监听器：

什么是监听器，监听器的作用？

在J2EE里面，所谓的监听器就是对整个ＷＥＢ的监听

监听主要分为以下三类：

·ServletContext:Servlet上下文监听

·session监听

·request监听

一：对ServletContext进行监听

对WEB进行监听，就是实现一系列的监听接口

1

·ServletContextListener:对Servlet上下文监听的接口（启动，销毁）

void contextInitialized(ServletContextEvent sce)：启动上下文监听

void contextDestroyed(ServletContextEvent sce)：销毁上下文监听

·ServletContextEvent：取得一个（application）对象

2

·ServletContextAttributeListener：对Servlet上下文属性进行监听

void attributeAdded(ServletContextAttributeEvent scab)：增加属性

void attributeRemoved(ServletContextAttributeEvent scab)：移出属性

void attributeReplaced(ServletContextAttributeEvent scab)：修改属性

ServletContextAttributeEvent：事件，得到设置属性的内容和名称

getName()：取得设置属性的名称

getValue():取得设置属性的值

|  |
| --- |
| package com.jjm.listenerdemo;  import javax.servlet.ServletContextAttributeEvent;  import javax.servlet.ServletContextAttributeListener;  import javax.servlet.ServletContextEvent;  import javax.servlet.ServletContextListener;  public class AppListener implements ServletContextListener,  ServletContextAttributeListener {  //覆写接口的方法  public void contextInitialized(ServletContextEvent sce) {  System.out.println("初始化上下文监听接口");  }  public void contextDestroyed(ServletContextEvent sce) {  System.out.println(" 销毁上下文监听接口");  }  public void attributeAdded(ServletContextAttributeEvent sca) {  System.out.println("app增加属性，属性名称："+sca.getName()+"——————>属性值："+sca.getValue());  }  public void attributeRemoved(ServletContextAttributeEvent sca) {  System.out.println("app删除属性，属性名称："+sca.getName()+"——————>属性值："+sca.getValue());  }  public void attributeReplaced(ServletContextAttributeEvent sca) {  System.out.println("app修改属性，属性名称："+sca.getName()+"——————>属性值："+sca.getValue());  }    }  /\*<listener>  <listener-class>com.jjm.listenerdemo.AppListener</listener-class>  </listener>  \*/ |

上下文监听主要是对容器初始化、销毁、属性的监听操作。

二：对session的监听

对session创建、销毁和属性的监听

·HttpSessionListener：对session整体状况的监听（创建、销毁的监听）

void sessionCreated(HttpSessionEvent se)：session创建的时候

void sessionDestroyed(HttpSessionEvent se)：session销毁的时候

HttpSessionEvent：事件

public HttpSession getSession()：取得当前的session

·HttpSessionAttributeListener:对session属性的监听

void attributeAdded(HttpSessionBindingEvent se)：增加属性

void attributeRemoved(HttpSessionBindingEvent se)：删除属性

void attributeReplaced(HttpSessionBindingEvent se)：修改属性

HttpSessionBindingEvent：事件

getName():取得设置属性的名称

getValue()：取得设置属性的值

getSession():取得当前的session

|  |
| --- |
| package com.jjm.listenerdemo;  import javax.servlet.ServletContextAttributeEvent;  import javax.servlet.ServletContextAttributeListener;  import javax.servlet.ServletContextEvent;  import javax.servlet.ServletContextListener;  public class AppListener implements ServletContextListener,  ServletContextAttributeListener {  //覆写接口的方法  public void contextInitialized(ServletContextEvent sce) {  System.out.println("初始化上下文监听接口");  }  public void contextDestroyed(ServletContextEvent sce) {  System.out.println(" 销毁上下文监听接口");  }  public void attributeAdded(ServletContextAttributeEvent sca) {  System.out.println("app增加属性，属性名称："+sca.getName()+"——————>属性值："+sca.getValue());  }  public void attributeRemoved(ServletContextAttributeEvent sca) {  System.out.println("app删除属性，属性名称："+sca.getName()+"——————>属性值："+sca.getValue());  }  public void attributeReplaced(ServletContextAttributeEvent sca) {  System.out.println("app修改属性，属性名称："+sca.getName()+"——————>属性值："+sca.getValue());  }    }  /\*<listener>  <listener-class>com.jjm.listenerdemo.SessListener</listener-class>  </listener>  \*/ |

session销毁：

·session超时

|  |
| --- |
| <session-config>  <session-timeout>1</session-timeout>  </session-config> |

session一分钟失效

·手动使session失效

invalidate()：是session失效

案例：

在线人员列表

实现那几个接口？

首先在线人员应该保存在服务器上下文中，所以必须实现ServletContext接口

一个用户就是一个Session要实现HttpSessionListener接口

用户登录之后要将用户名保存在上下文中，保存的内容是不是用户名？

登录成功之后需要把用户名保存在ServletContext中，所以在监听器中要通过设置属性的方式取得用户名所以要实现HttpSessionListener接口

|  |
| --- |
| **package** com.jjm.listenerdemo;  **import** java.util.ArrayList;  **import** java.util.List;  **import** javax.servlet.ServletContext;  **import** javax.servlet.ServletContextEvent;  **import** javax.servlet.ServletContextListener;  **import** javax.servlet.http.HttpSessionAttributeListener;  **import** javax.servlet.http.HttpSessionBindingEvent;  **import** javax.servlet.http.HttpSessionEvent;  **import** javax.servlet.http.HttpSessionListener;  **public** **class** OnlineListener **implements** ServletContextListener, HttpSessionListener,  HttpSessionAttributeListener {  **private** ServletContext application;  **public** **void** contextDestroyed(ServletContextEvent sce) {    }  **public** **void** contextInitialized(ServletContextEvent sce) {  //容器初始化，在application中存放一个空的集合  **this**.application = sce.getServletContext();  **this**.application.setAttribute("allUser", **new** ArrayList<String>());  }  **public** **void** sessionCreated(HttpSessionEvent se) {    }  **public** **void** sessionDestroyed(HttpSessionEvent se) {  System.*out*.print("+++++++++++++++++++++++");  //注销之后要删除集合中的对象的用户  List<String> list = (List<String>)**this**.application.getAttribute("allUser");  String user = (String)se.getSession().getAttribute("username");  list.remove(user);  **this**.application.setAttribute("allUser", list);  }  **public** **void** attributeAdded(HttpSessionBindingEvent se) {  //登陆之后 session设置属性  List list = (List)**this**.application.getAttribute("allUser");  list.add( se.getValue());  **this**.application.setAttribute("allUser", list);  }  **public** **void** attributeRemoved(HttpSessionBindingEvent se) {    }  **public** **void** attributeReplaced(HttpSessionBindingEvent se) {    }  } |
| <%@page contentType=*"text/html; charset=GBK"*%>  <%@page import=*"java.util.\*"* %>  <form action=*"Login.jsp"* method=*"post"*>  用户名：<input type=*"text"* name=*"username"*><input type=*"submit"* value=*"登陆"*>  </form>  <a href=*"demo03.jsp"*>注销</a>  <!-- 要向session中设置输入的用户名-->  <%  **if**(request.getParameter("username")!=**null**){  session.setAttribute("username",request.getParameter("username"));  }  %>  <h1>在线人员列表</h1>  <hr/>  <%  List<String> list = (List<String>)application.getAttribute("allUser");  Iterator<String> it = list.iterator();  **while**(it.hasNext()){    %>  <li><%=it.next()%></li>  <%  }  %> |
| <%  session.invalidate();  response.sendRedirect("Login.jsp");  %> |