# **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/>**

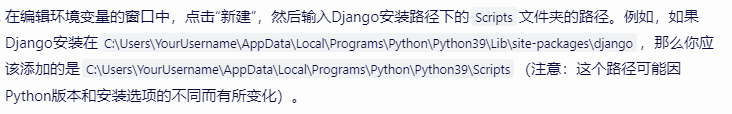
例子：

**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**

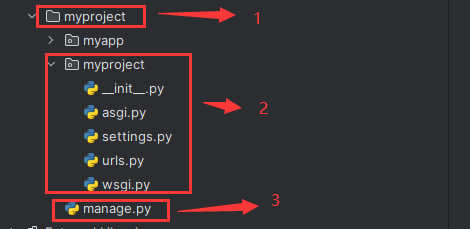
## 1.新建项目

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。

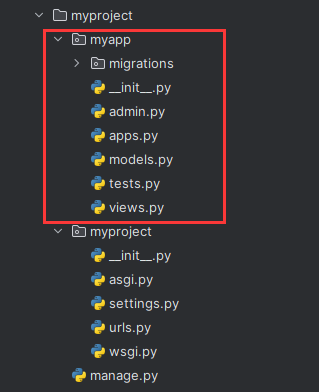
## **2.创建应用**

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。

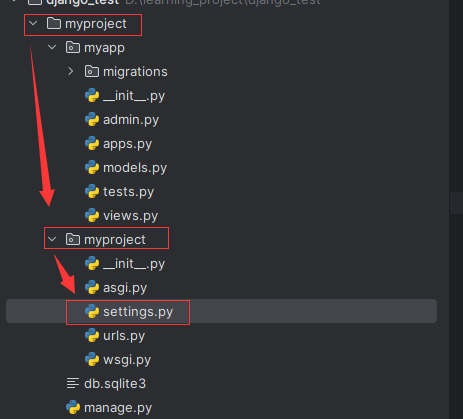
views.py：视图文件。

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。

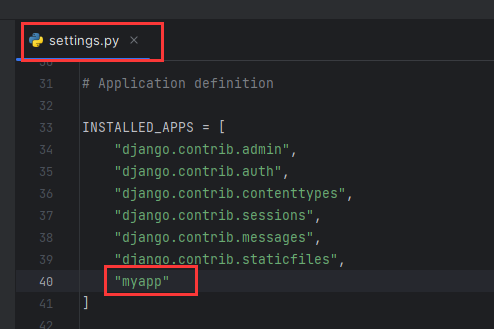
## **3.配置文件**

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**

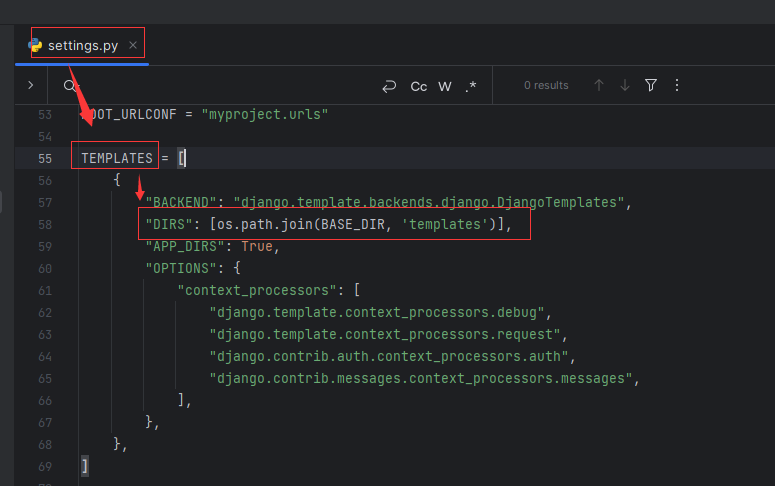
### **3.7配置**

## **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。



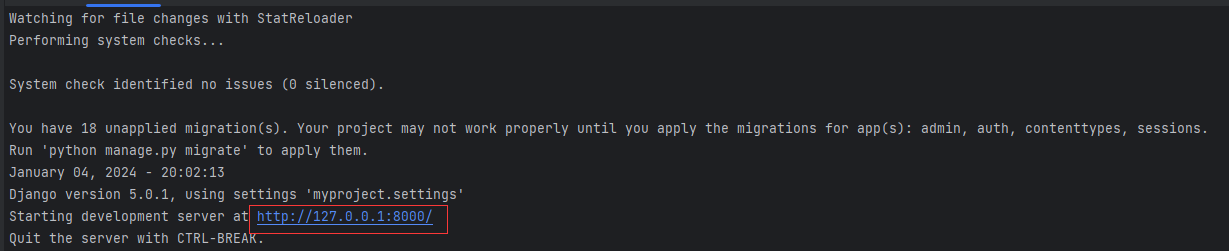
## **5.运行项目**

命令：

python manage.py runserver [port]

例子：

python manage.py runserver

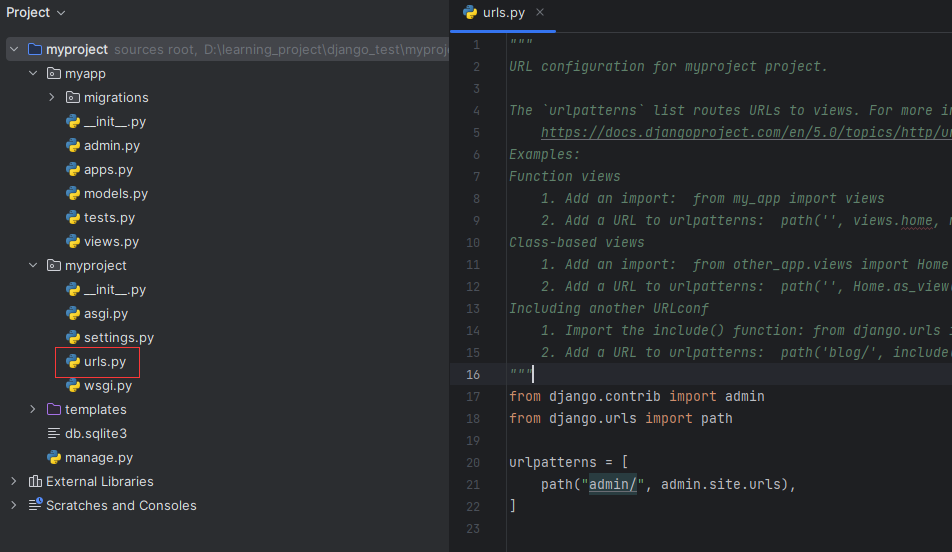


显示上述信息，并在浏览器中输入“http://localhost:8000/”，如果页面有详细信息，表示Django应用运行成功。

## **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。

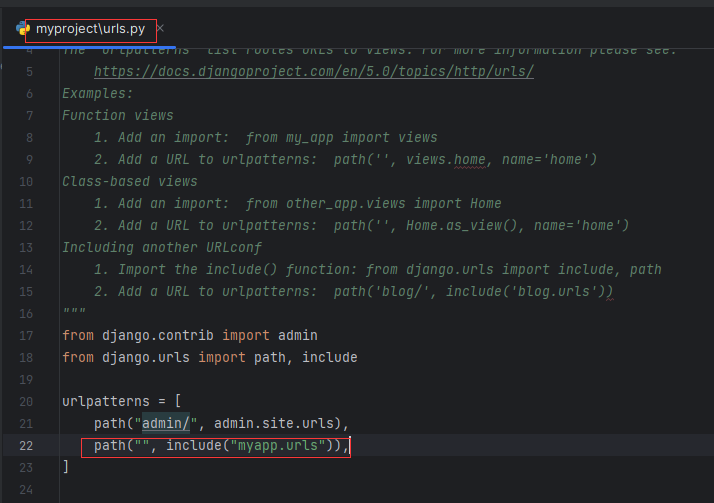
### 6.2 url配置

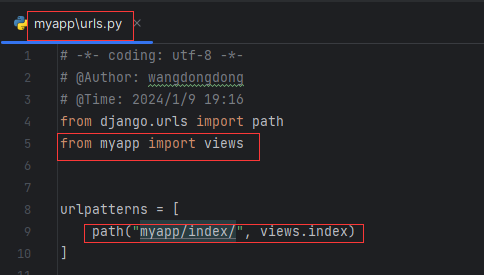
**路由匹配规则：**

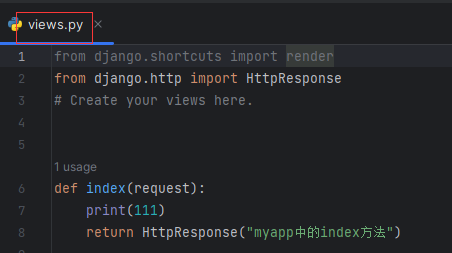
在项目的urls.py文件中，urlpatterns列表会从上往下进行匹配，如果匹配成功，则调用path()函数中第二个参数所指定的视图函数，且不再继续匹配，否则返回404错误。

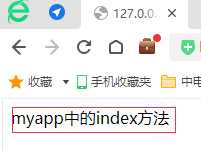
**注意：**

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；

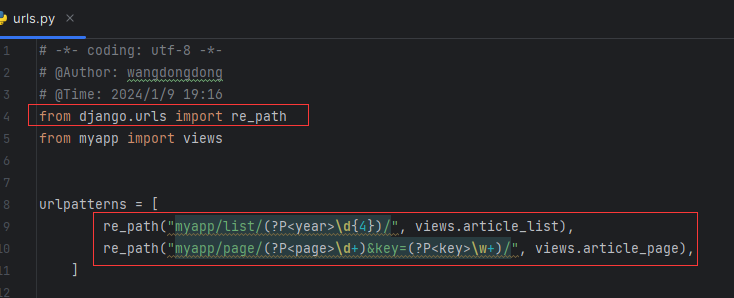
uuid：匹配一个UUID格式的字符串，该对象包含“-”，所有字母必须小写。

### 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：

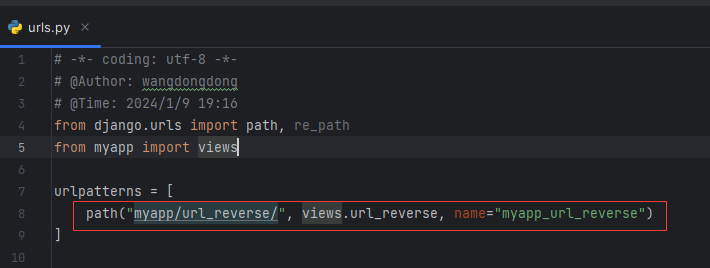
urlpatterns=[

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**

## **8.Django的模版**

## **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**