《Django 教程》

- 讲师: 魏明择
- 时间: 2019

目录

cookies 和 session(会话)

cookies

- cookies是保存在客户端浏览器上的存储空间，通常用来记录浏览器端自己的信息和当前连接的确认信息
- cookies 在浏览器上是以键-值对的形式进行存储的，键和值都是以ASCII字符串的形存储(不能是中文字符串)
- cookies 的内部的数据会在每次访问此网址时都会携带到服务器端，如果cookies过大会降低响应速度
- 在Django 服务器端来设置 设置浏览器的COOKIE 必须通过 HttpResponse 对象来完成
- HttpResponse 关于COOKIE的方法
  - 添加、修改COOKIE
    - HttpResponse.set_cookie(key, value='', max_age=None, expires=None)
      - key:cookie的名字
      - value:cookie的值
      - max_age:cookie存活时间，秒为单位
      - expires:具体过期时间

```
- 删除COOKIE
    - HttpResponse.delete_cookie(key)
    - 删除指定的key 的Cookie。 如果key 不存在则什么也不发生。
```

- Django中的cookies
  - 使用 响应对象HttpResponse 等 将cookie保存进客户端
    1. 方法1

```
from django.http import HttpResponse
resp = HttpResponse()
resp.set_cookie('cookies名', cookies值, 超期时间)
```

- 如:

```
resp = HttpResponse()
resp.set_cookie('myvar', "weimz", 超期时间)
```

2. 方法二, 使用render对象

```
from django.shortcuts import render
resp = render(request,'xxx.html',locals())
resp.set_cookie('cookies名', cookies值, 超期时间)
```

3. 方法三, 使用redirect对象

```
from django.shortcuts import redirect
resp = redirect('/')
resp.set_cookie('cookies名', cookies值, 超期时间)
```

3. 获取cookie
   - 通过 request.COOKIES 绑定的字典(dict) 获取客户端的 COOKIES数据

```
value = request.COOKIES.get('cookies名', '没有值!')
print("cookies名 = ", value)
```

4. 注:
   - Chrome 浏览器 可能通过开发者工具的 `Application` >> `Storage` >> `Cookies` 查看和操作浏览器端所有的 Cookies 值

- cookies 示例
  - 以下示例均在视图函数中调用
  - 添加cookie

```
# 为浏览器添加键为 my_var1,值为123, 过期时间为1个小时的cookie
responds = HttpResponse("已添加 my_var1,值为123")
responds.set_cookie('my_var1', 123, 3600)
return responds
```

  - 修改cookie

```
# 为浏览器添加键为 my_var1,修改值为456, 过期时间为2个小时的cookie
responds = HttpResponse("已修改 my_var1,值为456")
responds.set_cookie('my_var1', 456, 3600*2)
return responds
```

- 删除cookie

```
# 删除浏览器键为 my_var1的cookie
responds = HttpResponse("已删除 my_var1")
responds.delete_cookie('my_var1')
return responds
```

- 获取cookie

```
# 获取浏览器中 my_var变量对应的值
value = request.COOKIES.get('my_var1', '没有值!')
print("cookie my_var1 = ", value)
return HttpResponse("my_var1:" + value)
```

session 会话控制

- 什么是session
  - session又名会话控制，是在服务器上开辟一段空间用于保留浏览器和服务器交互时的重要数据
- session的起源
  - 客户端与服务器端的一次通信，就是一次会话
  - http协议是无状态的：每次请求都是一次新的请求，不会记得之前通信的状态
  - 实现状态保持的方式：在客户端或服务器端存储与会话有关的数据
  - 推荐使用sesison方式，所有数据存储在服务器端
- 实现方式
  - 使用session 需要在浏览器客户端启动 cookie，且用在cookie中存储sessionid
  - 每个客户端都可以在服务器端有一个独立的Session
  - 注意：不同的请求者之间不会共享这个数据，与请求者一一对应
- Django启用Session
  - 在 settings.py 文件中
  - 向 INSTALLED_APPS 列表中添加：

```
INSTALLED_APPS = [
    # 启用 sessions 应用
    'django.contrib.sessions',
]
```

  - 向 MIDDLEWARE_CLASSES 列表中添加：

```
MIDDLEWARE = [
    # 启用 Session 中间件
    'django.contrib.sessions.middleware.SessionMiddleware',
]
```

- session的基本操作:
  - session对于象是一个在似于字典的SessionStore类型的对象, 可以用类拟于字典的方式进行操作
  - session 只能够存储能够序列化的数据,如字典，列表等。
  - 保存 session 的值到服务器
    - `request.session[键] = 值`
    - 如: `request.session['KEY'] = VALUE`
  - 获取session的值
    - `VALUE = request.session['KEY']`
    - 或
    - `VALUE = request.session.get('KEY', 缺省值)`
  - 删除session的值
    - `del request.session['KEY']`
  - 在 settings.py 中有关 session 的设置
    1. SESSION_COOKIE_AGE
       - 作用: 指定sessionid在cookies中的保存时长(默认是2周)，如下:
       - `SESSION_COOKIE_AGE = 60 * 60 * 24 * 7 * 2`
    2. SESSION_EXPIRE_AT_BROWSER_CLOSE = True 设置只要浏览器关闭时,session就失效(默认为False)
  - session 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`
- 注: 当使用session时需要迁移数据库,否则会出现错误

```
$ python3 manage.py makemigrations
$ python3 manage.py migrate
```

- session 示例
  - 以下示例均在视图函数中调用
  - 添加sesson

```
# 在当前请求所对应的session中添加键为mysession_var, 值为 100的键值对
request.session['mysession_var'] = 100
responds = HttpResponse("添加session")
return responds
```

- 修改session键对应的值

```
# 修改当前请求所对应的session中键为 mysession_var, 值为 200
request.session['mysession_var'] = new_value
responds = HttpResponse("修改session成功")
return responds
```

- 删除session 键为mysession_var 的键值对

```
try:
    del request.session['mysession_var']
    responds = HttpResponse("删除session成功")
except:
    responds = HttpResponse("删除session失败")
return responds
```

- 查看session 中 mysession_var, 对应的值

```
mysession_var = request.session.get('mysession_var', None)
# 或:
if 'mysession_var' in request.session:
    mysession_var = request.session['mysession_var']
else:
    mysession_var = None

print("mysession_var = ", mysession_var)
return HttpResponse("mysession_var = " + str(mysession_var))
```

权限管理

- 权限管理是指由于业务逻辑需要，根据当前登陆的用户信息来决定视图处理函数能否有权限执行相应操作
- 练习:
  - 为图书的添加，删除和修改添加权限管理。
  - 让没有登陆的用户不允许添加图书等操作
  - 没有权限时抛出 Http404异常进入404页面(raise Http404)

- 练习:

    1. 创建一个userinfo 应用 python3 manage.py startapp userinfo
    2. 创建一个页面 路由: /userinfo/login 视图函数 def mylogin(request): .. 模版: login.html
    3. 写一个注册界面 路由: /userinfo/reg 视图函数: def myregister(request): .. 模板: templates/userinfo/register.html
    4. 修改登陆的逻辑
    5. 退出登陆 路由: /userinfo/logout 视图函数: def mylogout(request):
    6. 添加主页 路由: '/' 视图函数: 主视图views.py def homepage(request): 模板为: homepage.html

- 练习: 实现网云笔记

    - 功能: 注册 登陆 查看以前自己的笔记 自己写一个新的笔记 修改以前的笔记 删除笔记
    - 模型类 class User(models.Model): ... class Note(models.Model)： title = CharField(...) # 标题 content = CharField(...) # 内容 user = models.ForeignKey(User) # 关联用户外键 create_time = models.DateField(...) # 创建时间 mod_time = models.DateField(...) # 修改时间
    - 参考:
        - 登陆界面