

session和cookies会话机制详解



六尺帐篷 (/u/f8e9b1c246f1)

2016.07.20 11:40 字数 1979 阅读 2529 评论 10 喜欢 53 赞赏 1

(/u/f8e9b1c246f1)

编辑文章 (/writer#/notebooks/5150869/notes/4868276)

session management会话管理的原理

web请求与响应基于http，而http是无状态协议。所以我们为了跨越多个请求保留用户的状态，需要利用某种工具帮助我们记录与识别每一次请求及请求的其他信息。举个例子，我们在淘宝购物的时候，首先添加了一本《C++ primer》进入购物车，然后我们又继续去搜索《thinking in java》，继续添加购物车，这时购物车应该有两本书。但如果我们不采取session management会话管理的话，基于http无状态协议，我们在第二次向购物车发出添加请求时，他是无法知道我们第一次添加请求的信息的。所以，我们就需要session management会话管理！

会话管理的基本方式

会话管理的基本主要有隐藏域，cookies，与URL重写这几种实现方式。用得较多的是后两种。

隐藏域实现会话管理

以一个网络注册信息填写为例。

我们在填注册信息的时候，经常遇到填完一个页面的内容之后，还要继续填写下一个页面的内容。但由于http的无状态，那么容易造成的后果，当进入第二页填写的时候，服务器已经不记得我们上一页填写了什么。

怎么利用隐藏域解决这个问题呢？

顾名思义，其实就是既然服务器不会记得两次请求间的关系，那就由浏览器在每次请求时主动告诉服务器多次请求间的必要信息，但是上一页的信息并不显示在第二页中，而是采用隐藏域的方式。

然而显然这种方式是存在各种问题的。

比如关掉网页之后，就会遗失信息，而且查看网页源代码时，容易暴露信息，安全性不高。隐藏域并不是servlet/jsp实际会话管理的机制。

cookie实现会话管理

cookie是什么？举个简单的例子，现在当我们浏览网站的时候，经常会自动保存账号与密码，这样下次访问的时候，就可以直接登录了。这种技术的实现就是利用了cookie技术。cookie是存储key-value对的一个文件，务必记住，它是由服务器将cookie添加到response里一并返回给客户端，然后客户端会自动把response里的cookie接收下来，并且保存到本地，下次发出请求的时候，就会把cookie附加在request里，服务器在根据request里的cookie遍历搜索是否有与之符合的信息

具体cookie的实现我们会在后面详细讲到

URL重写实现会话管理

URL重写就是将需要记录的信息附加在请求的链接背后，以链接参数的形式发送给服务器识别。具体实现的过程会在后文结合cookie详解。



Servlet&JSP中的Session会话管理机制

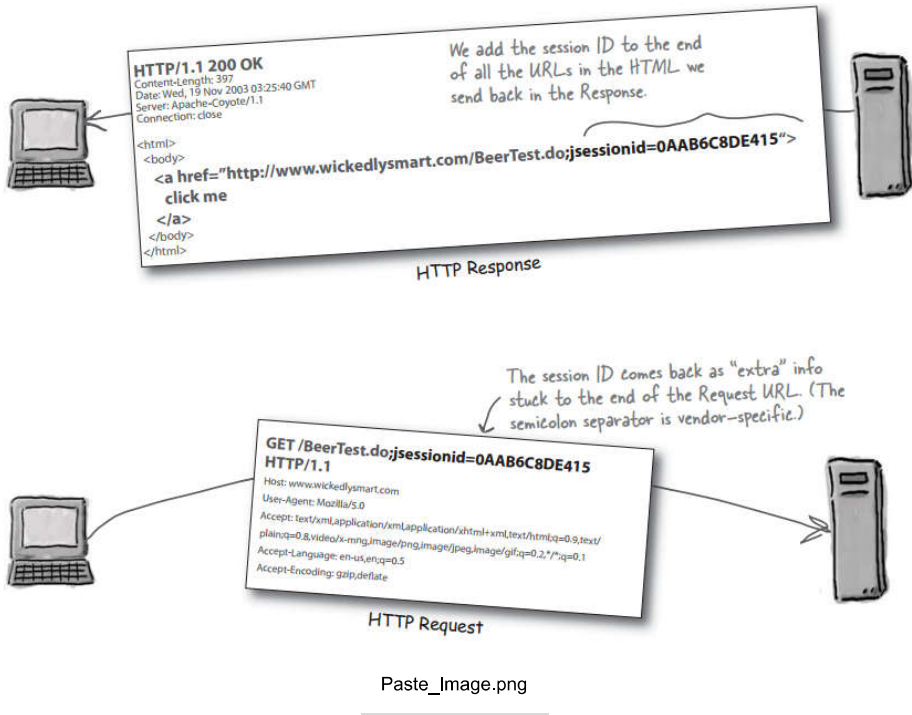
利用HttpSession对象进行会话管理。HttpSession对象可以保存跨同一个客户多个请求的会话状态。

换句话说，与一个特定客户的整个会话期间看，HttpSession会持久储存。

对于会话期间客户做的所有请求，从中得到的所有信息都可以用HttpSession对象保存。

HttpSession的工作机制

- 以之前的问卷调查为例，当一个新客户小明填写问卷时，服务器会生成一个HttpSession对象，用于保存会话期间小明所选择的信息，服务器会以setAttribute的方式将其保存到HttpSession对象中。
每个客户会有一个独立的HttpSession对象，保存这个客户所有请求所需要保存的信息。
- 服务器如何识别所有的请求是否来自同一个客户？
客户需要一个会话ID来标识自己。就跟我们每个人的身份证号一样。对于客户的第一个请求，容器会生成一个唯一的会话ID，并通过相应把它返回给用户，客户在以后发回一个请求中发回这个会话ID，容器看到ID之后，就会找到匹配的会话，并把这个会话与请求关联。
- 实现存储会话ID的就是通过cookie！



cookie存储在客户端，是被服务器放在response里发回客户端的，以后每次request时，都会把cookie加入到request里。
而session是存在服务器的，以属性的形式将会话中的信息存到HttpSession对象中。调用时，只要通过HttpSession对象调用相应attribute即可。

- 很多地方总是把session与cookie分开单独讲。但我们通过前面的介绍，不难知道，session实现其会话管理机制时，在如何确定所有请求是否来自同一个客户时，是利用了cookie技术的。所以不应该将cookie与session完全分开讲。
- 这里产生这个误解的原因。是因为我们对session的会话管理机制不够了解。因为容器在创建session对象时，会帮我们实现所有cookie相关的工作，而我们只需要实现这一句：

```
HttpSession session = request.getSession();
```

记住: ** 这个方法不只是创建一个会话, 而是会完成所有与cookie相关的工作, 只是容器都自动帮我们实现了。我们来看看容器在背后默默为我们做了什么:

- 建立新的httpsession对象
- 生成唯一的会话ID
- 建立新的会话对象
- 把会话ID与cookie关联
- 在响应中设置cookie

cookie所有的工作都在后台进行。

看到这里, 是不是很爽? 容器几乎帮我们实现了所有cookie工作。

- 从请求中得到会话ID

只需一行代码:

```
HttpSession session = request.getSession();
```

与上一部分为响应生成会话ID是一致的

其中也在后台实现了一些步骤:

if (请求包含一个会话ID)

找到与该ID匹配的会话

else if (没有会话ID或者没有匹配的ID)

创建一个新的会话。

还是那句话: **cookie所有工作都在后台自动进行**

cookie的更多用处

cookie原先设计的初衷就是为了帮助支持会话状态。但是因为cookie的简便性, 容器为我们封装了大量操作。现在cookie已经被越来越运用到各个方面。

首先, **我们明确cookie是存在客户端的, 实际上就是在客户端与服务端交换的一小段数据 (一个name/string对) 。**

由于session在用户关闭浏览器后, 会话结束, 就会消失, cookie随之应该也会消失。但servlet的API中提供了一些方法, 可以让客户端的cookie存活的时间更久一点。这就是cookie相对于session的一大优势所在。我们目前常用的记住用户名和密码, 下次登录就是利用cookie在session消失后, 还能存活实现的。

所以, 我们可以定制cookie为我们实现各种功能。



六尺帐篷 (/u/f8e9b1c246f1)

写了 248002 字, 被 15247 人关注, 获得了 1422 个喜欢

(/u/f8e9b1c246f1)

如果觉得我的文章对您有用, 请随意赞赏。您的支持将鼓励我继续创作!

赞赏支持



(/u/2dae36af8afd)

 喜欢

53











更多分享

(http://cwb.assets.jianshu.io/notes/images/21868276)










写下你的评论...

10条评论

只看作者 关闭评论

按喜欢排序 按时间正序 按时间倒序



杨万 (/u/464bd864c118)
7楼 · 2017.04.17 10:53
(/u/464bd864c118)
楼主你好
在文章最后阶段，楼主提到（由于session在用户关闭浏览器后，会话结束，就会消失），这个表达的是不对的。
通常cookie与session交互通过sessionId，浏览器关闭后清除内存cookie与相应的sessionId（保存在内存cookie中的），而session是保存服务器中（通常会设置时效）如果超出时效没有任何操作才会被web服务器释放

👍 1人赞


💬 回复



moreFine (/u/6aed5a589fee)
2楼 · 2016.07.21 09:07
(/u/6aed5a589fee)
OK

👍 赞

💬 回复



橙小麦 (/u/4a487b595908)
3楼 · 2016.07.21 15:36
(/u/4a487b595908)
看完之后，发现，没看懂。。

👍 赞

💬 回复

橙小麦 (/u/4a487b595908): @DJM209 (/users/4a487b595908) 🙄
2016.07.21 15:36 💬 回复

六尺帐篷 (/u/f8e9b1c246f1): @DJM209 (/users/4a487b595908) 🙄 我没讲清楚
2016.07.21 15:41 💬 回复

橙小麦 (/u/4a487b595908): @六尺帐篷 (/users/f8e9b1c246f1) 哈哈，你讲的很清楚，只是我不懂技术。。。🙄
2016.07.22 09:47 💬 回复


✎ 添加新评论



六尺帐篷 (/u/f8e9b1c246f1) 作者
4楼 · 2016.07.21 15:40
(/u/f8e9b1c246f1)
🙄 我没讲清楚

👍 赞

💬 回复



史志华 (/u/43ca6ceb7f53)
5楼 · 2016.07.21 20:56
(/u/43ca6ceb7f53)
介绍的很清楚，点赞

👍 赞

💬 回复

六尺帐篷 (/u/f8e9b1c246f1): @史志华 (/users/43ca6ceb7f53) 谢谢~多交流
2016.07.21 21:22 💬 回复


✎ 添加新评论

⬆

🔖

📖

🔗



捡淑 (/u/5b8db0f1c979)

6楼 · 2016.07.25 14:57

(/u/5b8db0f1c979)

mark

 赞  回复

被以下专题收入，发现更多相似内容

 投稿管理

- + 收入我的专题

 dotNET (/c/c6c7e127a22b?utm_source=desktop&utm_medium=notes-included-collection)

 编程的小秘密 (/c/549d722a0a5b?utm_source=desktop&utm_medium=notes-included-collection)

 前端那些事儿 (/c/003aa902e02a?utm_source=desktop&utm_medium=notes-included-collection)

 程序 (/c/d29fb6cb2700?utm_source=desktop&utm_medium=notes-included-collection)

 JAVA (/c/dbce139175b0?utm_source=desktop&utm_medium=notes-included-collection)

 JAVAwab (/c/a9f6c1d599c9?utm_source=desktop&utm_medium=notes-included-collection)

 PHP (/c/de53a9b87531?utm_source=desktop&utm_medium=notes-included-collection)

展开更多 