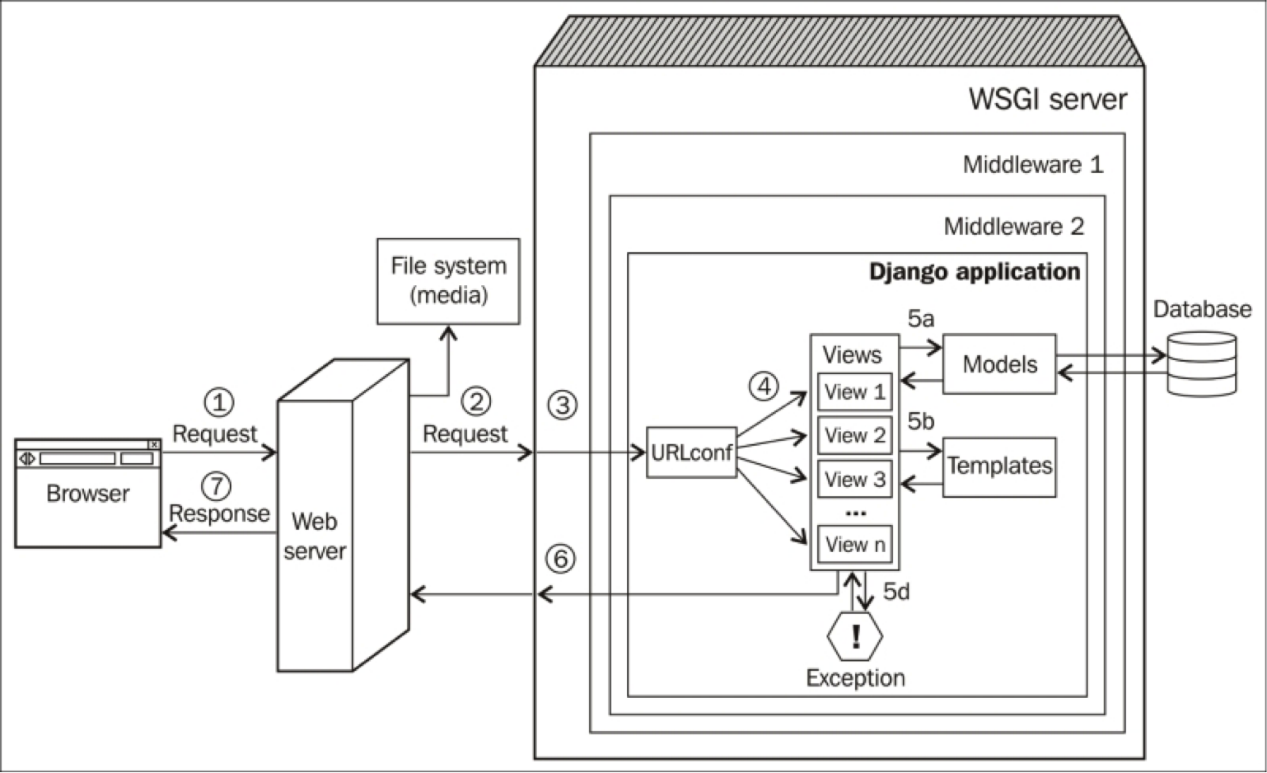
# Django基础



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

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

## 安装django

pip3 install django

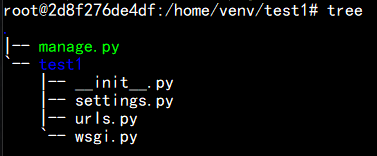
查看版本

import django

django.get\_version()

## 创建项目

django-admin startproject 项目名



**目录说明：**

mydjango: 项目的容器。

manage.py: 命令行工具，以各种方式与Django 项目交互。不需要编辑

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

mydjango/settings.py: 项目的设置/配置。

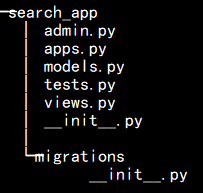
mydjango/urls.py: 项目的 URL 声明; 指出了什么样的 URL 调用什么的视图

mydjango/wsgi.py: 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

## 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 = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'search\_app'  
]

1. MIDDLEWARE 注释 csrf 校验

MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 # 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]

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

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

## 运行开发服务器

python manage.py runserver #(0.0.0.0:8000)

在端口8000启动一个本地服务器, 并且只能从你的这台电脑连接和访问。

如果想和其他开发人员共享同一开发站点的话，通过指定一个 IP 地址，可以告诉服务器–允许非本地连接访问。本地网络中的其它计算机就可以在浏览器中访问你的 IP 地址了。

0.0.0.0这个 IP 地址，告诉服务器去侦听任意的网络接口。

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。

# 模型