，包含由HTML、CSS、JavaScript组成的各种页面；

Controller：控制器，用于将用户请求转发给相应的模型进行处理，并根据模型的处理结果向用户提供相应的相应。

**MTV模式：**

对传统的MVC模式进行了修改，修改后的模式称为MTV模式。

Model：模型，负责业务对象和数据库的关系映射；

Template：模版，负责页面的显示和用户的交互；

View：视图，负责业务逻辑，并在适当的时候调用Model和Template。

# **二、Django安装**

## 1.windows系统：

安装命令：

**pip install django==版本 -i <https://mirrors.aliyun.com/pypi/simple/>**

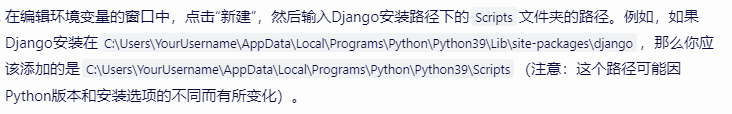
例子：

**pip install django==4.1 -i <https://mirrors.aliyun.com/pypi/simple/>**

查找Django的安装位置：

**pip show django**

配置环境变量：



查看Django版本：

**python -m django --version**

## **Linux系统：**

略

# **三、Django知识点**

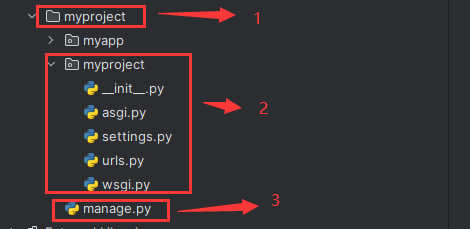
## 1.新建项目

命令：

django-admin startproject 项目名称

例子：

django-admin startproject myproject



**注意：**

manage.py：一个命令行实用程序，可以通过命令行方式与Django项目进行交互。

myproject：外面的为根目录，与Django无关，可以任意命名；里面的为项目目录，这个目录请勿修改。

asgi.py：与ASGI兼容的web服务器，为项目提供服务的入口。

settings：该项目的全局配置文件。

urls：该项目的路由配置文件。

wsgi.py：与WSGI兼容的web服务器，为项目提供服务的入口。

db.sqlit3：sqlit格式的数据库。新版本未显示。

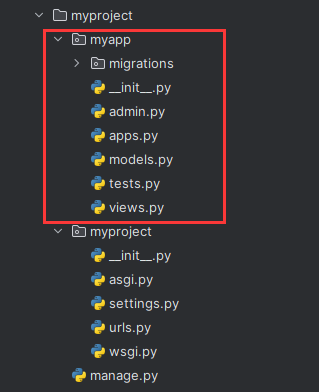
## **2.创建应用**

命令：

python manage.py startapp 应用名

例子：

python manage.py startapp myapp



**注意：**

myapp：应用目录，请勿修改这个目录名称。

admin.py：后台管理使用。

apps.py：应用管理文件。

models.py：模型文件。

tests.py：测试用例。

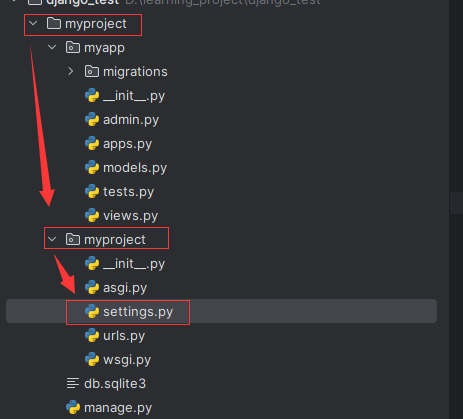
views.py：视图文件。

migrations：该目录包含数据的迁移文件。

创建应用时，需要切换至manage.py同级目录，再输入需要创建应用的命令。

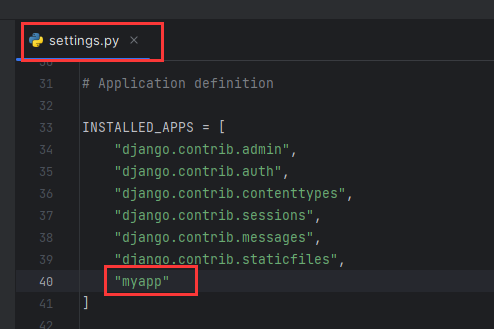
## **3.配置文件**

所有的配置信息都在setting.py文件中完成，该文件的位置如下：



### 3.1配置应用

在settings.py文件中找到INSTALLED\_APPS，将应用名称添加即可。



### **3.2配置数据库**

### **3.3配置静态文件**

### **3.4配置**

### **3.5配置**

### **3.6配置**

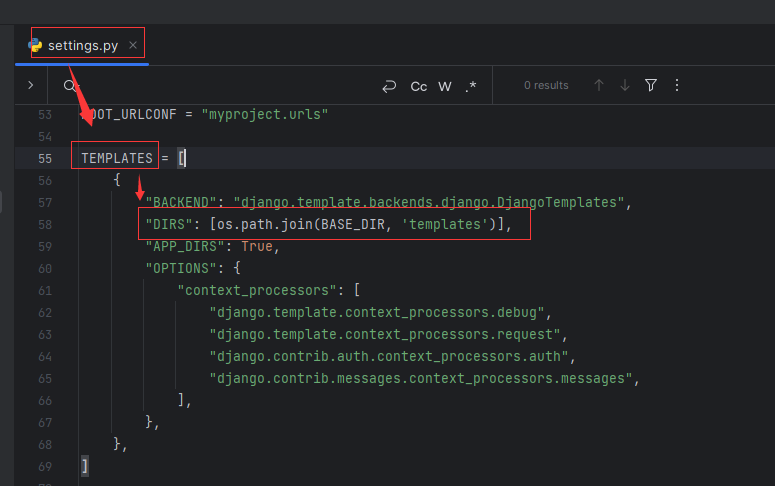
### **3.7配置**

## **4.创建模板**

使用命令行创建Django应用时，默认是没有模板目录，且目录名称为templates，可以更改，但建议保持不变。



模版创建完成后，还不能直接使用，需要在settings配置文件中进行注册。



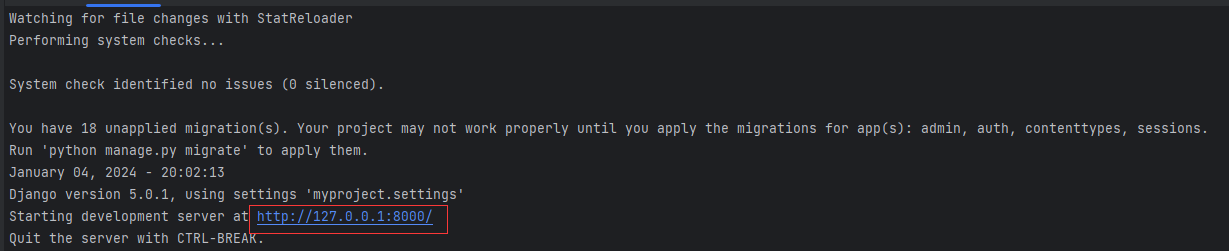
## **5.运行项目**

命令：

python manage.py runserver [port]

例子：

python manage.py runserver

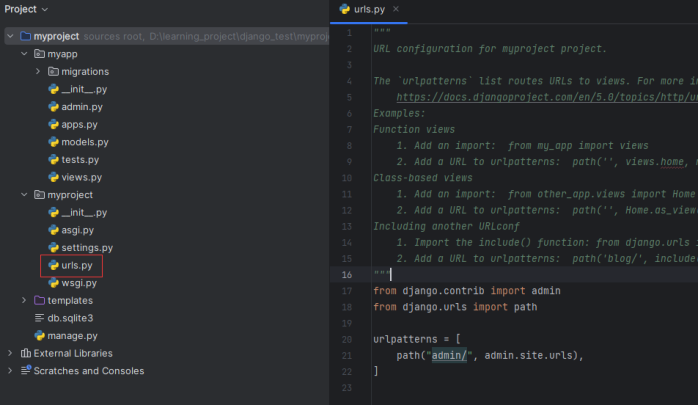


显示上述信息，并在浏览器中输入“http://localhost:8000/”，如果页面有详细信息，表示Django应用运行成功。

## **Django的路由**

### 6.1 简介

当我们创建一个Django项目时，在项目目录下会生成一个urls.py文件，在该文件下定义了Django项目的主要路由信息，该文件是整个项目路由解析的入口。



在新建的应用中也可以使用独立的urls.py，但是需要手动创建；随着业务模块越来越多，路由规则也越来越复杂。可以用“路由包含”来简化项目的复杂度，为每一个应用创建一个urls.py文件，把相关的路由配置都放在每个应用的urls.py文件中；当用户发送请求时，会从根路由开始寻找每个应用的路由信息，生成一个完整的路由列表。当Django从当前请求中获取路由地址后，会先在这个路由列表中进行匹配，然后执行路由相关的视图函数，从而完成整个请求过程。

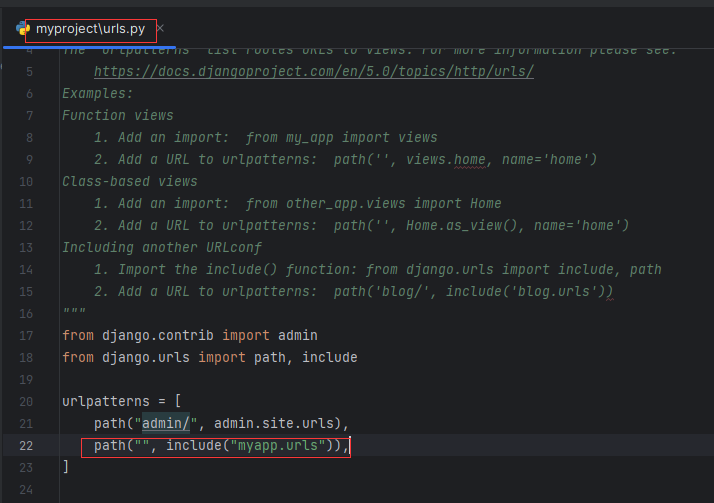
### 6.2 url配置

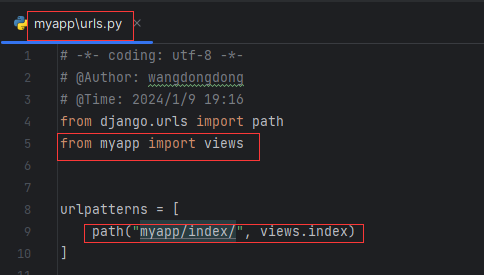
**路由匹配规则：**

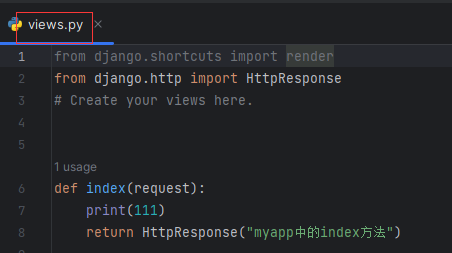
在项目的urls.py文件中，urlpatterns列表会从上往下进行匹配，如果匹配成功，则调用path()函数中第二个参数所指定的视图函数，且不再继续匹配，否则返回404错误。

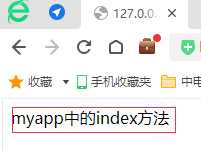
**注意：**

如果在应用中定义了子路由，则在根路由中使用include(“应用名.urls”)来加载子路由，路由信息一般以“/”结尾。









### 6.3 路由参数解析

路由规则：

<参数数据类型：参数名称>

例子：

urlpatterns=[

path(“show/<int:id>/”,views.show)

]

参数类型：

str：字符串，不包含“/”，默认类型；

int：匹配0和正整数；

slug：匹配任何ASCII字符，连接符和下划线；

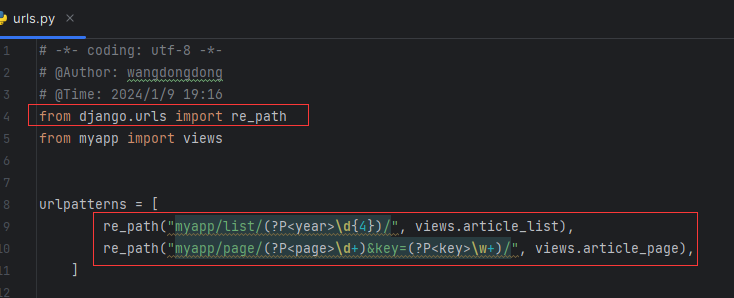
uuid：匹配一个UUID格式的字符串，该对象包含“-”，所有字母必须小写。

### 6.4 re\_path()方法正则匹配复杂路由

相对于path()，功能更强大。格式如下：

(?P<name>pattern)

例子：



“myapp/list/(?P<year>\d{4})/”：

接收以“myapp/list/”开头，后面跟4位整数的路由；

“myapp/page/(?P<page>\d+)&key=(?P<key>\w+)/”：

接收以“myapp/page/”开头，后面跟任意位数整数，且第2个参数可以是字母，数字和下划线的路由；

### 6.5 反向解析路由

在Django的路由配置中，可以给一个路由配置项命名，然后通过别名反向解析得到路由。

例子：

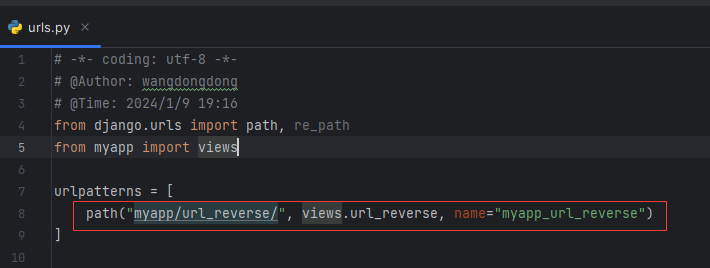
urlpatterns=[

path(“myapp/url\_rever/”,views.url\_rever,name=”myapp\_url\_rever)

]

注意：

name：在name后可以跟任意字符串，为了避免冲突，建议以“应用名\_配置项名”,根据name得到路由配置项中的URL地址，称作“反向解析路由”，这样的好处是，只要name不变，URL地址可以任意改变。







## **7.Django的视图**

### 7.1 简介

Django中的视图是MTV模式中的View层，用于处理客户端的请求并生成响应数据。在视图中使用函数处理请求的方式，被称为视图函数-FBV（Function Base Views），视图函数的代码一般放在应用目录下的views.py文件中。

### 7.2 视图函数的底层原理

视图函数主要使用HttpRequest请求对象和HttpResponse响应对象，当浏览器向服务端请求一个页面时，Django先创建一个HttpRequest请求对象（该对象包含请求元数据），然后加载相应的视图，将这个HttpRequest请求对象作为第一个参数传递给视图，视图函数会返回一个HttpResponse响应对象。

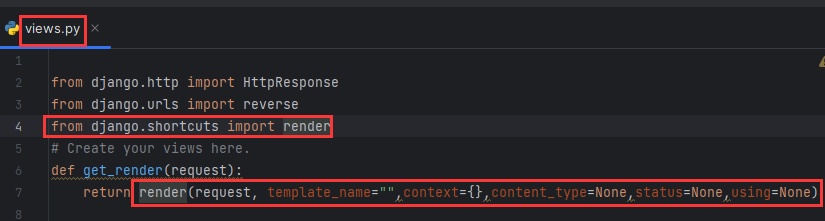
### 7.3 视图处理函数的使用

通过HttpRequest对象和HttpResponse对象，可以处理基本的数据请求并返回相应数据；但是这种方式繁琐，以下是几种简单方法：

1.用render()函数实现页面渲染

该函数根据模板文件和传递给模板文件的字典类型的变量，生成一个HttpResponse对象并返回。

格式：



参数含义：

request：传递给视图函数的所有请求；

template\_name：渲染的模板文件，一般放在templates目录下；

context：数据格式为字典类型，传递到HTML文件；

content\_type：用于生成文档的MIME，默认为text/html；

status：表示响应的状态码，默认为200；

using：设置模板引擎，用于解析模板文件。

2.用redirect()函数实现页面重定向

在项目开发中，经常遇到页面重定向的情况，如网站目录结构调整，网页被移到一个新地址这类情况，如果不做重定向，则通过用户收藏夹中的链接或搜索引擎数据库中的旧地址，会报404页面错误信息。

在Django中，使用重定向函数redirect()实现页面重定向，该函数的参数有3种情况：

(1)通过调用模型的get\_absolute\_url()函数进行重定向；

(2)通过路由反向解析进行重定向；

(3)通过一个绝对的或相对的URL，让浏览器跳转到指定的URL进行重定向。

### 7.4 视图类

在视图里使用类处理方式，被称为视图类-CBV(class base views)，视图类可以更好地处理不同的HTTP请求，在处理视图逻辑时，不用通过“if...else...”来区分请求是GET，还是POST，而是通过在视图类中定义的get()方法和post()方法进行区别。

视图类在调用时，只能是函数的方式，而不能是类的方式，因此，需要将视图类用as\_view()转化为视图函数。

## **8.Django的模版**

### 8.1 简介

Django模板，主要用于页面的展现，实现了业务逻辑和内容显示的分离。Django模板技术可分为两部分：

静态部分：如HMTL，CSS和JavaScript；

动态部分：如Django的模板语音DTL(Django Template Language)。

Django模板是一种带有DTL语言的HTML文件，这个HTML文件可以被Django编译，其中可以传递参数，以实现数据动态化，最终发送给客户端浏览器。Django模板语言包括模板变量，模板标签和模板过滤器。

### 8.2 模板变量

模板变量可以是字符串，列表，字典和类对象。

格式：

{{ 变量名 }}

例子：

我的姓名{{ name }}，我的年龄{{ age }}

注意：

在模板变量的名称前后都有空格，且模板变量的名称可以由字母，数字和下划线组成，但不能有空格等其他字符。

### 8.3 模板标签

模板标签需要用标签限定符{% %}进行包裹，常见的模板标签有{% if %}和{% endif %}，{% load %}，{% block %}和{% endblock %}等。有些标签属于闭合标签，例如，{% if %}的闭合标签是{% endif %}，模板标签的作用：载入代码渲染模板，或对传递过来的参数进行逻辑判断和计算。

举例：

{% if %}

{{ 内容1 }}

{% elif %}

{{ 内容2 }}

{% else %}

{{ 内容3 }}

{% endif %}

### 8.4 模板过滤器

模板过滤器用于对模板变量进行操作，常见的过滤器有字符串大小写转换过滤器，日期过滤器等。

格式：

{{ 变量名 | 过滤器：参数 }}

例子：

{{ value | date：Y年m月j日H时i分s秒 }}

### 8.5 自定义过滤器和标签

## **9.Django的模型**

## **10.用户认证**

## **11.Django的后台管理**

## **12.接口的设计与实现**

## **13.分层的自动化测试**

# **四、Django的开发流程**

# **五、Django的注意事项**

# **六、Django的常见问题**

# **七、学习网址**