Session 使用方法

1. 容器会做所有cookie工作

1)生成会话ID

2)创建新的cookie对象

3)把会话ID放到cookie中

4)把cookie设置为响应的一部分

5)从请求中的cookie获取会话ID

6)将会话ID与现有的会话匹配（会话主要是存储客户信息）

7)把会话与当前请求关联

2.在**响应**中**发送**一个会话cookie，我们要做的只是这一句：

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

向请求要一个会话，容器就会负责余下的工作：

1. 不用创建新的HttpSession对象
2. 不必自己生成唯一的会话ID
3. 不用创建新的cookie对象
4. 不用吧会话ID跟cookie关联
5. 不用在响应中设置Cookie，cookie所有工作都在后台运行

因为，在请求上第一次调用时，会导致随响应发送一个cookie，这大概是判断是否生成对象来实现的

3.从**请求**得到会话ID也只要一句话

HttpSession session = request.getSession();

这和为响应生成会话ID和cookie时用的方法一样

4.也因为都是用一句话，所以我们需要使用额外的方法来判断session是不是新创建的

if(session.**isNew**()){…}

5.当然，也可以不在servlet中获得会话，我们可以在与会话相关的4种listener中，通过事件处理回调方法来访问会话属性。这些事件处理方法取一个类型为HttpSessionEvent的参数，而HttpSessionEvent就有一个getSession方法！

public void sessionCreated(HttpSessionEvent event){

HttpSession session = event.getSession();

//事件处理代码

}

6.如果要为请求匹配一个已经有的会话，可以使用重载的getSession(boolean)

要么得到null，要么得到已有的HttpSession

**//false则表明只得到已有的HttpSession，而不创建新的**

HttpSession session = request.getSession(false);

if(session == null){

session = request.getSession();//此时创建的必然是新会话

一般我们只在需要得到已有的HttpSession时采用getSession(false)