《Django Web框架教学笔记》

目录

Django框架的介绍

起源&现状

- 2005年发布,采用Python语言编写的开源web框架
- 早期的时候Django主做新闻和内容管理的
- 一个重量级的 Python Web框架，Django 配备了常用的大部分组件
    1. 基本配置
    2. 路由系统
    3. 原生HTML模板系统
    4. 视图 view
    5. Model模型,数据库连接和ORM数据库管理
    6. 中间件
    7. Cookie & Seesion
    8. 分页
    9. 数据库后台管理系统admin
- Django的用途
    - 网站后端开发
    - 微信公众号、微信小程序等后台开发

    - 基于HTTP/HTTPS协议的后台服务器开发
        - 在线语音/图像识别服务器
        - 在线第三方身份验证服务器等
- Django的版本
    - 最新版本:3.0.x
    - 当前教学版本:2.2.12
- Django的官网
    - 官方网址: http://www.djangoproject.com
    - 中文文档(第三方):
        - https://yiyibooks.cn/

Django的安装

- 查看已安装的版本

```
>>> import django
>>> print(django.VERSION)
(2, 2, 12, 'final', 0)
```

- 安装
    1. 在线安装
        - `$ sudo pip3 install django` 安装django的最新版本
        - 或
        - `$ sudo pip3 install django[==版本]` 安装django的指定版本
        - 如:
            - `$ sudo pip3 install django==2.2.12
    2. 离线安装
        - 下载安装包:
        - 安装离线包
            - `$ tar -xvf Django-2.2.12.tar.gz`
            - `$ cd Django-2.2.12`
            - `$ sudo python3 setup.py install`
- Django的卸载
- $ pip3 uninstall django
- Django 的开发环境
    - Django 2.2.12 支持 3.5，3.6，3.7，3.8

创建Django项目

创建项目的指令

- $ django-admin startproject 项目名称
- 如:
    - $ django-admin startproject mysite1

- 运行

```
$ cd mysite1
$ python3 manage.py runserver
# 或
$ python3 manage.py runserver 5000  # 指定只能本机使用127.0.0.1的5000端口访问本机
```

Django项目的目录结构

- 示例:

```
$ django-admin startproject mysite1
$ tree mysite1/
mysite1/
├── manage.py
└── mysite1
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py

1 directory, 5 files
```

- 项目目录结构解析:
  - manage.py
    - 此文件是项目管理的主程序,在开发阶段用于管理整个项目的开发运行的调式
    - `manage.py` 包含项目管理的子命令, 如:
      - `python3 manage.py runserver` 启动服务
      - `python3 manage.py startapp` 创建应用
      - `python3 manage.py migrate` 数据库迁移
      - `...`
  - mysite1 项目文件夹
    - 项目包的主文件夹(默认与项目名称一致)

    1. `__init__.py`
       - 包初始化文件,当此项目包被导入(import)时此文件会自动运行
    2. `wsgi.py`
       - WSGI 即 Web Server Gateway Interface
       - WEB服务网关接口的配置文件，仅部署项目时使用
    3. `urls.py`
       - 项目的主路由配置文件，所有的动态路径必须先走该文件进行匹配
    4. `settings.py`
       - Django项目的配置文件, 此配置文件中的一些全局变量将为Django框架的运行传递一些参数
       - setting.py 配置文件,启动服务时自动调用，
       - 此配置文件中也可以定义一些自定义的变量用于作用全局作用域的数据传递

`settings.py` 文件介绍

https://docs.djangoproject.com/en/2.2/ref/settings/

1. `BASE_DIR`
    - 用于绑定当前项目的绝对路径(动态计算出来的), 所有文件都可以依懒此路径
2. `DEBUG`
    - 用于配置Django项目的启动模式, 取值:
        1. True 表示开发环境中使用 `调试模式` (用于开发中)
        2. False 表示当前项目运行在 `生产环境中` (不启用调试)
3. `ALLOWED_HOSTS`
    - 设置允许访问到本项目的host请求头的值,取值:
        1. [] 空列表,表示只有host请求头为 `127.0.0.1` , `localhost` 能访问本项目 - DEBUG = True时生效
        2. ['*']，表示任何请求头的host都能访问到当前项目
        3. ['192.168.1.3', '127.0.0.1'] 表示只有当前两个host头的值能访问当前项目
        - 注意:
            - 如果要在局域网其它主机也能访问此主机,启动方式应使用如下模式:
    - `python3 manage.py runserver 0.0.0.0:5000` # 指定网络设备如果内网环境下其他主机想正常访问该站点，需加`ALLOWED_HOSTS = ['内网ip']
4. `INSTALLED_APPS`
    - 指定当前项目中安装的应用列表
5. `MIDDLEWARE`
    - 用于注册中间件
6. `TEMPLATES`
    - 用于指定模板的配置信息
7. `DATABASES`
    - 用于指定数据库的配置信息
8. `LANGUAGE_CODE`
    - 用于指定语言配置
    - 取值:
        - 英文 : `"en-us"`
        - 中文 : `"zh-Hans"`
9. `TIME_ZONE`
    - 用于指定当前服务器端时区
    - 取值:
        - 世界标准时间: `"UTC"`
        - 中国时区 : `"Asia/Shanghai"`
10. `ROOT_URLCONF`
    - 用于配置根级 url 配置 'mysite1.urls'
    - 如:
        - `ROOT_URLCONF = 'mysite1.urls'`

> 注: 此模块可以通过 `from django.conf import settings` 导入和使用

URL 介绍

URL定义

- URL 即统一资源定位符 Uniform Resource Locator
- 作用:
  - 用来表示互联网上某个资源的地址。
- 说明:
  - 互联网上的每个文件都有一个唯一的URL，它包含的信息指出文件的位置以及浏览器应该怎么处理它。
- URL的一般语法格式为：

```
protocol :// hostname[:port] / path [?query][#fragment]
```

- 如:

```
http://tts.tmooc.cn/video/showVideo?menuId=657421&version=AID201908#subject
```

- 说明:
  - protocol（协议）
    - http 通过 HTTP 访问该资源。 格式 `HTTP://`
    - https 通过安全的 HTTPS 访问该资源。 格式 `HTTPS://`
    - file 资源是本地计算机上的文件。格式: `file:///`
    - ...
  - hostname（主机名）
    - 是指存放资源的服务器的域名系统(DNS) 主机名、域名 或 IP 地址。
  - port（端口号）
    - 整数，可选，省略时使用方案的默认端口；
    - 各种传输协议都有默认的端口号，如http的默认端口为80。
  - path（路由地址）
    - 由零或多个“/”符号隔开的字符串，一般用来表示主机上的一个目录或文件地址。路由地址决定了服务器端如何处理这个请求
  - query(查询)
    - 可选，用于给动态网页传递参数，可有多个参数，用“&”符号隔开，每个参数的名和值用“=”符号隔开。
  - fragment（信息片断）
    - 字符串，用于指定网络资源中的片断。例如一个网页中有多个名词解释，可使用fragment直接定位到某一名词解释。
  - 注: [] 代表其中的内容可省略

Django如何处理一个URL对应的请求

```
浏览器  http://127.0.0.1:8000/page/2003/

1,Django 从配置文件中 根据 ROOT_URLCONF 找到 主路由文件；默认情况下，该文件在 项目同名目录下的urls; 例如 mysite1/mysite1/urls.py
2,Django 加载 主路由文件中的 urlpatterns 变量
3,依次匹配 urlpatterns 中的 URL， 匹配到第一个合适的中断后续匹配
4,匹配成功 - 调用对应的视图函数处理请求，返回响应
5,匹配失败 - 返回404响应

主路由-urls.py样例
from django.urls import path
```

```
from . import views
urlpatterns = [
    path('admin/', admin.site.urls)
    path('page/2003/', views.page_2003),
    path('page/2004/', views.page_2004),
]
```

视图函数(view)

- 视图函数是用于接收一个浏览器请求并通过HttpResponse对象返回数据的函数。此函数可以接收浏览器请求并根据业务逻辑返回相应的内容给浏览器
- 视图处理的函数的语法格式:

```
def xxx_view(request[, 其它参数...]):
    return HttpResponse对象
```

- 参数:
  - request用于绑定HttpRequest对象，通过此对象可以获取浏览器的参数和数据
- 返回值

  HttpResponse的对象；Django会提供一系列的response对象；

- 示例:
  - 视图处理函数 `views.py`

```
# file : <项目同名文件夹下>/views.py
from django.http import HttpResponse
def page1_view(request):
    html = "<h1>这是第1个页面</h1>"
    return HttpResponse(html)
```

Django 路由配置

- settings.py 中的 `ROOT_URLCONF` 指定了主路由配置列表urlpatterns的文件位置
- urls.py 主路由配置文件

```
# file : <项目同名文件夹>/urls.py
urlpatterns = [
    path('admin/', admin.site.urls),
    ...  # 此处配置主路由
]
```

path() 函数

- 用于描述路由与视图函数的对应关系
- 模块
  - `from django.urls import path`
- 语法:
  - path(route, views, name=None)
  - 参数:

1. route: 字符串类型，匹配的请求路径
2. views: 指定路径所对应的视图处理函数的名称
3. name: 为地址起别名，在模板中地址反向解析时使用

- 练习
  - 建立一个小网站:
    - 输入网址: http://127.0.0.1:8000, 在网页中输出 : 这是我的首页
    - 输入网址: http://127.0.0.1:8000/page/1, 在网页中输出 : 这是编号为1的网页
    - 输入网址: http://127.0.0.1:8000/page/2, 在网页中输出 : 这是编号为2的网页
    - 思考
      - 建立如上一百个网页该怎么办?

path转换器

语法： <转换器类型:自定义名>

作用：若转换器类型匹配到对应类型的数据，则将数据按照关键字传参的方式传递给视图函数

转换器	效果	案例
str	匹配除了 `'/'` 之外的非空字符串	"v1/users/<str:username>"匹配 /v1/users/guoxiaonao
int	匹配0或任何正整数。返回一个 int	"page/<int:page>" 匹配 /page/100
slug	匹配任意由 ASCII 字母或数字以及连字符和下划线组成的短标签	"detail/<slug:sl>" 匹配 /detail/this-is-django
path	匹配非空字段，包括路径分隔符 `'/'`	"v1/users/<path:ph>"匹配 /v1/goods/a/b/c

- 练习:
  - 定义一个路由的格式为:
    - http://127.0.0.1:8000/整数/操作字符串/整数
  - 从路由中提取数据，做相应的操作后返回给浏览器
  - 如:

```
输入: 127.0.0.1:8000/100/add/200
    页面显示结果：300
输入: 127.0.0.1:8000/100/sub/200
    页面显示结果：-100
输入: 127.0.0.1:8000/100/mul/200
    页面显示结果：20000
```

re_path()函数

- 在url 的匹配过程中可以使用正则表达式进行精确匹配
- 语法:
  - re_path(reg, view, name=xxx)
  - 正则表达式为命名分组模式 `(?P<name>pattern)` ； 匹配提取参数后用关键字传参方式传递给视图函数

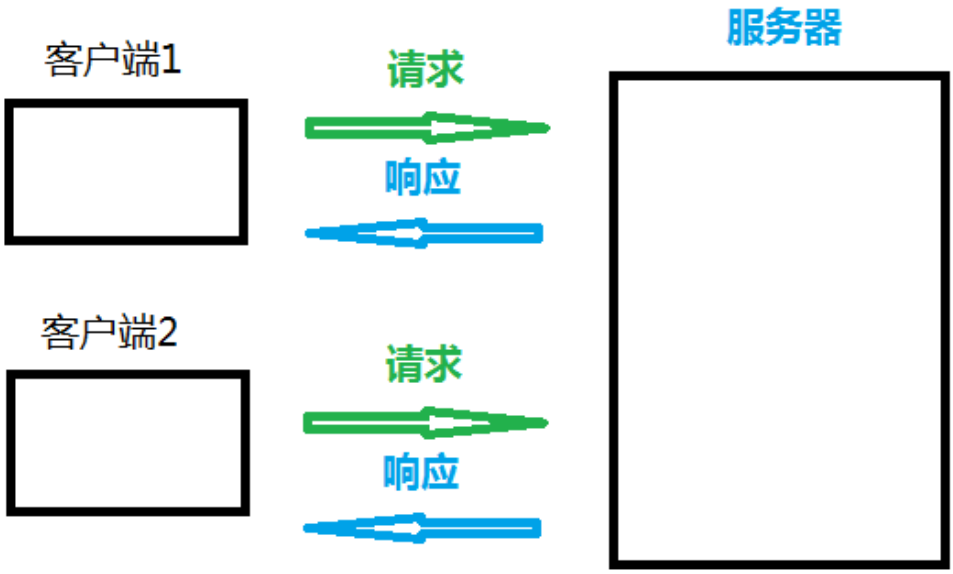
- 示例:
  - 路由配置文件

```
# file : <项目同名文件夹>/urls.py
# 以下示例匹配
# 可匹配 http://127.0.0.1:8000/20/mul/40
# 不可匹配 http://127.0.0.1:8000/200/mul/400
urlpatterns = [
    path('admin/', admin.site.urls),
    re_path(r'^(?P<x>\d{1,2})/(?P<op>\w+)/(?
P<y>\d{1,2})$',views.cal_view),
]
```

- 练习:
  - 访问地址:
    - http://127.0.0.1:8000/birthday/四位数字/一到两位数字/一到两位数字
    - http://127.0.0.1:8000/birthday/一到两位数字/一到两位数字/四位数字
  - 最终输出: 生日为: xxxx年xx月xx日
  - 如:
    输入网址: http://127.0.0.1:8000/birthday/2015/12/11
    显示为: 生日为:2015年12月11日
    输入网址: http://127.0.0.1:8000/birthday/2/28/2008
    显示为: 生日为:2008年2月28日

HTTP协议的请求和响应

- 请求是指浏览器端通过HTTP协议发送给服务器端的数据
- 响应是指服务器端接收到请求后做相应的处理后再回复给浏览器端的数据



HTTP 请求

```
POST /v1/tokens HTTP/1.1      -> 起始行

Host: 127.0.0.1:8000          -> headers
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:74.0) Gecko/20100101
Firefox/74.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 58
Origin: http://127.0.0.1:7000
Connection: keep-alive
Referer: http://127.0.0.1:7000/dadashop/templates/login.html

{"username":"guoxiaonao"}     -> body
```

- 根据HTTP标准，HTTP请求可以使用多种请求方法。
- HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法(最常用)
- HTTP1.1新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。
- HTTP1.1 请求详述

序号	方法	描述
1	GET	请求指定的页面信息，并返回实体主体。
2	HEAD	类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
3	POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
4	PUT	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	请求服务器删除指定的页面。
6	CONNECT	HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
7	OPTIONS	允许客户端查看服务器的性能。
8	TRACE	回显服务器收到的请求，主要用于测试或诊断。

- HttpRequest对象
  - 视图函数的第一个参数是HttpRequest对象
  - 服务器接收到http协议的请求后，会根据请求数据报文创建HttpRequest对象
  - HttpRequest属性
    - path_info: URL字符串
    - method：字符串，表示HTTP请求方法，常用值：'GET'、'POST'
    - GET：QueryDict查询字典的对象，包含查询字符串的所有数据
    - POST：QueryDict查询字典的对象，包含post表单提交方式的所有数据
    - FILES：类似于字典的对象，包含所有的上传文件信息

    - COOKIES：Python字典，包含所有的cookie，键和值都为字符串
    - session：似于字典的对象，表示当前的会话
    - body: 字符串，请求体的内容(POST或PUT)
    - scheme : 请求协议('http'/'https')
    - request.get_full_path() : 请求的完整路径
    - request.get_host() : 请求的主机
    - request.META : 请求中的元数据(消息头)
        - request.META['REMOTE_ADDR'] : 客户端IP地址

HTTP 响应

```
HTTP/1.0 200 OK                          -> 起始行

Date: Sat, 21 Mar 2020 09:44:15 GMT      -> headers
Server: WSGIServer/0.2 CPython/3.6.8
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Content-Length: 217
Vary: Origin
Access-Control-Allow-Origin: *

{"code": 200, "username": "guoxiaonao"} -> body
```

- 当浏览者访问一个网页时，浏览者的浏览器会向网页所在服务器发出请求。当浏览器接收并显示网页前，此网页所在的服务器会返回一个包含HTTP状态码的信息头用以响应浏览器的请求。
- HTTP状态码的英文为HTTP Status Code。
- 下面是常见的HTTP状态码：
    - 200 - 请求成功
    - 301 - 永久重定向-资源（网页等）被永久转移到其它URL
    - 302 - 临时重定向
    - 404 - 请求的资源（网页等）不存在
    - 500 - 内部服务器错误
- HTTP状态码分类
    - HTTP状态码由三个十进制数字组成，第一个十进制数字定义了状态码的类型，后两个数字没有分类的作用。HTTP状态码共分为5种类型：

分类	分类描述
1**	信息，服务器收到请求，需要请求者继续执行操作
2**	成功，操作被成功接收并处理
3**	重定向，需要进一步的操作以完成请求
4**	客户端错误，请求包含语法错误或无法完成请求
5**	服务器错误，服务器在处理请求的过程中发生了错误

- Django中的响应对象HttpResponse:

- 构造函数格式:
  - `HttpResponse(content=响应体, content_type=响应体数据类型, status=状态码)`
- 作用:
  - 向客户端浏览器返回响应，同时携带响应体内容
- 参数:
  - content：表示返回的内容。
  - status_code：返回的HTTP响应状态码(默认为200)。
  - content_type：指定返回数据的的MIME类型(默认为"text/html")。浏览器会根据这个属性，来显示数据。如果是text/html，那么就会解析这个字符串，如果text/plain，那么就会显示一个纯文本。
    - 常用的Content-Type如下：
      - `'text/html'`（默认的，html文件）
      - `'text/plain'`（纯文本）
      - `'text/css'`（css文件）
      - `'text/javascript'`（js文件）
      - `'multipart/form-data'`（文件提交）
      - `'application/json'`（json传输）
    - `'application/xml'`（xml文件）

    > 注： 关键字MIME(Multipurpose Internet Mail Extensions)是指多用途互联网邮件扩展类型。

- HttpResponse 子类

类型	作用	状态码
HttpResponseRedirect	重定向	302
HttpResponseNotModified	未修改	304
HttpResponseBadRequest	错误请求	400
HttpResponseNotFound	没有对应的资源	404
HttpResponseForbidden	请求被禁止	403
HttpResponseServerError	服务器错误	500

Django处理GET和POST请求

- 无论是GET还是POST，统一都由视图函数接收请求，通过判断request.method 区分具体的请求动作
- 样例:

```
if request.method == 'GET':
    处理GET请求时的业务逻辑
elif request.method == 'POST':
    处理POST请求的业务逻辑
else:
    其他请求业务逻辑
```

GET处理

- GET请求动作，一般用于向服务器获取数据
- 能够产生GET请求的场景：
  - 浏览器地址栏中输入URL,回车后
  - `<a href="地址?参数=值&参数=值">`
  - form表单中的method为get

```
<form method='get' action="/user/login">
    姓名:<input type="text" name="uname">
</form>
```

- GET请求方式中，如果有数据需要传递给服务器，通常会用查询字符串(Query String)传递 【注意：不要传递敏感数据】
  - URL 格式: `xxx?参数名1=值1&参数名2=值2...`
    - 如: `http://127.0.0.1:8000/page1?a=100&b=200`
  - 服务器端接收参数

    获取客户端请求GET请求提交的数据

```
request.GET['参数名']  # QueryDict
request.GET.get('参数名','默认值')
request.GET.getlist('参数名')
# mypage?a=100&b=200&c=300&b=400
# request.GET=QueryDict({'a':['100'], 'b':['200','400'], 'c':['300']})
# a = request.GET['a']
# b = request.GET['b']  # Error
```

- 练习:
  - 访问地址:http://127.0.0.1:8000/birthday?year=四位整数&month=整数&day=整数
  - 最终输出: 生日为: xxxx年xx月xx日
  - 如:
    - 输入网址: http://127.0.0.1:8000/birthday?year=2025&month=12&day=11
    - 显示为: 生日为:2025年12月11日

POST处理

- POST请求动作，一般用于向服务器提交大量数据
- 客户端通过表单等POST请求将数据传递给服务器端,如:

```
<form method='post' action="/login">
    姓名:<input type="text" name="username">
    <input type='submit' value='登陆'>
</form>
```

  - form 表单的name属性

        - 在form表单控件提交数据时，会自动搜索本表单控件内部的子标签的name属性及相应的值，再将这些名字和值以键-值对的形式提交给action指定的服务器相关位置
        - 在form内能自动搜集到的name属性的标签的控件有

```
<input name='xxx'>
<select name='yyy'></select>
<textarea name='zzz'></textarea>
```

- 服务器端接收参数
    - 通过 request.method 来判断是否为POST请求,如:

```
if request.method == 'POST':
    处理POST请求的数据并响应
else:
    处理非POST 请求的响应
```

- 使用post方式接收客户端数据

```
request.POST['参数名']  # request.POST 绑定QueryDict
request.POST.get('参数名','')
request.POST.getlist('参数名')
```

- 取消csrf验证,否则Django将会拒绝客户端发来的POST请求
    - 取消 csrf 验证
        - 禁止掉 settings.py 中 MIDDLEWARE 中的 CsrfViewsMiddleWare 的中间件

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