《Django Web框架》

```
主讲: 魏明择
版权: tedu.cn
时间: 2019
```

- 2019年3月最新更新的内容
- 此课程很重要

目录

day01

Django框架的介绍

- 2005年发布,采用Python语言编写的开源框架
- 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. 路由
2. URL解析
3. 原生HTML模板系统
4. 数据库连接和ORM数据库管理
5. 用户管理认证系统
6. ORM模型系统
7. 电子邮件发送系统
8. CSRF 跨站点请求伪造的保护
9. 表单验证
10. 数据库后台管理系统
11. 自带强大的后台管理功能

- Django的用途
    - 网站后端开发
    - 微信公众号后台开发
    - 微信小程序后台开发
    - 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:2.2.x
    - 当前教学版本:1.11.8

- Django的官网
    - 官网: http://www.djangoproject.com
    - 中文文档(第三方):
        - https://yiyibooks.cn/
        - http://djangobook.py3k.cn/

Django的安装

- 查看已安装的版本

```
>>> import django
>>> print(django.VERSION)
(1, 11, 8, '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]`
        - 如:
            - `$ sudo pip3 install django==1.11.8` (安装django的指定版本)

2. 离线安装
    - 下载安装包:
    - 安装离线包
        - `$ tar -xvf Django-1.11.8.tar.gz`
        - `$ cd Django-1.11.8`
        - `$ sudo python3 setup.py install`
3. 用wheel离线安装
    - 下载安装包:
        - `pip3 download -d /home/weimz/django_packs django`
    - 安装离线包
        - $ pip install Django-1.11.8.whl

- Django的卸载
    - $ pip uninstall django

Django框架开发

创建项目的指令

- $ django-admin startproject 项目名称
- 如:
    - $ django-admin startproject mywebsite1
- 运行

```
$ cd mywebsite1
$ python3 manage.py runserver
# 或
$ python3 manage.py runserver 5000  # 指定网络设备入口有端口5000
$ python3 manage.py runserver 192.168.1.111:5000  # 指定网络设备入口有端
口5000
```

Django项目的目录结构

- 示例:

```
$ django-admin startproject mywebsite1
$ tree mywebsite1/
mywebsite1/
├── manage.py
└── mywebsite1
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py

1 directory, 5 files
```

- 目录解析:
    - 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
            - `...`
    - mywebsite1 项目包文件夹
        - 项目包的主文件夹(默认与项目名称一致)
        1. `__init__.py`
            - 项目初始化文件,服务启动时自动运行
        2. `wsgi.py`
            - WEB服务网关接口的配置文件，仅部署项目时使用
        3. `urls.py`
            - 项目的基础路由配置文件，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

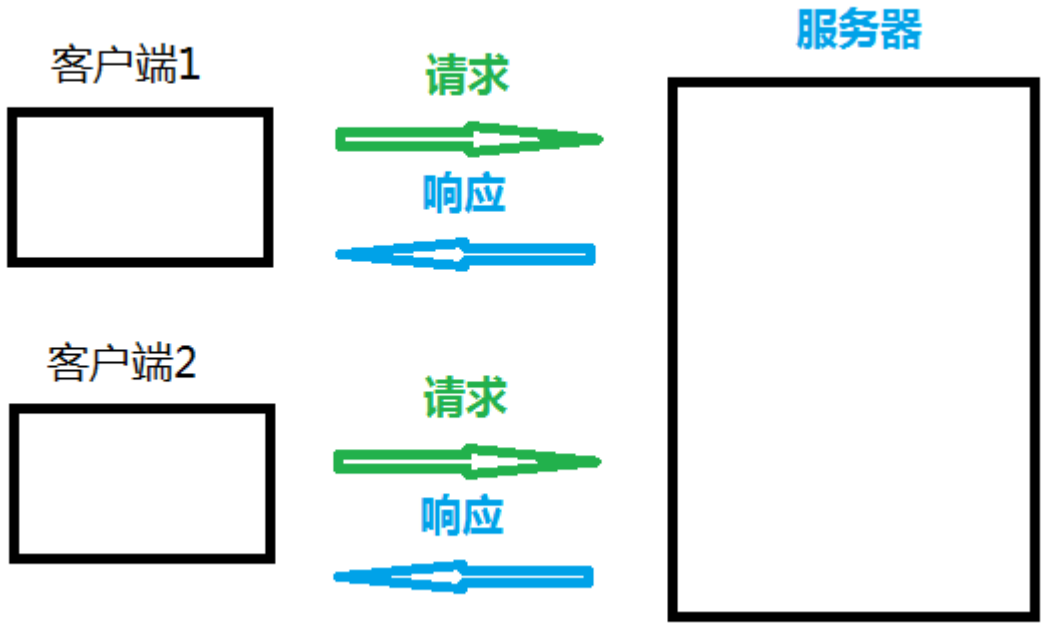
    1. `BASE_DIR`
        - 用于绑定=当前项目的绝对路径(动态计算出来的), 所有文件都可以依懒此路径
    2. `DEBUG` 用于配置Django项目的启用模式
        1. True 表示开发环境中使用 调试模式(用于开发中)
        2. False 表示当前项目运行在生产环境中(不启用调试)
    3. `ALLOWED_HOSTS`
        - 设置允许访问到本项目的网络地址列表
        - 取值:
            1. 如果为空列表, 表示只有`127.0.0.1`能访问本项目
            2. ['*'] 表示任何网络地址都能访问到当前项目 如: localhost / 127.0.0.1 / 0.0.0.0 / IP 局域网内也允许访问 注意: 如果要在局域网内访问的话,启动方式: python3 manage.py runserver 0.0.0.0:端口号
    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:
            - 中文 : "zh-Hans"
    9. TIME_ZONE
        - 用于指定当前服务器端时区
        - 取值:
            - 中国时区 : "Asia/Shanghai"
    10. ROOT_URLCONF
        - 用于配置根级 url 配置 'mywebsite1.urls'
        - 如:
            - ROOT_URLCONF = 'mywebsite1.urls'
  - 缺省配置
        - 模块
            - import django.conf.global_settings
        - Linux下文件位置:
            - /usr/lib/python3/dist-packages/django/conf/global_settings.py

HTTP协议

- 在TCP/IP协议中位于应用层的协议

HTTP协议的请求和响应



请求request 和 响应 response

```
+--客户端1---+  request    +---- 服务器 ----+
|          -+------------+->              |
|           |  response  |                |
|         <-+------------+--              |
```

```
+-----------+              |                |
                           |                |
+--客户端2---+  request    |                |
|          -+-------------+->              |
|           |  response   |                |
|         <-+-------------+--              |
+-----------+              |                |
    ....                   +---------------+
```

URL 介绍

- url 即统一资源定位符 Uniform Resource Locator
- 作用:
  - 对互联网上得到的资源的位置和访问方法的一种简洁的表示，是互联网上标准资源的地址。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。
- 如:
  - http://www.tmooc.cn/live/
  - https://www.baidu.com/
  - http://www.tmooc.cn/course/300246.shtml
  - http://www.tedu.com:8000/article/python.html

路由

- 不看域名部分

Django 中的 URL路由配置

1. settings.py 中的ROOT_URLCONF
   - 通过ROOT_URLCONF指定顶级url的配置
     - 默认存在于主文件夹内,主路由配置文件
2. urlpatterns 是一个 url() 实例的列表,如:

```
# file : <项目名>/urls.py
urlpatterns = [
    url(r'^admin/', admin.site.urls),
]
```

   - 作用: 该文件会包含 urlpatterns 的列表用于表示路由-视图映射,通过 url() 表示具体映射
3. url() 函数
   - 用于描述路由与视图函数的对应关系
   - 模块
     - `from django.conf.urls import url`

- 语法:
  - url(regex,views,kwargs=None,name=None)
    1. regex: 字符串类型,匹配的请求路径,允许是正则表达式
    2. views: 指定路径所对应的视图处理函数的名称
    3. kwargs: 向视图中传递的参数
    4. name: 为地址起别名,反向解析时使用
- 注:
  - 每个正则表达式前面的r表示'\'不转义的原始字符串
  - 当urlpatterns内有多个url对象时,按自上而下的顺序进行配置，一但有 路由与url配置成功,则后面的所有url被忽略

视图view

- 用于接收请求,处理请求并做出响应
- 视图处理的函数的语法格式

```
def xxx(request[, 其它参数...]):
    return 响应对象
```

- 示例:

```
# file : <项目名>/urls.py
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^/', views.homepage),
    url(r'^page1$', views.page1),
    url(r'^page2$', views.page2),
]

# file : <项目名>/views.py
from . import views
from django.http import HttpResponse

def homepage(request):
    return HttpResponse("这是首页")
def page1(request):
    return HttpResponse("这是第1个页面")
def page2(request):
    return HttpResponse("这是第2个页面")
```

- 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/page1, 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/page2, 在网页中输出：这是编号为2的网页

url和带有参数的视图函数

- 在视图函数内，可以用正规表达式分组()提取参数后传送给视图函数
- 一个分组表示一个参数,多个参数需要使用多个分组,并且使用个/隔开
    - 如:
        - http://127.0.0.1:8000/year/2018
        - http://127.0.0.1:8000/year/2019
        - http://127.0.0.1:8000/year/???? # 四位数字
    - 分组:
- 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
    - 最终输出: 生日为: xxxx年xx月xx日
    - 如: 输入网址: http://127.0.0.1:8000/2015/12/11 显示为: 生日为:2015年12月11日

url 正则表达式命名分组(?pxxx) 和 带有参数的视图函数

- 在url 的正则表达式中可以使用命名分组(捕获分组)
- 正表达式分名的名称必须在view中以关键字传参方式传入，因此视图函数分能接收此参数
- 每个捕获的参数都作为一个普通的python字符串传递给视图
- urlpatterns中的每个正则表达式在第一次访问它们时被编译，这使得系统相当快
- 示例:

```
    # 以下示例匹配
    # http://127.0.0.1:8000/person/weimingze/35
    # http://127.0.0.1:8000/person/shibowen/29
    # http://127.0.0.1:8000/person/xiaowei/9
    # file : <项目名>/urls.py
    urlpatterns = [
        url(r'^admin/', admin.site.urls),
        url(r'^person/(?P<name>\w+)/(?P<age>\d{1,2})', views.person_info),
    ]

    # file : <项目名>/views.py
    from . import views
    from django.http import HttpResponse

    def person_info(request, name, age):
        return HttpResponse("姓名:" + name + " 年龄:" + age)
```

请求和响应

HTTP 请求

- 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：字符串，表示请求的路由信息
        - method：字符串，表示HTTP请求方法，常用值：'GET'、'POST'
        - encoding：字符串，表示提交的数据的编码方式
            - 如果为None则表示使用浏览器的默认设置，一般为'utf-8'
            - 这个属性是可写的，可以通过修改它来修改访问表单数据使用的编码，接下来对属性的任何访问将使用新的encoding值
        - GET：QueryDict查询字典的对象，包含get请求方式的所有数据
        - POST：QueryDict查询字典的对象，包含post请求方式的所有数据
        - FILES：类似于字典的对象，包含所有的上传文件
        - 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)
        - environ: 字符串,客户端运行的环境变量信息
        - scheme : 请求协议('http'/'https')
        - path_info: URL字符串
        - 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
            - request.META['HTTP_REFERER'] : 请求源地址

- 请求示例

```
# file : urls.py
from django.conf.urls import url
from django.contrib import admin

from . import views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^show_info', views.show_info),
]

# file : views.py
from django.http import HttpResponse

def show_info(request):
    html = '<p>' + "请求方式:" + request.method + '</p>'
    html += '<p>' + "request.scheme:" + request.scheme + '</p>'
    html += '<p>' + "request.GET:" + str(request.GET) + '</p>'
    html += '<p>' + "request.POST:" + str(request.POST) + '</p>'
    html += '<p>' + "request.COOKIES:" + str(request.COOKIES) + '</p>'
    html += '<p>' + "request.scheme:" + request.scheme + '</p>'
    html += '<p>' + "request.META:" + str(request.META) + '</p>'
    html += '<p>' + "request.META['REMOTE_ADDR']:" +
str(request.META['REMOTE_ADDR']) + '</p>'

    return HttpResponse(html)
```

HTTP 响应

- 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（server header）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 - 资源（网页等）被永久转移到其它URL
    - 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

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。
        - 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。

- 其它HttpResponse响应对象
    - HttpResponseRedirect 重定响 状态码 301
    - HttpResponseNotModified 未修改 状态码 304
    - HttpResponseBadRequest 错误请求 状态码 400
    - HttpResponseNotFound 没有对应的资源 状态码 404
    - HttpResponseForbidden 请求被禁止 状态码 403
    - HttpResponseServerError 服务器错误 状态码 500

GET方式传参

- GET请求方式中可以通过查询字符串（Query String） 将数据传递给服务器

GET方式传参参数获取(查询字符串 Query String)

- 客户端传递参数给服务器端
    - URL 格式: 网址?参数名1=值1&参数名2=值2...
- 服务器端接收参数
    1. 判断 request.method 的值判断请求方式是否是get请求

```
if request.method == 'GET':
    去往指定的模板进行显示
else:
    接收其它请求提交的数据
```

    2. 获取客户端请求GET请求提交的数据
        1. 语法

```
request.GET['参数名']
request.GET.get('参数名','默认值')
request.GET.getlist('参数名')
```

        2. 能够产生get请求方式的场合
            1. 地址栏手动输入, 如: http://www.sina.com.cn/?a=100&b=200
            2. <a href="地址?参数=值&参数=值">
            3. 表单中的method为get

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

- 练习:
    - 访问地址:http://127.0.0.1:8000/birthday?year=四位数字&month=数字&day=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?year=2015&month=12&day=11 显示为: 生日为:2015年12月11日

POST传递参数的获取

- 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="/user/login">
    姓名:<input type="text" name="username">
</form>
```

- 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

    1. 方法

```
request.POST['参数名']
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表求
    - 取消 csrf 验证
        - 删除 settings.py 中 MIDDLEWARE 中的 CsrfViewsMiddleWare 的中间件

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

Django中的应用 - app

什么是应用(app)

- 应用在Django项目中是一个独立的业务模块,可以包含自己的路由,视图,... ...
- Django中,主文件夹是不处理用户具体请求的.主文件夹的作用是做项目的初始化以及请求的分发(分布式请求处理).具体的请求是由应用来进行处理的

创建应用app

- 创建应用的指令
    - python3 manage.py startapp 应用名称
    - 如:
        - python3 manage.py startapp music

Django应用的结构组成

```
1. `migrations` 文件夹
    - 保存数据迁移的中间文件
2. `__init__.py`
    - 应用子包的初始化文件
3. `admin.py`
    - 应用的后台管理配置文件
4. `apps.py`
    - 应用的属性配置文件
5. `models.py`
    - 与数据库相关的模型映射类文件
6. `tests.py`
    - 应用的单元测试文件
7. `views.py`
    - 定义视图处理函数的文件
```

- 配置安装应用
    - 在 settings.py 中配置应用, 让此应用能和整个项目融为一体

```
# file : settings.py
INSTALLED_APPS = [
    ... ...,
    '自定义应用名称'
]
```

    - 如:

```
INSTALLED_APPS = [
    # ....
    'user',  # 用户信息模块
    'music',  # 收藏模块
]
```

- 应用的分布式路由
    - 使用include 函数让某个正则匹配后关联分支到某个app

```
# file : <项目名>/urls.py
from django.conf.urls import include

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^music/', include('music.urls')),
    url(r'^sport/',include('sport.urls')),
    url(r'^news/',include('news.urls')),
]
```

```
# file : <App名>/urls.py
from django.conf.urls import url
from . import views

urlpatterns = [
    # 购物车模块用到的路由
    url(r'^page1', views.page1),
    url(r'^page2', views.page2),
    url(r'^page3', views.page3),
    # ...
]
```

- 练习:

```
1.创建三个应用
    1.创建 music 应用,并注册
    2.创建 sport 应用,并注册
    3.创建 news  应用,并注册
2.创建分布式路由系统
    主路由配置只做分发
    每个应用中处理具体访问路径和视图
    1. 127.0.0.1:8000/music/index
        交给 music 应用中的 index() 函数处理
    2. 127.0.0.1:8000/sport/index
        交给 sport 应用中的 index() 函数处理
    3. 127.0.0.1:8000/news/index
        交给 news  应用中的 index() 处理处理
```