08 Django会话技术 Cookie & Session

Cookie 和 Session

Cookie

理论上，一个用户的所有请求操作都应该属于同一个会话，而另一个用户的所有请求操作则应该属于另一个会话，二者不能混淆。而Web应用程序是使用HTTP协议传输数据的。HTTP协议是无状态的协议。一旦数据交换完毕，客户端与服务器端的连接就会关闭，再次交换数据需要建立新的连接。这就意味着服务器无法从连接上跟踪会话。要跟踪该会话，必须引入一种机制。

Cookie就是这样的一种机制。它可以弥补HTTP协议无状态的不足。在Session出现之前，基本上所有的网站都采用Cookie来跟踪会话。

Cookie实际上是一小段的文本信息。客户端请求服务器，如果服务器需要记录该用户状态，就使用response向客户端浏览器颁发一个Cookie。客户端浏览器会把Cookie保存起来。当浏览器再请求该网站时，浏览器把请求的网址连同该Cookie一同提交给服务器。服务器检查该Cookie，以此来辨认用户状态。服务器还可以根据需要修改Cookie的内容。

由于HTTP是一种无状态的协议，服务器单从网络连接上无从知道客户身份。怎么办呢？就给客户端们颁发一个通行证吧，每人一个，无论谁访问都必须携带自己通行证。这样服务器就能从通行证上确认客户身份了。这就是Cookie的工作原理。

cookie本身由服务器生成，通过Response将cookie写到浏览器上，下一次访问，浏览器会根据不同的规则携带cookie过来。

注意：cookie不能跨浏览器，一般不跨域

设置cookie（使用response设置）：

```
response.set_cookie(key,value[,max_age=None,expires=None])
  max_age：整数 单位为秒，指定cookie过期时间
                  设置为None:浏览器关闭失效,默认值
  expires：指定过期时间，还支持datetime或timedelta，可以指定一个具体日期时间
              expires=datetime.datetime(2030, 1, 1, 2, 3, 4)
              或 datetime.datetime.now() + datetime.timedelta(days=10)

        注意:max_age和expries两个选一个指定
  # response.set_cookie('username', username, max_age=10)
  # response.set_cookie("username", username1, expires=d)
```

获取cookie(使用request获取)：

```
  request.COOKIES.get('username')
```

删除cookie(使用response删除)：

```
  response.delete_cookie('username')
```

cookie存储到客户端

优点：

数据存在在客户端，减轻服务器端的压力，提高网站的性能。

缺点：

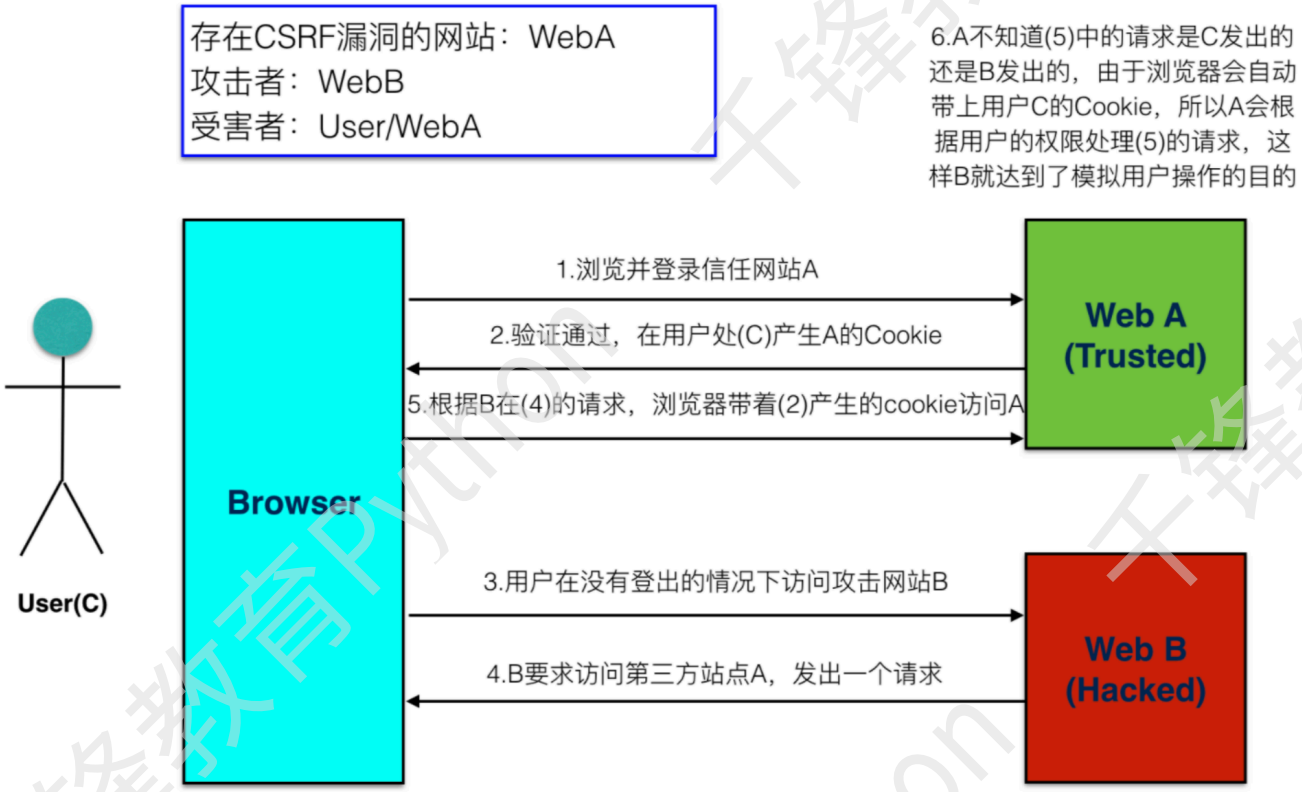
安全性不高：在客户端机很容易被查看或破解用户会话信息

Session

```
服务器端会话技术,依赖于cookie.
  django中启用SESSION
    settings中
      INSTALLED_APPS:
        'django.contrib.sessions'
      MIDDLEWARE:
        'django.contrib.sessions.middleware.SessionMiddleware'
  基本操作
    设置Sessions值（使用request设置）
          request.session['user_id'] = user.id
          request.session.set_expiry(86400)  # 设置过期时间

    获取Sessions值
        get(key,default=None) 根据键获取会话的值
          username = request.session.get("user_id")
          # 或 session_name = request.session["session_name"]

    删除Sessions值
        # 获取当前请求的session的key
        session_key = request.session.session_key
          del request.session[session_key]
          # request.session.delete(session_key)
          flush() 删除当前的会话数据并删除会话的cookie

    clear() 清除所有会话

    数据存储到数据库中会进行编码,使用的是Base64
  每个HttpRequest对象都有一个session属性，也是一个类字典对象.
```

CSRF

- CSRF全拼为Cross Site Request Forgery，跨站请求伪造。
- CSRF指攻击者盗用了你的身份，以你的名义发送恶意请求
  - 包括：以你名义发送邮件，发消息，盗取你的账号，甚至于购买商品，虚拟货币转账...
- 造成的问题：个人隐私泄露以及财产安全。



防止CSRF

- Django下的CSRF预防机制
  django 第一次响应来自某个客户端的请求时，会在服务器端随机生成一个 token，把这个 token 放在 cookie 里。然后每次 POST 请求都会带上这个 token，这样就能避免被 CSRF 攻击。
- 在Post请求时，表单中添加 {% csrf_token %}

```
<form action="" method="post">
    {% csrf_token %}
    ...
</form>
```