cookie和session

HTTP被设计为"无态"，也就是俗称"脸盲"。 这一次请求和下一次请求 之间没有任何状态保持，我们无法根据请求的任何方面(IP地址，用户代理等)来识别来自同一人的连续请求。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（客户端与服务器端的一次通信，就是一次会话）

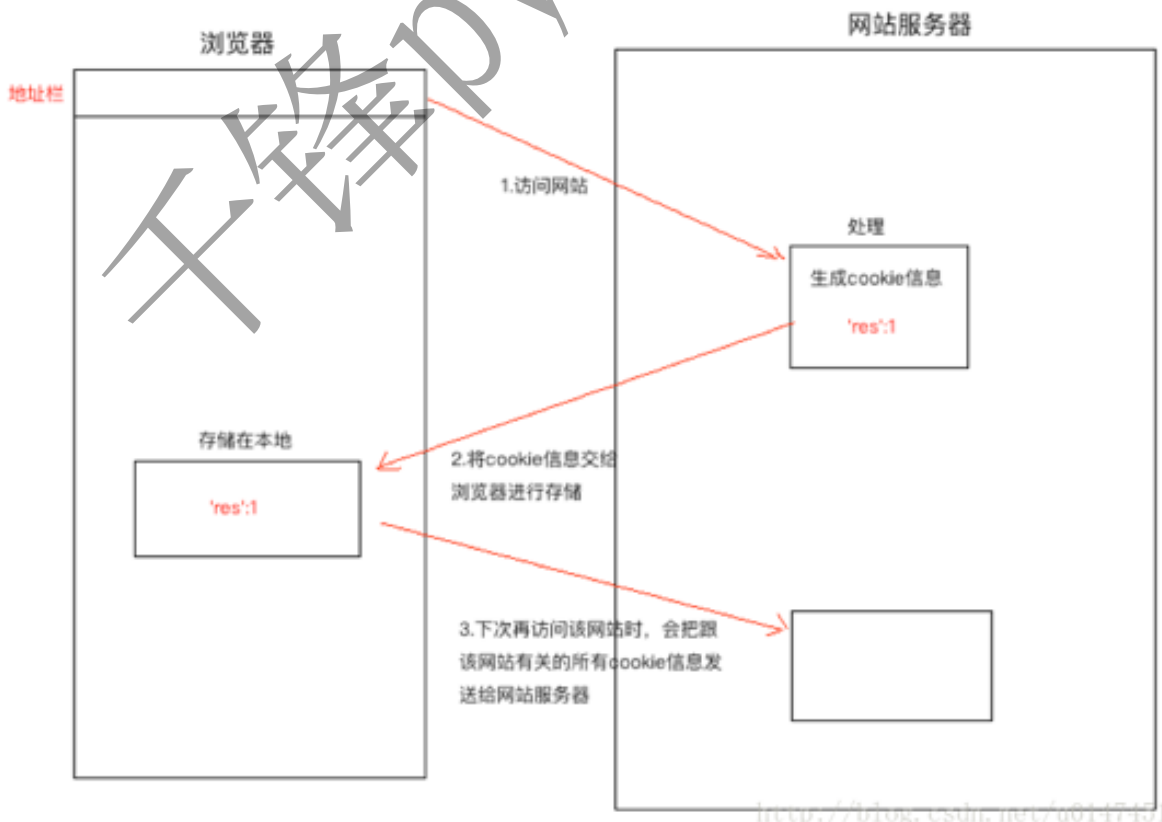
- cookie
- session

不同的请求者之间不会共享这些数据，cookie和session与请求者一一对应。

一、cookie

cookies 是浏览器为 Web 服务器存的一小信息。 每次浏览器从某个服务器请求页面 时，都会自动带上以前收到的cookie。cookie保存在客户端，安全性较差，注意不要保存敏感信息。典型应用：

- 网站登录
- 购物车



用法：

```
# 1.设置cookie
HttpResponse.set_cookie(key, value='', max_age=None, expires=None,
path='/', domain=None, secure=None, httponly=False)
参数：
    key：     cookie的名称(*)
    value:   cookie的值,默认是空字符
    max_age：cookies的持续有效时间（以秒计），如果设置为 None，cookies 在
浏览器关闭的时候就失效了。
    expires：cookies的过期时间，格式:"Wdy, DD-Mth-YY HH:MM:SS GMT" 如果
设置这个参数，它将覆盖max_age。
    path:    cookie生效的路径前缀，浏览器只会把cookie回传给带有该路径的页
面，这样你可以避免将cookie传给
             站点中的其他的应用。/ 表示根路径，特殊的：根路径的cookie可以
被任何url的页面访问
    domain:   cookie生效的站点。你可用这个参数来构造一个跨站cookie。如，
domain=".example.com" 所构造的
              cookie对下面这些站点都是可 读的： www.example.com 、
www2.example.com。
              如果该参数设置为None，cookie只能由设置它的站点读取。
    secure:   如果设置为 True ，浏览器将通过HTTPS来回传cookie。
    httponly: 仅http传输    不能使用js获取cookie

#同set_cookie,不同点在于设置salt，即加盐，加密存储cookie数据
HttpResponse.set_signed_cookie(key, value, salt='', max_age=None,
expires=None, path='/', domain=None, secure=None, httponly=False)
```

```
#2 获取cookie
HttpRequest.COOKIES.get(key)

#获取加“盐”的cookie
HttpRequest.get_signed_cookie(key, default=RAISE_ERROR, salt='',
max_age=None)
```

```
# 3删除cookie
HttpResponse.delete_cookie(key, path='/', domain=None)
```

二、session

cookie看似解决了HTTP（短连接、无状态）的会话保持问题，但把全部用户数据保存在客户端，存在安全隐患，

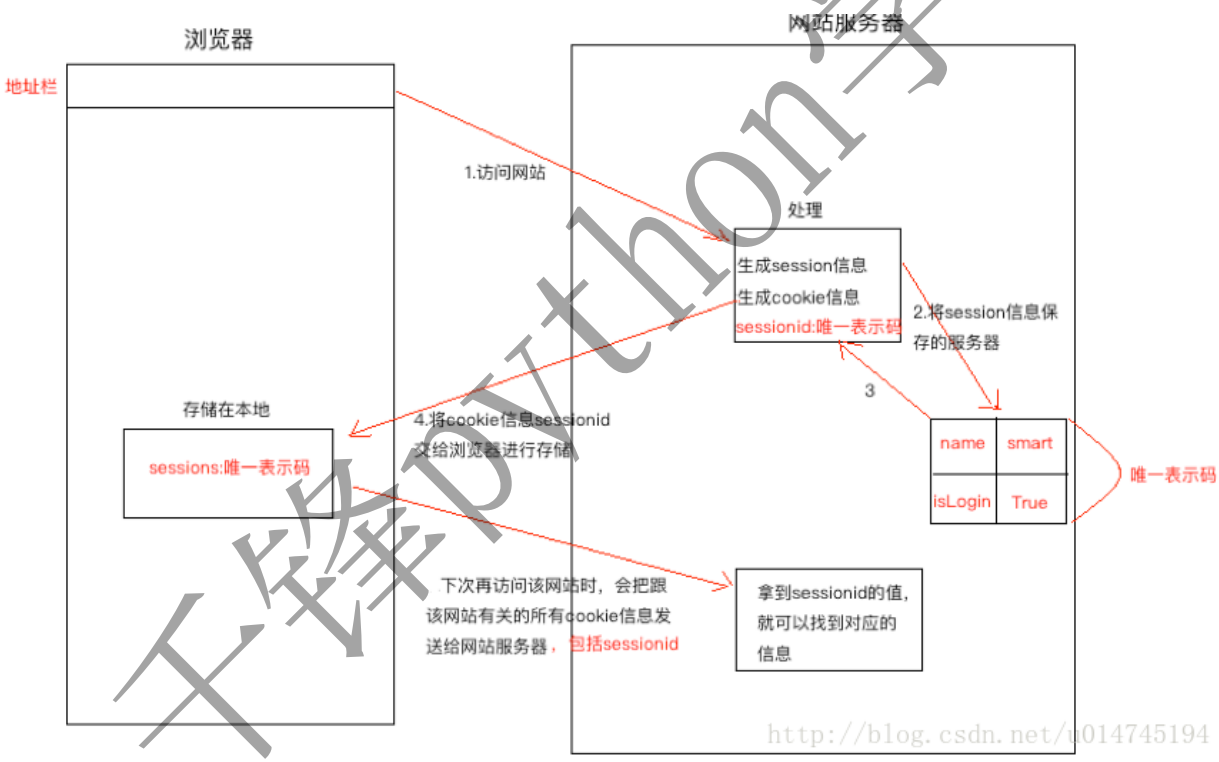
于是session出现了。我们可以 把关于用户的数据保存在服务端，在客户端cookie里加一个sessionID（随机字符串）。其**工作流程**：

(1)、当用户来访问服务端时,服务端会生成一个随机字符串；

(2)、当用户登录成功后 把 {sessionID :随机字符串} 组织成键值对加到cookie里发送给用户；

(3)、服务器以发送给客户端 cookie中的随机字符串做键，用户信息做值，保存用户信息；

(4)、再访问服务时客户端会带上sessionid，服务器根据sessionid来确认用户是否访问过网站



2.1 cookie和session的区别与联系

- 区别
  - session将数据存储与服务器端 cookie存储在客户端
  - cookie 存储在客户端，不安全，sess存储在服务器端，客户端只存sesseionid,安全
  - cookie在客户端存储值有大小的限制，大约几kb。session没有限制
- 联系

- session 基于cookie

2.2 session配置

1. **首先在settings.py中有如下配置（系统默认），**

```
INSTALLED_APPS = [
    'django.contrib.sessions',
]
MIDDLEWARE = [
    'django.contrib.sessions.middleware.SessionMiddleware',
]
```

2. 进行数据迁移，生成session使用的数据库表

2.3 session操作

- session设置

```
def doregister(request):
    username = request.POST.get('username')
    password = request.POST.get('password')
    email = request.POST.get('email')
    user = User()
    user.username = username
    user.password = md5(password.encode('utf8')).hexdigest()
    user.email = email
    user.save()
    # 设置session
    request.session['username'] = username
    return render(request,"common/notice.html",context={
        'code':1,
        'msg':'注册成功',
        'url':'three:index',
        'wait':3
    })
```

- session获取

```
def index(request):
    # session获取
    username = request.session.get('username')
    return render(request,'three/index.html',context=
{'username':username})
```

- session删除
  - clear() 清空所有session 但是不会将session表中的数据删除
  - flush() 清空所有 并删除表中的数据
  - logout() 退出登录 清除所有 并删除表中的数据
  - del req.session['key'] 删除某一个session的值

```
def logout(request):
    request.session.flush()
    return redirect(reverse("three:index"))
```

- session过期时间

  req.session.set_expiry(5)

三、短信验证码

APP、网站注册账号，向手机下发验证码； 登录账户、异地登录时的安全提醒；找回密码时的安全验证； 支付认证、身份校验、手机绑定等。本例采用阿里云短信验证。

- 安装

  如果您使用Python 3.x，执行以下命令，安装阿里云SDK核心库：

  ```
  pip install aliyun-python-sdk-core-v3
  ```

- 发送步骤：
  - 1.创建Client实例。在创建Client实例时，您需要获取Region ID、AccessKey ID和AccessKey Secret。
  - 2.创建API请求并设置参数。
  - 3.发起请求并处理应答或异常。

```
from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.request import CommonRequest
from day06.settings import SMS_CONFIG

def send_sms(phone,templateParam):
    client = AcsClient(SMS_CONFIG['ACCESS_KEY_ID'],
SMS_CONFIG['ACCESS_KEY_SECRET'], 'default')
    request = CommonRequest()
    request.set_accept_format('json')
    request.set_domain('dysmsapi.aliyuncs.com')
    request.set_method('POST')
    request.set_protocol_type('https') # https | http
    request.set_version('2017-05-25')
    request.set_action_name('SendSms')

    request.add_query_param('PhoneNumbers', phone)
    request.add_query_param('SignName', SMS_CONFIG['SignName'])
    request.add_query_param('TemplateCode',
SMS_CONFIG['TemplateCode'])
    request.add_query_param('TemplateParam', templateParam)
    response = client.do_action_with_exception(request)
    return str(response, encoding = 'utf-8')
if __name__ == "__main__":
    send_sms('15116905290',{'number':'390983'})
```

四、分页

4.1 Paginator 分页器

Paginator用于分页，但Paginator并不具体管理具体的页的处理，而是使用Page对象管理具体页面

- 创建分页器对象

格式： Paginator(<query_set查询集>,每页显示数据的条数)

- 对象的属性

  count 分页对象的个数

  num_pages 总页数

  page_range 页码的列表

- 方法

  page(num) 返回page对象 如果给定的页码不存在 则抛出异常

4.2 page 对象

page对象具体负责每页的处理，包括每页的数据，当前页的页码，是否有上一页或下一页等。

类别	名称	说明
属性	object_list	当前页码上的所有数据
属性	number	当前页码值
属性	paginator	返回Paginator的对象
方法	has_next	是否有下一页
方法	has_previous	是否有上一页
方法	has_other_pages	是否有上一页 或者下一页
方法	next_page_number	返回下一页的页码
方法	previous_page_number	返回上一页的页码
方法	len	返回当前页数据的个数

- 实例

```
#路由
url(r'^userlist/$',views.userlist,name='userlist'),
url(r'^userlist/(\d+)/$',views.userlist,name='userlist1'),
```

```
# views.py
def userlist(request,page=1):
    users = User.objects.all()
    # 实例化分页对象，一页两条记录
    pagination = Paginator(users,2)
    page = pagination.page(page)  #某一页的分页对象

    return render(request,'userlist.html',context={
        'data':page.object_list,  #当前页的数据(列表)
        'page_range':pagination.page_range,#页码范围
        'page':page
    })
```

```
#模板文件
{% load static %}
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -
->
    <title>用户列表</title>
    <!-- Bootstrap -->
    <link href="{% static 'bootstrap/css/bootstrap.min.css' %}"
rel="stylesheet">
  </head>
  <body>
    <div class="container-fluid">
        <div class="row">
            <div class="bs-example" data-example-id="bordered-table">
                <table class="table table-bordered">
                    <thead>
                    <tr>
                        <th>#</th>
                        <th>用户名</th>
                        <th>密码</th>
                        <th>年龄</th>
                    </tr>
                    </thead>
```

```
                    <tbody>
                    {% for user in data %}
                        <tr>
                            <th scope="row">{{ forloop.counter }}
</th>
                            <td>{{ user.username }}</td>
                            <td>{{ user.password }}</td>
                            <td>{{ user.age }}</td>
                        </tr>
                    {% endfor %}



                    </tbody>
                </table>
            </div>
        </div>
    </div>
    <div class="container-fluid">
        <div class="row">
            <div class="col-md-4"></div>
            <div class="col-md-5">
                <nav aria-label="Page navigation">
                    <ul class="pagination">
                        {% if page.has_previous %}
                            <li>
                                <a href="{% url 'app:userlist1'
page.number|add:'-1' %}" aria-label="Previous">
                                    <span aria-hidden="true">&laquo;
</span>
                                </a>
                            </li>
                        {% else %}
                             <li class="disabled">
                            <span href="{% url 'app:userlist1'
page.number|add:'-1' %}" aria-label="Previous">
                                <span aria-hidden="true">&laquo;
</span>
                            </span>
                            </li>

                        {% endif %}
```

```
                        {% for num in pagerange %}
                            {% ifequal num page.number %}
                                <li class="active"><a href="{% url
'app:userlist1' num %}">{{ num }}</a></li>
                            {% else %}
                                <li><a href="{% url 'app:userlist1'
num %}">{{ num }}</a></li>
                            {% endifequal %}

                        {% endfor %}
                        {% if page.has_next %}
                            <li>
                                <a href="{% url 'app:userlist1'
page.number|add:'1' %}" aria-label="Next">
                                    <span aria-hidden="true">&raquo;
</span>
                                </a>
                            </li>
                        {% else %}
                             <li class="disabled">
                                <span href="#" aria-label="Next">
                                    <span aria-hidden="true">&raquo;
</span>
                                </span>
                            </li>
                        {% endif %}
                    </ul>
                </nav>

            </div>
            <div class="col-md-3"></div>
        </div>
    </div>
    <!-- jQuery (Bootstrap 的所有 JavaScript 插件都依赖 jQuery，所以必
须放在前边) -->
    <script src="{% static 'bootstrap/js/jquery-1.12.4.min.js' %}">
</script>
    <!-- 加载 Bootstrap 的所有 JavaScript 插件。你也可以根据需要只加载单
个插件。 -->
    <script src="{% static 'bootstrap/js/bootstrap.min.js' %}">
</script>
  </body>
```

```
</html>
```