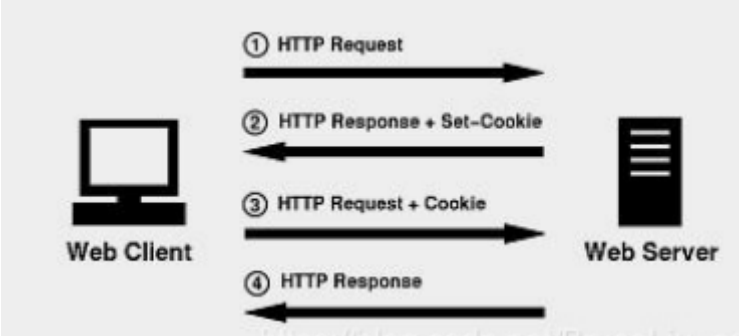
# Session是什么？

**Session是一种将会话数据保存到服务器端的技术！**

**生活场景：**在现实生活中，当人们去医院就诊时，医院会给病人办理就诊卡，卡上只有卡号， 而没有其它信息，其它信息都保存在医院的系统中。病人每次去该医院就诊时，只要出示就诊卡，医务人员便可根据卡号查询到病人的就诊信息。

Session技术就好比医院发放给病人的就诊卡和医院为每个病人保留病例档案的过程。在一次会话中， 当浏览器第一次访问Web服务器时，服务器就会为客户端创建一个Session对象和该session对象的唯一标识，其中，Session对象就相当于病历档案，标识就相当于就诊卡号，当客户端在当次会话中再次访问服务器时，只要将标识传递给服务器，服务器就能根据标识选择与之对应的Session对象为其服务。需要注意的是，由于服务器需要根据客户端传递的标识获取与标识对应的session，因此客户端需要每次传递Session对象的唯一标识，因此，通常情况下，Session是借助Cookie技术来传递标识的。




Session：表示浏览器和服务器之间一连串的访问过程！一个Session表示一次会话（一个会话中包含若干次用户的请求！）。

# Session的获取

通过HttpServletRequest对象中的getSession()方法来获取session

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

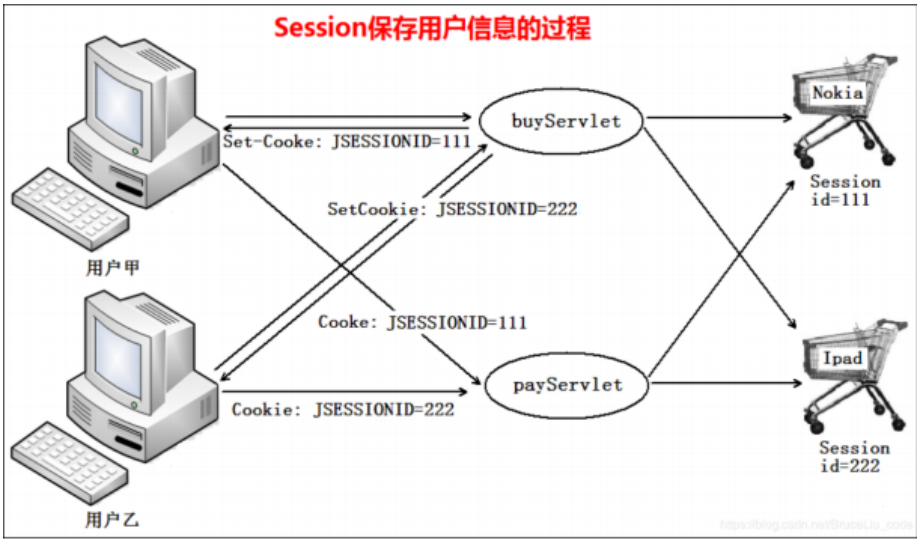
*此方法会获得专属于当前会话的**Session**对象，如果服务器端没有该会话的**Session**对象，则会创建一个新的**Session**返回，如果已经有了属于该会话的**Session**直接将已有的**Session**返回。*

该方法，还有一个重载，如果指定一个true（默认），如果不存在专属于当前会话的session，则创建；如果为false，如果不存在专属于当前会话的session时候，不会自动创建HttpSession，而是返回一个null。

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

# Session的工作原理

以网站购物为例，通过一张图来描述Session保存用户信息的原理，具体如下图所示。



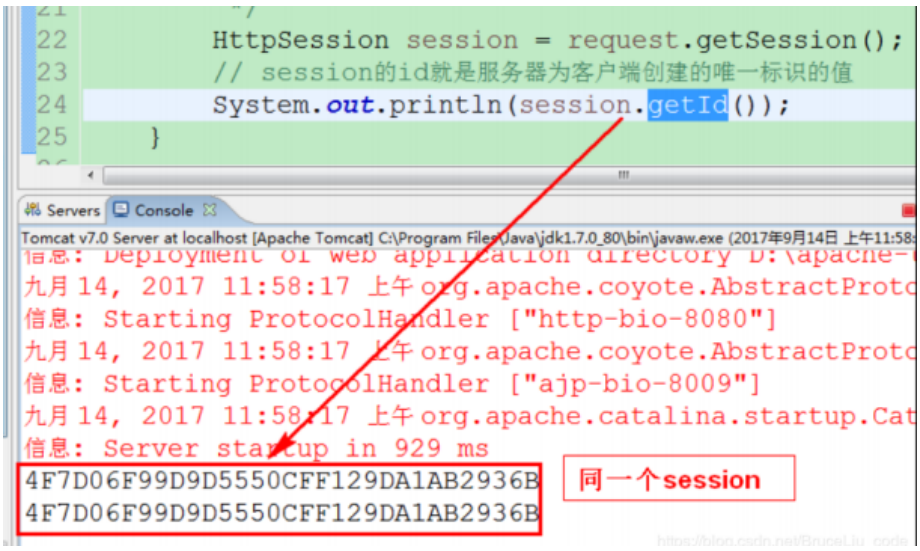

在上图中，用户甲和乙都调用buyServlet将商品添加到购物车，调用payServlet进行 商品结算。由于甲和乙购买商品的过程类似，在此，以用户甲为例进行详细说明。当用户甲访问购物网站时，服务器为甲创建了一个Session对象（相当于购物车）。当甲将Nokia手机添加到购物车时，Nokia手机的信息便存放到了用户甲的Session对象 中 。 同 时 ， 服 务 器 将 Session 对 象 的 唯 一 标 识 以 Cookie (Set-Cookie: JSESSIONID=xxxx)的形式返回给甲的浏览器。当甲完成购物进行结账时，需要向服务器发送结账请求，这时，浏览器自动在请求消息头中将Cookie (Cookie: JSESSIONID=111)信息回送给服务器，服务器根据Cookie中的JSESSIONID属性找到为 用户甲所创建的Session对象，并将Session对象中所存放的Nokia手机信息取出进行结算。

```
1  package web;
2
3  import java.io.IOException;
4  import javax.servlet.ServletException;
5  import javax.servlet.http.HttpServlet;
6  import javax.servlet.http.HttpServletRequest;
7  import javax.servlet.http.HttpServletResponse;
8  import javax.servlet.http.HttpSession;
9
10  @SuppressWarnings("all")
11  public class GetSession extends HttpServlet {
12      public void service(HttpServletRequest
request,HttpServletResponse response) throws
ServletException,IOException
        {
13          // 1. 获取session对象：服务器为当前用户创建一个专属于
他的对象来保存数据
```

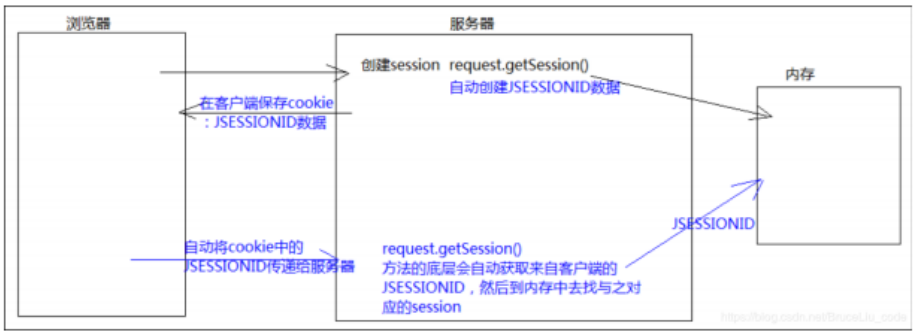
```
14          HttpSession session = request.getSession();
15          String id = session.getId();
16          System.out.println(id);
17          response.getWriter().write("session create
ok!");
18      }
19  }
```

第一次访问服务器，获取session，服务器自动创建cookie数据（JSESSIONID），作为session的唯一标识


再次访问服务器，获取session，服务器根据session的唯一标识（JSESSIONID）查找session

原理图：




> session的数据，实际上是放在服务器的内存当中的，因此session数据的大小和服务器 的内存有关！

# session的生命周期

**生命周期**：一个对象从创建到销毁的过程，称为对象的生命周期
**session的创建时间**：第一次调用request.getSession()方法的时候创建

修改session存活时间的方式：

1. **session的存活时间**：在tomcat/conf/web.xml中配置了session的存活时间，默认是30分钟，也可以在当前的项目中配置session的过期时间：

```
<!-- session的存活时间，单位是：分钟 -->
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
```

2. 使用HttpSession类中的**setMaxInactiveInterval**方法

30分钟指的是，30分钟之内，客户端没有与服务器交互(操作session)。如果29分钟都没有访问服务器获取，最后一分钟内访问了服务器获取session，那么session的时间再次回到了30分钟。

# session的失效

1. 手动调用session中的方法

```
invalidate()
        Invalidates this session then unbinds any objects bound to it.
```

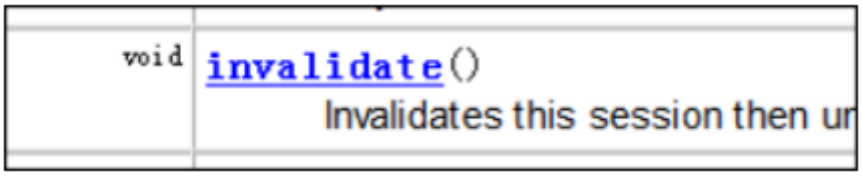
2. 超过session的存活时间（长时间不与服务器交互）（通过**setMaxInactiveInterval**方法指定存活时间）
3. 非正常关闭服务器时销毁， 当正常关闭服务器时，session中的数据将会序列化到服务器的磁盘中（Session的的钝化），当服务器再次启动

的时候，会自动将磁盘中的session数据反序列化到服务器的内存中（Session的活化）。

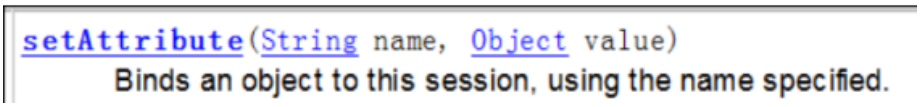
# Session的常用API

获取session的唯一标识，即Cookie中的JSESSIONID

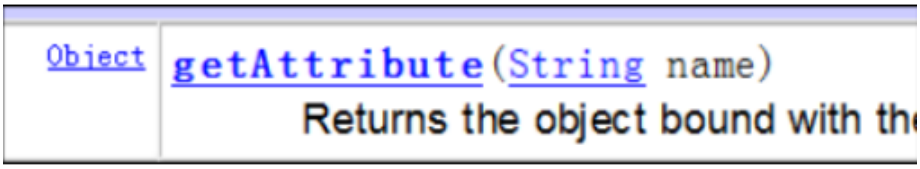

销毁session的方法，通常在退出功能中使用！退出本质就是把用户的session干掉！然后重新跳转到首页




在session中保存某些数据的方法




获取在session中保存的一些数据




从session中移除数据


# Session的使用场景

1.使用session保存用户的登录状态
2.使用session存储验证码信息

# Cookie和session总结（对比）

> *保存位置：cookie保存在客户端（浏览器），而session保存在服务器端（内存中！）*
>
> *安全性：因为cookie数据保存在客户端的磁盘中，因此不安全，应该尽量避免往cookie中存储重要的数据，即使非要往cookie存储数据，也应该加密存储。而session是保存在服务器端，因此相对于cookie要安全！*
>
> *数据大小：浏览器最多为每个域保存20个cookie数据，每个浏览器最多保存300 个cookie，每个cookie的数据量为4KB，而session中的数据保存在服务器的内存中，因此数据的大小和服务器的内存有关！*
>
> *存活时间：*
>
> 1. *cookie和session都可以在服务端设置存活时间，但是Cookie是设置存储在客户端的时间，而Session是设置存储在服务端的时间；*
> 2. *Cookie可以保存在客户端的硬盘中，所以有用户关闭浏览器之后再打开可以再次使用关闭浏览器前的Cookie，因为服务端设置Cookie的最大存活时间为正数，因此Cookie可以在客户端长期存储；*
>
>    *但是Session没有这样的情况，因为保存在客户端的Cookie（Session ID）的最大存活时间为负数（浏览器关闭就失效了），因此Session不能在客户端长期存储；*
> 3. *Cookie可以在客户端手动删除，但是Session不能在客户端操作*
>
> *作用范围：在服务器端可以设置在用户浏览器存储的cookie可以用于那些域、路径下（调用setDomain、setPath方法）【对比第五章中的《会话技术之Cookie.md》】；而session没有这些限制，也就是说session对应的保存在客户端的cookie（JSESSIONID—>该键对应的值）的Domain时localhost、Path是项目的根路径，即该项目下所有网页都可以访问到该cookie。*