

《JavaEE 实验手册》

Java 教研室

版本 1.3

文档提供: Java 教研室 孙丽萍

《JavaEE 实验手册》 第 1 页 共 15 页

修改记录

修改时间	修改人	修改内容
2009. 9. 01	刘战洪	文档创建
2012. 8. 27	张立飞	修改
2016. 3. 7	孙丽萍	修改

《JavaEE 实验手册》 第 2 页 共 15 页

目录

– ,	内容概述	4
_,	实验一 掌握 ServletContextListener 接口的使用	4
	2.1 实验目的	4
	2.2 准备	4
	2.3 实验步骤	
	2.4 实验结论	10
三、	实验二 理解 ServletContextAttributeListener 接口的作用与使用	10
	3.1 实验目的	
	3.2 准备	10
	3.3 实验步骤	
	3.4 实验结论	11
四、	实验三 利用 Session 实现计数器	11
	4.1 实验目的	11
	4.2 准备	11
	4.3 实验步骤	
	4.4 实验结论	

第五章 监听器(Listener)

一、内容概述

本章的教学内容是介绍 Serv1et2. 5/JSP2. 0 中的监听器 (Listener) 的运作方式和使用时机。

通过本章实验应该达到的目的:认识8个监听器接口及6个事件对象,能够正确的使用监听器接口对 JavaWeb 应用程序运行时所触发的各种事件编写事件处理程序。

二、实验一 掌握 ServletContextListener 接口的使用

2.1 实验目的

掌握 ServletContextListener 接口的使用,理解 JavaWeb 服务器启动时就开始进行监听,当有 Web 应用程序(一个 Server 下可以有多个应用)启动或停止的时候,该监听器会接收到相应的事件。如果想在应用程序启动和关闭时做一些事情,可以实现该接口的方法来编程。

2.2 准备

保证 Eclipse 开发环境的正确配置。

2.3 实验步骤

步骤一:

在 Eclipse 中新建 Dynamic project "02_listener", 用鼠标右键单击 "02_listener /src"目录,选择"New"->"Other",如图 2-1

《JavaEE 实验手册》 第 4 页 共 15 页

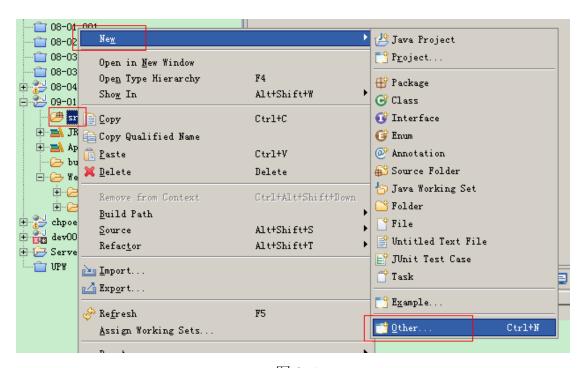


图 2-1 在弹出的窗体中选择"Web"目录下的"Listener"项,如*图 2-2*

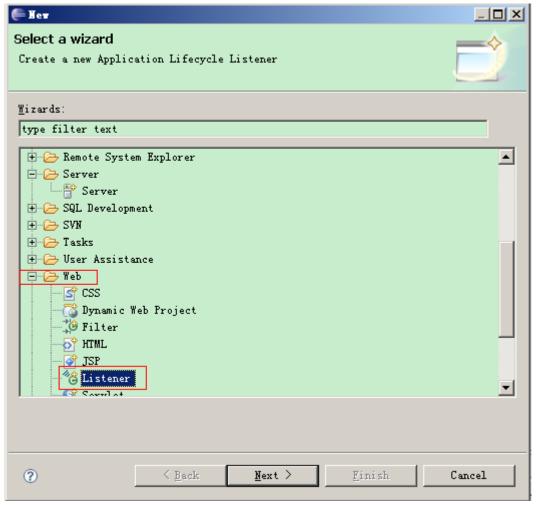


图 2-2

《JavaEE 实验手册》 第 5 页 共 15 页

步骤二:

点击"Next",输入包名(onest.dev)与类名(Serv1etContextListenerDemo),如图 2-3

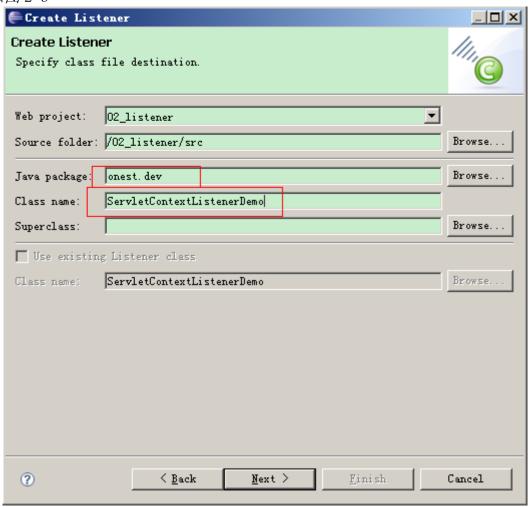


图 2-3

步骤三:

点击 "Next" 按钮,在弹出的窗体中选择 "Lifecycle" 并点击 "Finish",如图 2-4

《JavaEE 实验手册》 第 6 页 共 15 页

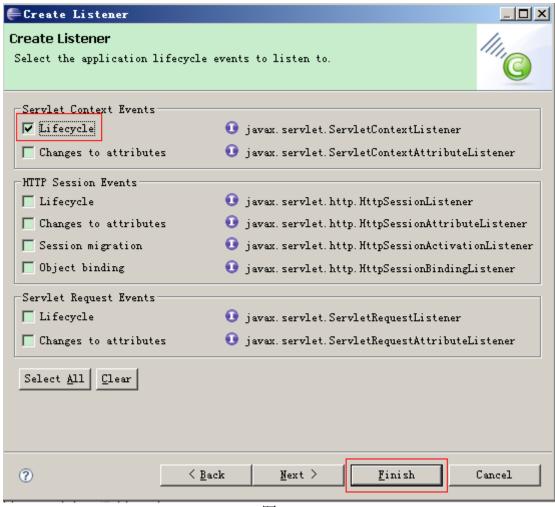


图 2-4

步骤四:

Eclipse 会帮我们生成一个实现了 ServletContextListener 接口的类,并在部署描述符 (web. xml) 中生成相应的配置信息,打开 web. xml 文件,观察生成的配置信息如图 2–5

⟨listener⟩

tener-class>onest.dev.ServletContextListenerDemo</listener-class></listener>

图 2-5

步骤五:

打开生成的 ServletContextListenerDemo 类,并重写 contextInitialized 与 contextDestroyed 两个方法,内容如下

《JavaEE 实验手册》 第 7 页 共 15 页

```
      public void contextInitialized(ServletContextEvent arg0) {

      System.out.println("当Web应用启动时会执行该方法,我们可以在这里做一些初始化工作");

      }

      public void contextDestroyed(ServletContextEvent arg0) {

      System.out.println("当Web应用停止时会执行该方法,我们可以在这里做一些清理工作");

      }
```

步骤六:

运 行 " 02_1 is tener " 项 目 , 在 浏 览 器 中 输 入 : http://localhost:8080/02_1 is tener,回车,观察 Eclipse 的控制台输出,找 到如图 2-6 所示内容

```
Problems @ Javadoc @ Declaration # Servers Console 知 Properties Console 和 Properties Console
```

步骤七:

由于 Tomcat 服务器启动时会运行所有的 Web 程序,所以我们可以在 Eclipse 的控制台下看到相应的 Web 应用启动的信息,但 Eclipse 启动的服务器没有站点管理程序,所以我们将 02_listener 应用部署到真正的 Tomcat 运行环境中(部署应用的步骤参照"Servlet 程序结构与部署")。

启动 Tomcat,观察控制台的输出,如图 2-7

```
osoft SQL Server\90\Tools\binn\;C:\Program Files\TopDomain\e-Learning Class\;C:\_
Program Files\TopDomain\e-Learning Class\;C:\Program Files\Common Files\Thunder
Network\KanKan\Codecs
2009-11-8 2:49:51 org.apache.coyote.http11.Http11Protocol init
信息: Initializing Coyote HTTP/1.1 on http-8080
2009-11-8 2:49:51 org.apache.catalina.startup.Catalina load
信息: Initialization processed in 456 ms
2009-11-8 2:49:51 org.apache.catalina.core.StandardService start
信息: Starting service Catalina
2009-11-8 2:49:51 org.apache.catalina.core.StandardEngine start
信息: Starting Servlet Engine: Apache Tomcat/6.0.18
2009-11-8 2:49:51 org.apache.catalina.core.StandardContext addApplicationListene
信息: The listener "onest.dev.ServletContextListenerDemo" is already configured
for this context. The duplicate definition has been ignored.
当Web应用启动时会执行该方法,我们可以在这里做一些初始化工作
2009-11-8 2:49:52 org.apache.coyote.http11.Http11Protocol start
信息: Starting Coyote HTTP/1.1 on http-8080
2009-11-8 2:49:52 org.apache.jk.common.ChannelSocket init
信息: JK: ajp13 listening on /0.0.0.0:8009
2009-11-8 2:49:52 org.apache.jk.server.JkMain start
信息: Jk running ID=0 time=0/31 config=null
009-11-8 2:49:52 org.apache.catalina.startup.Catalina start
信息: Server startup in 745 ms
```

《JavaEE 实验手册》 第 8 页 共 15 页

图 2-7

进入 Tomcat 的管理程序(用户名和密码在 tomcat-user.xml 中设置), 停止 02 listener 应用,如图 2-8。

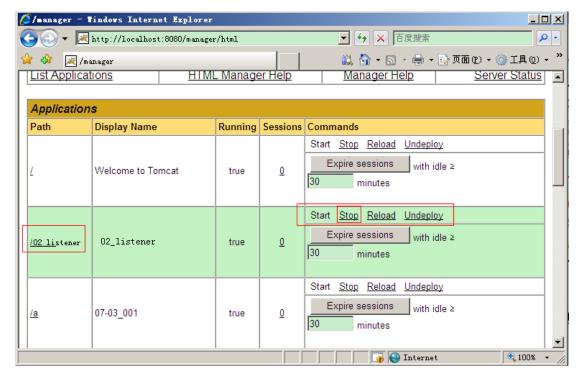


图 2-8

观察控制台的变化,如图 2-9

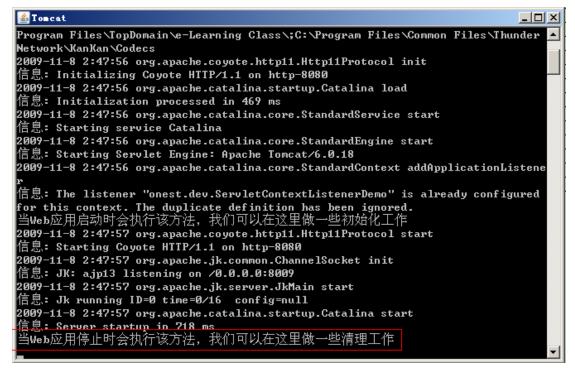


图 2-9

《JavaEE 实验手册》 第 9 页 共 15 页

2.4实验结论

通过以上实验可以知道,当 Servlet 容器启动时就开始进行监听工作,如果在该服务器上有 Web 应用程序运行(启动),就会被监听到并执行 ServletContextListener 监听器的 contextInitialized 方法。当 Web 应用程序停止时,会调用 contextDestroyed 方法,我们可以在相应的方法中编写自己的代码完成自己所需要的功能。

三、实验二 理解 ServletContextAttributeListener 接口的作用与使用

3.1 实验目的

编写与部署 ServletContextAttributeListener 接口的实现,理解该接口的作用与使用。

3.2 准备

完成实验一,在实验一的基础上进行。

3.3 实验步骤

步骤一:

在 "02_listener" 项 目 中 源 代 码 目 录 下 新 建 ServletContextAttributeListener 接口的实现类, 创建过程与实验一的过程相 同, 只是实现的接口不同。

实现过程:

[由学生完成]

步骤二:

观察 web. xml 中的配置信息,并重写 attributeAdded、attributeReplaced、attributeRemoved 三个方法,要求完成以下功能:

- ◆ 添加 ServletContext 属性时,在控制台输出: AttributeAdd
- ◆ 更改 ServletContext 属性时,在控制台输出: AttributeReplace
- ◆ 移除 ServletContext 属性时,在控制台输出: AttributeRemove

代码:

[由学生完成]

步骤三:

编写测试页面

代码:

[由学生完成]

《JavaEE 实验手册》

提示: 可以使用 JSP 隐含对象 application 的 setAttribute 等方法对属性进行添加、更改和移除操作。

3.4 实验结论

通过以上实验掌握 ServletContextAttributeListener 的使用。

四、实验三 利用 Session 实现计数器

4.1 实验目的

编写与部署与Session相关的监听器接口的实现,理解各接口的作用与使用,并能正确区分各接口主要功能的异同。

编写一个计数器,当有客户访问网站时,统计在线人数。当有会员登录时,将会员姓名保存到 Session,并统计在线会员人数。

4.2 准备

完成实验二,在实验二的基础上进行实验。

4.3 实验步骤

步骤一:

在"02-listener/src"目录下新建 HttpSessionLinstenerDemo 类并实现 HttpSessionListener 接口, 重写 sessionCreated 和 sessionDestroyed 两个方法,代码如下:

《JavaEE 实验手册》 第 11 页 共 15 页

```
public void sessionCreated(HttpSessionEvent arg0) {
    int onlineNum = (Integer)
arg0.getSession().getServletContext().getAttribute("onlineNum");
    arg0.getSession().getServletContext().setAttribute("onlineNum",
++onlineNum);
}

public void sessionDestroyed(HttpSessionEvent arg0) {
    int onlineNum = (Integer)
arg0.getSession().getServletContext().getAttribute("onlineNum");
    arg0.getSession().getServletContext().setAttribute("onlineNum",
--onlineNum);
    int onlineMemberNum = (Integer)
arg0.getSession().getServletContext().getAttribute("onlineMemberNum");
    arg0.getSession().getServletContext().setAttribute("onlineMemberNum");
    arg0.getSession().getServletContext().setAttribute("onlineMemberNum",
--onlineMemberNum);
}
```

说明: 当有客户端访问站点时,服务器端都会为其创建一个 HttpSession 对象,当有新的 HttpSession 对象被创建时,会执行 sessionCreated 方法,在该方法中我们使用 application 范围 (ServletContext 对象的属性,类似全局变量)的变量 onlineNum 来保存在线用户的人数,将在线人数加 1。当会话失效时,会调用 sessionDestroyed 方法,我们在该方法中将在线人数和已经登录的会员数均减 1 (已经登录的会员数用 onlineMemberNum 保存)。

步骤二:

为保证全局变量中有 onlineNum 和 onlineMemberNum 两个变量,所以我们修改 ServletContextListenerDemo 类中的 contextInitialized 方法,该方法是当 Web 应用程序启动时被执行的,所以我们可以在这里进行初始化的设置,代码如下所示:

```
public void contextInitialized(ServletContextEvent arg0) {
    System.out.println("当Web应用启动时会执行该方法,我们可以在这里做一些初始化工作");
    //向application范围添加两个全局变量,分别统计在线人数和在线会员数 arg0.getServletContext().setAttribute("onlineNum", 0);
    arg0.getServletContext().setAttribute("onlineMemberNum", 0);
}
```

说明: 当 Web 应用一启动时就在全局范围内设置两个变量 onlineNum 和 onlineMemberNum,并设置其初始值均为零。

步骤三:

模拟会员登录的动作,当有会员登录操作时,会在 Session 中保存其会员名称,会触发 HttpSessionAttributeListener 监听器的 attributeAdded 方法,所以新增 HttpSessionAttributeLinstenerDemo 实现类,修改 attributeAdded 方法,代码如下

```
public void attributeAdded(HttpSessionBindingEvent arg0) {
    int onlineMemberNum = (Integer)
arg0.getSession().getServletContext().getAttribute("onlineMemberNum");
arg0.getSession().getServletContext().setAttribute("onlineMemberNum", ++onlineMemberNum);
}
```

说明:已登录的会员数保存在 onlineMemberNum 全局变量中,当有会员登录,在 HttpSession 新增 membername 属性时 Servlet 容器会调用 attributeAdded 方法,将已经登录会员数加 1。因为会员登出或者超出会话有效期时都会导致会话失效,在 HttpSessionLinstenerDemo 类中已经完成了人数减 1 的动作,在这里不再重复计算。

步骤四:

修改 index. jsp 文件,提供用户输入用户名并登录的操作界面,关键代码如下所示:

步骤五:

当用户输入名称点击"登录"按钮时,将用户名提交到 login. jsp 页面在 Session 中保存会员名称, login. jsp 的关键代码如下:

```
<body>

<%

String memberName = request.getParameter("name");
session.setAttribute("membername", memberName);
response.sendRedirect("index.jsp");
%>
</body>
```

步骤六:

修改 web. xml 文件,加入以下标签:

说明:设置会话超时期为1分钟。

步骤七:

运行项目,分别打开 IE 浏览器和 FireFox 浏览器,并都访问 http://localhost:8080/02-sessionlistener /index.jsp。效果如图4-1和图4-2

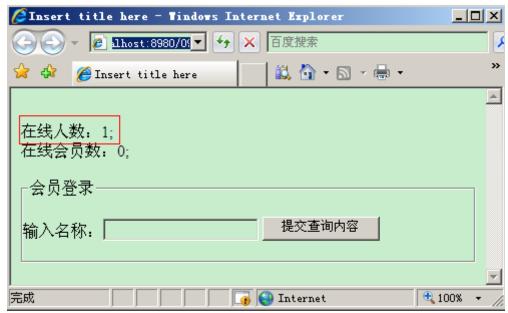


图 4-1

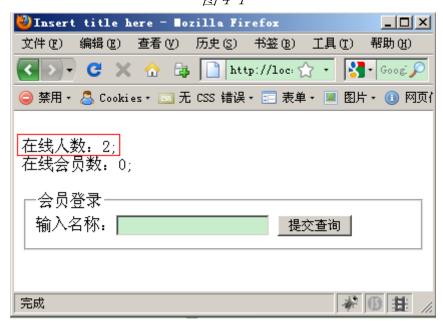


图 4-2

说明: 刷新 IE 浏览器,会看到在线人数为 2,在 FireFox 浏览器中输入任 意用户名并提交,会显示在线会员数为 1。不进行任何操作等待一分钟后,分别刷新 IE 和 FireFox 浏览器,这时两个浏览器的会话都已失效,在线人数仍为 2,

《JavaEE 实验手册》 第 14 页 共 15 页

这是因为刷新时向服务器提交请求,服务器又为该客户端请求创建了新的会话。 在线会员数为-1,因为无论已经登录的 Session 还是未登录的 Session 过期时都 会将已登录人数减 1,所以产生了负数,这是程序中的一个 Bug。

Bug 修改方法: [由学生完成]

4.4 实验结论

通过以上实验掌握与 Session 相关监听器的使用。

《JavaEE 实验手册》 第 15 页 共 15 页