# 监听器概念：

监听器就是一个实现特定接口的普通java程序，这个程序专门用于监听一个java对象的方法调用或属性改变，当被监听对象发生上述事件后，监听器某个方法将立即被执行。

1、事件源：发生事件的对象。

2、监听器：是一个接口，监听事件源上要发生的动作

3、事件：事件对象一般是作为监听器方法的参数存在的，它封装了发生事件的对象

# Demo示例

**public** **class** FrameDemo {

**public** **static** **void** main(String[] args) {

Frame f = **new** Frame("我的窗体");//事件源

f.setSize(600, 400);

f.setVisible(**true**);

//注册监听器

// f.addWindowListener(new WindowListener() {

//

// public void windowOpened(WindowEvent e) {

//

// }

//

// public void windowIconified(WindowEvent e) {

//

// }

//

// public void windowDeiconified(WindowEvent e) {

//

// }

//

// public void windowDeactivated(WindowEvent e) {

//

// }

//

// public void windowClosing(WindowEvent e) {//事件对象一般是作为监听器方法的参数存在的，它封装了发生事件的对象

// Frame ff = (Frame)e.getSource();

// ff.dispose();//关闭窗口

// }

//

// public void windowClosed(WindowEvent e) {

//

// }

//

// public void windowActivated(WindowEvent e) {

//

// }

// });

f.addWindowListener(**new** WindowAdapter() {

@Override

**public** **void** windowClosing(WindowEvent e) {

Frame ff = (Frame)e.getSource();

ff.dispose();//关闭窗口

}

});

}

}

执行后显示一个窗体，点击关闭后将关闭当前窗体

# 自己动手写一个模拟示例来理解

事件源：

//事件源

**public** **class** Student **implements** Serializable,HttpSessionActivationListener,HttpSessionBindingListener{

**private** String name;

**private** StudentListener listener;

**public** Student(String name){//注入：通过构造方法

**this**.name = name;

}

**public** String getName() {

**return** name;

}

**public** **void** eat(){

**if**(listener!=**null**){

listener.preEat(**new** StudentEvent(**this**));

}

System.*out*.println("开始吃");

}

**public** **void** study(){

**if**(listener!=**null**){

listener.preStudy(**new** StudentEvent(**this**));

}

System.*out*.println("开始学");

}

//注册监听

**public** **void** addStudentListener(StudentListener listener){

**this**.listener = listener;

}

}

监听器：

**public** **interface** StudentListener {

**void** preEat(StudentEvent e);//监听吃

**void** preStudy(StudentEvent e);//监听学

}

事件：

**public** **class** StudentEvent {

**private** Object source;

**public** StudentEvent(Object source){

**this**.source = source;

}

**public** Object getSource() {//获取事件源

**return** source;

}

}

测试：

**public** **class** StudentTest {

**public** **static** **void** main(String[] args) {

Student s = **new** Student("葛付以");

//注册监听器

s.addStudentListener(**new** StudentListener() {

**public** **void** preStudy(StudentEvent e) {

Student ss = (Student) e.getSource();

System.*out*.println(ss.getName()+"喝杯奶吧");

}

**public** **void** preEat(StudentEvent e) {

Student ss = (Student) e.getSource();

System.*out*.println(ss.getName()+"慢慢吃");

}

});

s.eat();

s.study();

}

}

# Servlet中提供的监听器（8个）

八个监听器分类：

## 监听ServletContext、HttpSession、ServletRequest对象的创建和销毁的监听器。

ServletContextListener:监听ServletContext对象的创建和销毁。

HttpSessionListener:监听HttpSession对象的创建和销毁。

创建：第一次调用request.getSession()时。

销毁：1、主动调用invalidate()方法

2、超时

ServletRequestListener:监听ServletRequest对象的创建和销毁。

## 监听ServletContext、HttpSession、ServletRequest对象中域变化（新来的，替换的，删除的）的监听器。

ServletContextAttributeListener:

HttpSessionAttributeListener:

ServletRequestAttributeListener:

## 感知型监听器：**谁实现了这些接口，谁就能感知自己被怎么着了。这种监听器不需要注册。**

HttpSessionActivationListener：感知自己何时随着HttpSession对象钝化和活化

HttpSessionBindingListener：感知自己何时被HttpSession对象绑了（绑在域中）和解绑了。

# 编写步骤：

1、编写一个类实现某个监听器接口

2、在web.xml中注册监听器

<listener>

<listener-class>cn.itcast.listener.ServletContextDemoListener</listener-class>

</listener>

监听器的实现类会实现某个监听器，所以根据这个实现的监听器就可以知道具体监听的对象是哪个

# 示例代码演示

## ServletContextListener

Spring融合到web项目中的原理就是使用类实现了ServletContextListener，在web容器初始化时，将配置文件中的bean实体进行实例化并放入到内存中，所以Spring融合到web项目中只需要在web.xml文件中加入spring自定义的监听器即可

//相当有用：

//public void contextInitialized(ServletContextEvent sce):此方法中的代码是在应用被加载时执行的，执行一次。

//因此，可以把系统的一些初始化工作代码写在此处。应用场景

**public** **class** ServletContextDemoListener **implements** ServletContextListener {

//当ServletContext对象被销毁时执行

**public** **void** contextDestroyed(ServletContextEvent sce) {

System.*out*.println("ServletContext被销毁了");

}

//当ServletContext对象被创建时执行

**public** **void** contextInitialized(ServletContextEvent sce) {

System.*out*.println("ServletContext被创建了");

init();

}

/\*\*

<beans>

<bean name="uDao" class="cn.itcast.dao.impl.UserDao"/>

<bean name="rDao" class="cn.itcast.dao.impl.RoleDao"/>

<bean name="mDao" class="cn.itcast.dao.impl.MenuDao"/>

<bean name="..Dao" class="cn.itcast.dao.impl...Dao"/>

</beans>

\*/

**public** **void** init(){//Spring的初始化原理

//Map<String,Object>:key:对象引用名，value：对象本身

//XML中读取所有的bean元素

//遍历：取到bean元素的name和class值

//利用反射：Object obj = Class.forName(classValue).newInstance();

//map.put(name,obj);

}

}

<listener>

<listener-class>cn.itcast.listener.ServletContextDemoListener</listener-class>

</listener>

Spring融合Web项目中：

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

## HttpSessionListener

**public** **class** HttpSessionDemoListener **implements** HttpSessionListener {

//HttpSession对象被创建时调用

**public** **void** sessionCreated(HttpSessionEvent se) {

HttpSession session = se.getSession();

System.***out***.println(session+"被创建了");

}

**public** **void** sessionDestroyed(HttpSessionEvent se) {

HttpSession session = se.getSession();

System.***out***.println(session+"被销毁了");

}

}

### 应用场景：自定义HttpSession扫描器

//自定义HttpSession扫描器，如果一个HttpSession对象超过1分钟没有用了，主动干掉。

//基础：集合在遍历时，不允许增删其中的元素的。否则会java.util.ConcurrentModificationException

// java.util.ConcurrentModificationException

**public** **class** HttpSessionScannerListener **implements** HttpSessionListener,

ServletContextListener {

**private** List<HttpSession> sessions = **new** ArrayList<HttpSession>();// 存放所有的HttpSession对象

**private** Object lock = **new** Object();

// 启动定时器

**public** **void** contextInitialized(ServletContextEvent sce) {

Timer t = **new** Timer();// 每隔一分钟扫描一次

t.schedule(**new** SessionScanTask(sessions,lock), 0, 1 \* 60 \* 1000);// 启动扫描任务

}

// 拿到一个个会话对象

**public** **void** sessionCreated(HttpSessionEvent se) {

**synchronized** (lock) {

HttpSession session = se.getSession();

System.***out***.println(session + "来了");

sessions.add(session);

}

}

**public** **void** sessionDestroyed(HttpSessionEvent se) {

HttpSession session = se.getSession();

System.***out***.println(session + "去了");

}

**public** **void** contextDestroyed(ServletContextEvent sce) {

}

}

**class** SessionScanTask **extends** TimerTask {

**private** List<HttpSession> sessions;

**private** Object lock;

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

**this**.sessions = sessions;

**this**.lock = lock;

}

// 扫描HttpSession对象，看看有没有超时的

**public** **void** run() {

**synchronized** (lock) {

ListIterator<HttpSession> it = sessions.listIterator();

**while** (it.hasNext()) {

HttpSession session = it.next();

**if** ((System.*currentTimeMillis*() - session.getLastAccessedTime()) > 1 \* 60 \* 1000) {

session.invalidate();

it.remove();

}

}

}

}

}

## ServletRequestListener

//有什么用？

//统计每个资源的访问次数

**public** **class** ServletRequestDemoListener **implements** ServletRequestListener {

**public** **void** requestDestroyed(ServletRequestEvent sre) {

System.***out***.println(sre.getServletRequest()+"销毁了");

}

**public** **void** requestInitialized(ServletRequestEvent sre) {

System.***out***.println(sre.getServletRequest()+"创建了");

}

}

## HttpSessionAttributeListener

//踢人案例

**public** **class** HttpSessionAttributeListenerDemo **implements**

HttpSessionAttributeListener {

**public** **void** attributeAdded(HttpSessionBindingEvent se) {

System.***out***.println(se.getSession()+"放了新东东");

}

**public** **void** attributeRemoved(HttpSessionBindingEvent se) {

System.***out***.println(se.getSession()+"删除了东东");

}

**public** **void** attributeReplaced(HttpSessionBindingEvent se) {

System.***out***.println(se.getSession()+"东东替换成了西西");

}

}

## HttpSessionActivationListener,HttpSessionBindingListener

**public** **class** Student **implements** Serializable,HttpSessionActivationListener,HttpSessionBindingListener{

**public** **void** sessionDidActivate(HttpSessionEvent se) {

System.***out***.println("自己随着session活化了");

}

**public** **void** sessionWillPassivate(HttpSessionEvent se) {

System.***out***.println("自己随着session钝化了");

}

**public** **void** valueBound(HttpSessionBindingEvent event) {

System.***out***.println("自己被绑到了session域中");

}

**public** **void** valueUnbound(HttpSessionBindingEvent event) {

System.***out***.println("自己从session域中被移除了");

}

}

## 踢人案例

User:

**public** **class** User **implements** Serializable {

**private** String username;

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

}

login.jsp:

<%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>

<%@ taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"*%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<title></title>

<meta http-equiv=*"pragma"* content=*"no-cache"*>

<meta http-equiv=*"cache-control"* content=*"no-cache"*>

<meta http-equiv=*"expires"* content=*"0"*>

<!--

<link rel="stylesheet" type="text/css" href="styles.css">

-->

</head>

<body>

<c:if test=*"*${sessionScope.user==null}*"*>

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

username:<input type=*"text"* name=*"username"*/>(用户名不能重复，且不能为空)<br/>

<input type=*"submit"* value=*"登录"*/>

</form>

</c:if>

<c:if test=*"*${sessionScope.user!=null}*"*>

欢迎您：${sessionScope.user.username}<a href=*"#"*>注销</a>

</c:if>

<a href=*"onlineuser.jsp"*>在线用户</a>

</body>

</html>

LoginServlet:

**public** **class** LoginServlet **extends** HttpServlet {

**public** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("UTF-8");

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

//取到用户名：

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

//验证用户是否合法（略）

User user = **new** User();

user.setUsername(username);

//正确：放到HttpSession中

HttpSession session = request.getSession();

session.setAttribute("user", user);

out.write("登录成功");

}

**public** **void** doPost(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

doGet(request, response);

}

}

OnlineUserListener :

//此处核心代码

**public** **class** OnlineUserListener **implements** HttpSessionAttributeListener {

//只要向HttpSession的域中放东东都会执行该方法

//我们要的是放的是User对象，标记还是user的东东

//找个地方存起来：容器要求唯一；页面上还能访问到该容器

**public** **synchronized** **void** attributeAdded(HttpSessionBindingEvent se) {

//找到存储的是user标记，且绑定的对象时User类

HttpSession session = se.getSession();

ServletContext sc = session.getServletContext();

Object obj = session.getAttribute("user");

**if**(obj!=**null**&&obj **instanceof** User){

//向容器中存起来

Map<String,HttpSession> map = (Map<String, HttpSession>) sc.getAttribute("users");//存放东东的容器（Map<String,HttpSession>.key:用户名；value:HttpSession对象）

**if**(map==**null**){

map = **new** HashMap<String, HttpSession>();//Collections.synchronizedMap(map)

sc.setAttribute("users", map);

}

map.put(((User)obj).getUsername(), session);

}

}

**public** **synchronized** **void** attributeRemoved(HttpSessionBindingEvent se) {

}

**public** **void** attributeReplaced(HttpSessionBindingEvent se) {

}

}

onlineuser.jsp:

<%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>

<%@ taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"*%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<title></title>

<meta http-equiv=*"pragma"* content=*"no-cache"*>

<meta http-equiv=*"cache-control"* content=*"no-cache"*>

<meta http-equiv=*"expires"* content=*"0"*>

<!--

<link rel="stylesheet" type="text/css" href="styles.css">

-->

</head>

<body>

<h1>在线用户</h1>

<c:if test=*"*${**empty** applicationScope.users}*"*>

对不起！没有在线用户

</c:if>

<c:if test=*"*${!**empty** applicationScope.users}*"*>

<c:forEach items=*"*${applicationScope.users}*"* var=*"me"*>

<c:url value=*"/servlet/KickServlet"* var=*"url"*>

<c:param name=*"username"* value=*"*${me.key}*"*></c:param>

</c:url>

${me.key}&nbsp;&nbsp;<a href=*"*${url}*"*>踢出</a><br/>

</c:forEach>

</c:if>

</body>

</html>

KickServlet:

//踢人

**public** **class** KickServlet **extends** HttpServlet {

**public** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

ServletContext sc = getServletContext();

Map<String, HttpSession> users = (Map<String, HttpSession>) sc.getAttribute("users");

**if**(users!=**null**){

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

username = **new** String(username.getBytes("ISO-8859-1"),"UTF-8");

HttpSession s = users.get(username);

**if**(s!=**null**){

s.invalidate();

users.remove(username);

}

}

response.sendRedirect(request.getContextPath()+"/onlineuser.jsp");

}

**public** **void** doPost(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

doGet(request, response);

}

}

配置监听器：

<listener>

<listener-class>cn.itcast.domain.OnlineUserListener</listener-class>

</listener>