统计网站在线人数

分析

我们在网站中一般使用Session来标识某用户是否登陆了，如果登陆了，就在Session域中保存相对应的属性。如果没有登陆，那么Session的属性就应该为空。

现在，我们想要统计的是网站的在线人数。我们应该这样做：我们监听是否有新的Session创建了，如果新创建了Sesssion，那么在线人数就应该+1。这个在线人数是整个站点的，所以应该有Context对象保存。

大致思路：

监听Session是否被创建了

如果Session被创建了，那么在Context的域对象的值就应该+1

如果Session从内存中移除了，那么在Context的域对象的值就应该-1.

代码

监听器代码：

public class CountOnline implements HttpSessionListener {

public void sessionCreated(HttpSessionEvent se) {

//获取得到Context对象，使用Context域对象保存用户在线的个数

ServletContext context = se.getSession().getServletContext();

//直接判断Context对象是否存在这个域，如果存在就人数+1,如果不存在，那么就将属性设置到Context域中

Integer num = (Integer) context.getAttribute("num");

if (num == null) {

context.setAttribute("num", 1);

} else {

num++;

context.setAttribute("num", num);

}

}

public void sessionDestroyed(HttpSessionEvent se) {

ServletContext context = se.getSession().getServletContext();

Integer num = (Integer) se.getSession().getAttribute("num");

if (num == null) {

context.setAttribute("num", 1);

} else {

num--;

context.setAttribute("num", num);

}

}

}

显示页面代码：

在线人数：${num}

测试

我们每使用一个浏览器访问服务器，都会新创建一个Session。那么网站的在线人数就会+1。

使用同一个页面刷新，还是使用的是那个Sesssion，所以网站的在线人数是不会变的。

自定义Session扫描器

我们都知道Session是保存在内存中的，如果Session过多，服务器的压力就会非常大。

但是呢，Session的默认失效时间是30分钟(30分钟没人用才会失效)，这造成Seesion可能会过多（没人用也存在内存中，这不是明显浪费吗？）

当然啦，我们可以在web.xml文件中配置Session的生命周期。但是呢，这是由服务器来做的，我嫌它的时间不够准确。（有时候我配置了3分钟，它用4分钟才帮我移除掉Session）

所以，我决定自己用程序手工移除那些长时间没人用的Session。

分析

要想移除长时间没人用的Session，肯定要先拿到全部的Session啦。所以我们使用一个容器来装载站点所有的Session。。

只要Sesssion一创建了，就把Session添加到容器里边。毫无疑问的，我们需要监听Session了。

接着，我们要做的就是隔一段时间就去扫描一下全部Session，如果有Session长时间没使用了，我们就把它从内存中移除。隔一段时间去做某事，这肯定是定时器的任务呀。

定时器应该在服务器一启动的时候，就应该被创建了。因此还需要监听Context

最后，我们还要考虑到并发的问题，如果有人同时访问站点，那么监听Session创建的方法就会被并发访问了。定时器扫描容器的时候，可能是获取不到所有的Session的。

这需要我们做同步

于是乎，我们已经有大致的思路了

监听Session和Context的创建

使用一个容器来装载Session

定时去扫描Session，如果它长时间没有使用到了，就把该Session从内存中移除。

并发访问的问题

代码

监听器代码：

public class Listener1 implements ServletContextListener,

HttpSessionListener {

//服务器一启动，就应该创建容器。我们使用的是LinkList(涉及到增删)。容器也应该是线程安全的。

List<HttpSession> list = Collections.synchronizedList(new LinkedList<HttpSession>());

//定义一把锁（Session添加到容器和扫描容器这两个操作应该同步起来）

private Object lock = 1;

public void contextInitialized(ServletContextEvent sce) {

Timer timer = new Timer();

//执行我想要的任务，0秒延时，每10秒执行一次

timer.schedule(new MyTask(list, lock), 0, 10 \* 1000);

}

public void sessionCreated(HttpSessionEvent se) {

//只要Session一创建了，就应该添加到容器中

synchronized (lock) {

list.add(se.getSession());

}

System.out.println("Session被创建啦");

}

public void sessionDestroyed(HttpSessionEvent se) {

System.out.println("Session被销毁啦。");

}

public void contextDestroyed(ServletContextEvent sce) {

}

}

任务代码：

/\*

\* 在任务中应该扫描容器，容器在监听器上，只能传递进来了。

\*

\* 要想得到在监听器上的锁，也只能是传递进来

\*

\* \*/

class MyTask extends TimerTask {

private List<HttpSession> sessions;

private Object lock;

public MyTask(List<HttpSession> sessions, Object lock) {

this.sessions = sessions;

this.lock = lock;

}

@Override

public void run() {

synchronized (lock) {

//遍历容器

for (HttpSession session : sessions) {

//只要15秒没人使用，我就移除它啦

if (System.currentTimeMillis() - session.getLastAccessedTime() > (1000 \* 15)) {

session.invalidate();

sessions.remove(session);

}

}

}

}

}

测试：

15秒如果Session没有活跃，那么就被删除！

使用集合来装载我们所有的Session

使用定时器来扫描session的声明周期【由于定时器没有session，我们传进去就好了】

关于并发访问的问题，我们在扫描和检测session添加的时候，同步起来就好了【当然，定时器的锁也是要外面传递进来的】

踢人小案列

列出所有的在线用户，后台管理者拥有踢人的权利，点击踢人的超链接，该用户就被注销了。

分析

首先，怎么能列出所有的在线用户呢？？一般我们在线用户都是用Session来标记的，所有的在线用户就应该用一个容器来装载所有的Session。。

我们监听Session的是否有属性添加(监听Session的属性有添加、修改、删除三个方法。如果监听到Session添加了，那么这个肯定是个在线用户！)。

装载Session的容器应该是在Context里边的【属于全站点】，并且容器应该使用Map集合【待会还要通过用户的名字来把用户踢了】

思路：

写监听器，监听是否有属性添加在Session里边了。

写简单的登陆页面。

列出所有的在线用户

实现踢人功能(也就是摧毁Session)

代码

监听器

public class KickPerson implements HttpSessionAttributeListener {

// Public constructor is required by servlet spec

public KickPerson() {

}

public void attributeAdded(HttpSessionBindingEvent sbe) {

//得到context对象，看看context对象是否有容器装载Session

ServletContext context = sbe.getSession().getServletContext();

//如果没有，就创建一个呗

Map map = (Map) context.getAttribute("map");

if (map == null) {

map = new HashMap();

context.setAttribute("map", map);

}

//---------------------------------------------------------------------------------------

//得到Session属性的值

Object o = sbe.getValue();

//判断属性的内容是否是User对象

if (o instanceof User) {

User user = (User) o;

map.put(user.getUsername(), sbe.getSession());

}

}

public void attributeRemoved(HttpSessionBindingEvent sbe) {

/\* This method is called when an attribute

is removed from a session.

\*/

}

public void attributeReplaced(HttpSessionBindingEvent sbe) {

/\* This method is invoked when an attibute

is replaced in a session.

\*/

}

}

登陆页面

<form action="${pageContext.request.contextPath }/LoginServlet" method="post">

用户名：<input type="text" name="username">

<input type="submit" value="登陆">

</form>

处理登陆Servlet

//得到传递过来的数据

String username = request.getParameter("username");

User user = new User();

user.setUsername(username);

//标记该用户登陆了！

request.getSession().setAttribute("user", user);

//提供界面，告诉用户登陆是否成功

request.setAttribute("message", "恭喜你，登陆成功了！");

request.getRequestDispatcher("/message.jsp").forward(request, response);

列出在线用户

<c:forEach items="${map}" var="me">

${me.key} <a href="${pageContext.request.contextPath}/KickPersonServlet?username=${me.key}">踢了他吧</a>

<br>

</c:forEach>

处理踢人的Servlet

String username = request.getParameter("username");

//得到装载所有的Session的容器

Map map = (Map) this.getServletContext().getAttribute("map");

//通过名字得到Session

HttpSession httpSession = (HttpSession) map.get(username);

httpSession.invalidate();

map.remove(username);

//摧毁完Session后，返回列出在线用户页面

request.getRequestDispatcher("/listUser.jsp").forward(request, response);

测试

使用多个浏览器登陆来模拟在线用户（同一个浏览器使用的都是同一个Session）