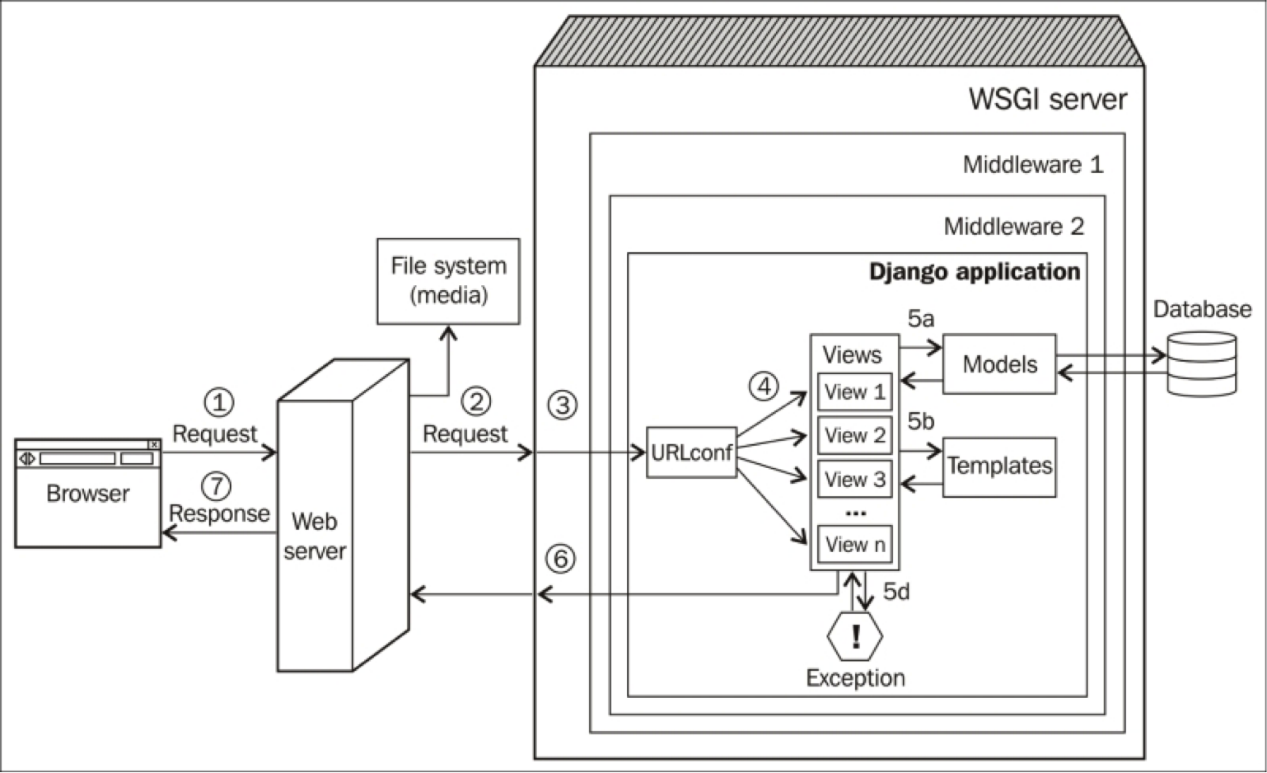
# Django基础



1. 浏览器发送请求（基本上是字节类型的字符串）到web服务器。

2. web服务器（比如，Nginx）把这个请求转交到一个WSGI（比如，uWSGI），或者直接地文件系统能够取出一个文件（比如，一个CSS文件）。

3. 不像web服务器那样，WSGI服务器可以直接运行Python应用。请求生成一个被称为environ的Ptyhon字典，而且，可以选择传递过去几个中间件的层，最终，达到Django应用。

4. URLconf中含有属于应用的urls.py选择一个视图处理基于请求的URL的那个请求，这个请求就已经变成了HttpRequest——一个Python字典对象。

5. 被选择的那个视图通常要做下面所列出的一件或者更多件事情：

* + 通过模型与数据库对话。
  + 使用模板渲染HTML或者任何格式化过的响应。
  + 返回一个纯文本响应（不被显示的）。
  + 抛出一个异常。

6. HttpResponse对象离开Django后，被渲染为一个字符串。

7. 在浏览器见到一个美化的，渲染后的web页面。

## 组成及结构

MVT

Django是一款python的web开发框架，与MVC有所不同，属于MVT框架

* m表示model，负责与数据库交互
* v表示view，是核心，负责接收请求、获取数据、返回结果
* t表示template，负责呈现内容到浏览器

组成结构

1. 管理工具（management）:内置的创建站点、迁移数据、维护静态文件的命令工具
2. 模型（model）：提供数据访问接口和模块，包括数据字段，元数据、数据关系等定义和操作
3. 视图（view）：视图层封装了HTTP request和response的一系列操作和数据流，主要功能包括URL的映射机制、绑定模板等
4. 表单（template）：通过内置的数据类型和控件生成HTML表单
5. 管理站（admin）：通过声明需要管理的model，快速生成后台数据管理网站

## 虚拟环境搭建

### virtualenv

1. 创建：virtualenv [虚拟环境名称]
2. 删除：rmvirtualenv [虚拟环境名称]
3. 进入：activate (scripts路径加入系统变量)
4. 退出：deactivate
5. x虚拟环境希望使用到y虚拟环境中所安装的所有的包

步骤：

1. 将安装在x虚拟环境中的包导入到一个文件中去,如果在home目录中执行的操作，就到home中找这个文件

pip freeze > output.txt

1. 在y虚拟环境中找到这个文件，并进行安装

pip install -r output.txt

1. 等待安装，并检查y虚拟环境中是否已成功安装工具包

pip freeze

### pipenv

1. pipenv –three
2. pipenv shell

## 安装

pip3 install django

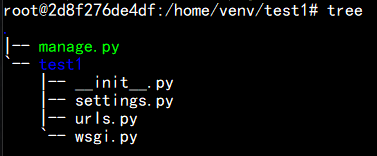
查看版本

import django

django.get\_version()

## 创建项目

django-admin startproject 项目名



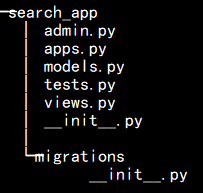
**目录说明：**

* test1: 项目的容器。
* [manage.py](https://docs.djangoproject.com/zh-hans/2.1/ref/django-admin/): 命令行工具，以各种方式与Django 项目交互。不需要编辑
* 里面一层的 mysite/ 目录包含你的项目，它是一个纯 Python 包。它的名字就是当你引用它内部任何东西时需要用到的 Python 包名。
* test1/\_\_init\_\_.py: 一个空文件，告诉 Python 该目录是一个 Python 包。
* test1/[settings.py](https://docs.djangoproject.com/zh-hans/2.1/topics/settings/): 项目的设置/配置。
* test1/[urls.py](https://docs.djangoproject.com/zh-hans/2.1/topics/http/urls/): 项目的 URL 声明; 指出了什么样的 URL 调用什么的视图
* test1/[wsgi.py](https://docs.djangoproject.com/zh-hans/2.1/howto/deployment/wsgi/): 一个 WSGI 兼容的 Web 服务器的入口，定义wsgi接口信息，与其他web服务器集成，一般无需改动。

## 建立应用

为了在项目中开发符合架构的实际应用程序，需要建立应用，每个项目可以包含多个应用。应用是一组拥有特定功能的Python包。

python manage.py startapp 应用名称



─search\_app

│ admin.py 管理站点模型的声明文件

│ apps.py 应用信息定义文件，其中生成了类AppConfig，用于定义应用名等Meta数据

│ models.py 添加模型层数据类的文件

│ tests.py 测试代码文件

│ views.py 定义URL响应函数

│ \_\_init\_\_.py 一个空文件，告诉 Python 该目录是一个 Python 包。

│

└─migrations 用于定义引用迁移功能

\_\_init\_\_.py

项目 VS 应用

应用是一个专门做某件事的网络应用程序——比如博客系统，或者公共记录的数据库，或者简单的投票程序。项目则是一个网站使用的配置和应用的集合。项目可以包含很多个应用。应用可以被很多个项目使用。

## 应用配置

1. 修改工程配置文件 Search\_Engine/setting.py
2. INSTALLED\_APPS 添加应用名称

[INSTALLED\_APPS](https://docs.djangoproject.com/zh-hans/2.1/ref/settings/#std:setting-INSTALLED_APPS) 默认包括了以下 Django 的自带应用：

* [django.contrib.admin](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/admin/#module-django.contrib.admin) -- 管理员站点。
* [django.contrib.auth](https://docs.djangoproject.com/zh-hans/2.1/topics/auth/#module-django.contrib.auth) -- 认证授权系统。
* [django.contrib.contenttypes](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/contenttypes/#module-django.contrib.contenttypes) -- 内容类型框架。
* [django.contrib.sessions](https://docs.djangoproject.com/zh-hans/2.1/topics/http/sessions/#module-django.contrib.sessions) -- 会话框架。
* [django.contrib.messages](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/messages/#module-django.contrib.messages) -- 消息框架。
* [django.contrib.staticfiles](https://docs.djangoproject.com/zh-hans/2.1/ref/contrib/staticfiles/#module-django.contrib.staticfiles) -- 管理静态文件的框架。

INSTALLED\_APPS = [  
 'search\_app'  
]

1. MIDDLEWARE 注释 csrf 校验

MIDDLEWARE = [  
 # 'django.middleware.csrf.CsrfViewMiddleware',]

1. DATABASES  修改连库配置默认 sqlite3 改mysql

DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.mysql',  
 'NAME': 'python',  
 'USER': 'root',  
 'PASSWORD': 'zkyr1006',  
 'HOST': '127.0.0.1',  
 'PORT': '3306',  
 }  
}

1. 修改时间和语言

LANGUAGE\_CODE = 'zh-Hans'  
TIME\_ZONE = 'Asia/Shanghai'

1. 允许所有IP访问

ALLOWED\_HOSTS = ['\*']

1. 定义用户信息的数据模型 search\_app/ models.py

定义字段名和数据类型

1. 初始化模型数据库并生成数据库文件

python manage.py migrate

migrate检查 INSTALLED\_APPS 设置，为其中的每个应用创建需要的数据表

1. 显示注册信息修改默认标题 search\_app/admin.py

*from* django.contrib *import* admin  
*from* search\_app.models *import* User  
  
# Register your models here.  
*class* HostAdmin(admin.ModelAdmin):  
 list\_display = [  
 'name',  
 'age',  
 'birthday',  
 'gender',  
 'account',  
 ]  
 search\_fields = ('name',)  
  
admin.site.register(User,HostAdmin)  
admin.AdminSite.site\_header='运维系统管理后台'  
admin.AdminSite.site\_title='运维系统'

1. 添加应用的url访问 Search\_Engine/urls.py

*from* django.contrib *import* admin  
*from* django.urls *import* path  
  
urlpatterns = [  
 path('admin/',admin.site.urls)  
]

1. mysql创建表

## 创建、激活模型

### 创建模型

编辑app中的model.py文件

*class* Question(models.Model):  
 question\_text = models.CharField(max\_length=200)  
 pub\_date = models.DateTimeField('date published')  
*class* Choice(models.Model):  
 question = models.ForeignKey(Question, on\_delete=models.CASCADE)  
 choice\_text = models.CharField(max\_length=200)  
 votes = models.IntegerField(default=0)

每个模型被表示为 [django.db.models.Model](https://docs.djangoproject.com/zh-hans/2.1/ref/models/instances/#django.db.models.Model) 类的子类。每个模型有一些类变量，它们都表示模型里的一个数据库字段。

每个字段都是 Field 类的实例 - 比如，字符字段被表示为 CharField ，日期时间字段被表示为 DateTimeField 。这将告诉 Django 每个字段要处理的数据类型。

每个 Field 类实例变量的名字（例如 question\_text 或 pub\_date ）也是字段名，数据库会将它们作为列名。

使用 ForeignKey 定义了一个关系。这将告诉 Django，每个 Choice 对象都关联到一个 Question 对象。Django 支持所有常用的数据库关系：多对一、多对多和一对一。

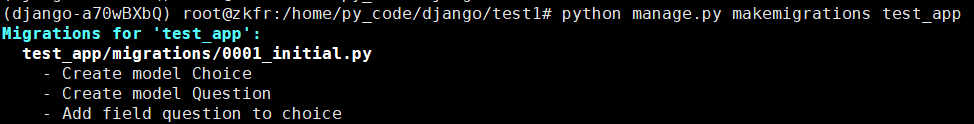
### 激活模型

1. 在项目test1下修改settings.py，NSTALLED\_APPS 子项添加点式路径app的Config

INSTALLED\_APPS = [  
 'test\_app.apps.TestAppConfig'  
]

1. 为模型的改变生成迁移文件

**python manage.py makemigrations test\_app**



通过运行 makemigrations 命令，Django 会检测你对模型文件的修改（在这种情况下，你已经取得了新的），并且把修改的部分储存为一次 *迁移*。

迁移是 Django 对于模型定义（也就是你的数据库结构）的变化的储存形式，其实只是一些磁盘上的文件。被储存在 test\_app/migrations/0001\_initial.py 里

1. 应用数据库迁移

**python manage.py migrate**

migrate 命令选中所有还没有执行过的迁移（Django 通过在数据库中创建一个特殊的表 django\_migrations 来跟踪执行过哪些迁移）并应用在数据库上 - 也就是将你对模型的更改同步到数据库结构上。

迁移是非常强大的功能，它能让你在开发过程中持续的改变数据库结构而不需要重新删除和创建表 - 它专注于使数据库平滑升级而不会丢失数据。

tip：

**python manage.py check**

这个命令帮助你检查项目中的问题，并且在检查过程中不会对数据库进行任何操作。

## 运行简易开发服务器

**python manage.py runserver** #默认端口8000

**python manage.py runserver 8080**  #指定端口8080

**python manage.py runserver 0:8000** # 0.0.0.0:8000

django开发服务是可用在开发期间的，一个内建的，轻量的web服务。在准备发布产品之前，无需进行产品级 Web 服务器（比如 Apache）的配置工作。

不要在正式的应用布署环境中使用它。 在同一时间，该服务器只能可靠地处理一次单个请求，并且没有进行任何类型的安全审计。

runserver在需要的情况下会对每一次的访问请求重新载入一遍 Python 代码。所以你不需要为了让修改的代码生效而频繁的重新启动服务器。然而，一些动作，比如添加新文件，将不会触发自动重新加载，这时你得自己手动重启服务器。

## Django命令行

**python manage.py shell**

manage.py 会设置 DJANGO\_SETTINGS\_MODULE 环境变量，这个变量会让 Django 根据 test1/settings.py 文件来设置 Python 包的导入路径。

## 管理页面

1. 创建一个管理员账号

**python manage.py createsuperuser**

1. 向管理页面加入应用

修改app的admin.py文件

*from* django.contrib *import* admin  
*from* .models *import* Question  
admin.site.register(Question)

* 应用的表单是在model文件中自动生成的
* 不同的字段类型（日期时间字段 DateTimeField 、字符字段 CharField）会生成对应的 HTML 输入控件。每个类型的字段都知道它们该如何在管理页面里显示自己。

# 模型