第 11 章 监听器

第 2 节主要内容

11.1 监听器概述

1. 监听器本质

（1）特殊类

JavaWeb 中的监听器是 Servlet 规范中定义的一种特殊类

（2）作用

它用于监听 web 应用程序中的 ServletContext，HttpSession 和 ServletRequest 等作用域对象的创建与销毁事件，以及监听这些域对象中的属性发生修改的事件。

2. 监听器分类

事件源（监听对象）	作用域对象定义	监听事件
共享对象	ServletContext	自身创建和销毁
		属性增加、修改、删除
会话对象	HttpSession	自身创建和销毁
		属性增加、修改、删除
		绑定到 HttpSession 域中的某个对象的状态
请求对象	ServletRequest	自身创建和销毁
		属性增加、修改、删除

11.2 监听作用域对象的创建和销毁

1. 监听 ServletContext 域对象的创建和销毁

（1）实现 ServletContextListener 接口

（2）何时触发哪个方法

何时	激活（触发）方法
创建 ServletContext 对象	contextInitialized()
销毁 ServletContext 对象	contextDestroyed()

（3）实例代码

```
public class MyServletContextListener implements ServletContextListener {
   public void contextInitialized(ServletContextEvent sce) {
      System.out.println("ServletContext 对象创建");
   }
   public void contextDestroyed(ServletContextEvent sce) {
      System.out.println("ServletContext 对象销毁");
   }
}
```

（4）web.xml 文件中注册监听器代码

```
<listener>
      <!--实现了 ServletContextListener 接口的监听器类 -->
      <listener-class>me.gacl.web.listener.MyServletContextListener</listener-class>
</listener>
```

经过这两个步骤，我们就完成了监听器的编写和注册，Web 服务器在启动时，就会自动把在 web.xml 中配置的监听器注册到 ServletContext 对象上，这样开发好的 MyServletContextListener 监听器就可以对 ServletContext 对象进行监听了。

2. 监听 HttpSession 域对象的创建和销毁

（1）实现 HttpSessionListener 接口

（2）何时触发哪个方法

何时	激活（触发）方法
创建 Session 对象	sessionCreated (HttpSessionEvent se)
销毁 Session 对象	sessionDestroyed (HttpSessionEvent se)

（3）实例代码

```
public class MyHttpSessionListener implements HttpSessionListener {
   public void sessionCreated(HttpSessionEvent se) {
         System.out.println( se.getSession() + "创建了！！");
   }
     /* HttpSession 的销毁时机需要在 web.xml 中进行配置，如下：
```

```
        * <session-config>
                <session-timeout>1</session-timeout>
            </session-config>
            这样配置就表示 session 在 1 分钟之后就被销毁
     */
   public void sessionDestroyed(HttpSessionEvent se) {
          System.out.println("session 销毁了！！");
   }
}
```

（4）在 web.xml 文件中注册监听器

```
<!--注册针对 HttpSession 对象进行监听的监听器-->
<listener>
    <listener-class>me.gacl.web.listener.MyHttpSessionListener</listener-class>
</listener>
<!-- 配置 HttpSession 对象的销毁时机 -->
<session-config>
        <!--配置 HttpSession 对象的 1 分钟之后销毁 -->
        <session-timeout>1</session-timeout>
</session-config>
```

当我们访问 jsp 页面，HttpSession 对象就会创建，此时就可以在 HttpSessionListener 观察到 HttpSession 对象的创建过程了，我们可以写一个 jsp 页面观察 HttpSession 对象创建的过程。如下：index.jsp

一访问 JSP 页面，HttpSession 就创建了，创建好的 Session 的 Id 是：${pageContext.session.id}

3. 监听 ServletRequest 域对象的创建和销毁

（1）实现 ServletRequestListener 接口

（2）何时触发哪个方法

何时	激活（触发）方法
创建 Request 对象	requestInitialized(ServletRequestEvent sre)
销毁 Request 对象	requestDestroyed(ServletRequestEvent sre)

（3）实例代码

```
public class MyServletRequestListener 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！！");
    }
}
```

11.3 监听作用域对象中属性的变更的监听器

1. 基本概念

类实现的监听器接口	方法名称	参数类型	作用
ServletContextAttributeListener	attributeAdded	ServletContextAttributeEvent	添加属性时触发
	attributeReplaced		替换属性时触发
	attributeRemoved		删除属性时触发
HttpSessionAttributeListener	attributeAdded	HttpSessionBindingEvent	添加属性时触发
	attributeReplaced		替换属性时触发
	attributeRemoved		删除属性时触发
ServletRequestAttributeListener	attributeAdded	ServletRequestAttributeEvent	添加属性时触发
	attributeReplaced		替换属性时触发
	attributeRemoved		删除属性时触发

注：这三个监听器接口分别是 ServletContextAttributeListener，HttpSessionAttributeListener，ServletRequestAttributeListener，这三个接口中同一个事件在这三个接口中对应的方法名称完全相同，只是接受的参数类型不同。

2. ServletContextAttributeListener 监听器范例：

（1）编写 ServletContextAttributeListener 监听器监听 ServletContext 域对象的属性值变化情况，代码如下：

```
public class MyServletContextAttributeListener
        implements ServletContextAttributeListener {
public void attributeAdded(ServletContextAttributeEvent scab) {
            String str =MessageFormat.format(
                        "ServletContext 域对象中添加了属性:{0}，属性值是:{1}"
                        ,scab.getName()
                       ,scab.getValue());
            System.out.println(str);
      }
public void attributeRemoved(ServletContextAttributeEvent scab) {
            String str =MessageFormat.format(
                        "ServletContext 域对象中删除属性:{0}，属性值是:{1}"
                        ,scab.getName()
                        ,scab.getValue());
            System.out.println(str);
}
public void attributeReplaced(ServletContextAttributeEvent scab) {
            String str =MessageFormat.format(
                        "ServletContext 域对象中替换了属性:{0}的值"
                         ,scab.getName());
             System.out.println(str);
}
}
```

（2）在 web.xml 文件中注册监听器

```
<listener>
<listener-class>me.gacl.web.listener.MyServletContextAttributeListener</listener-class>
</listener>
```

（3）编写 ServletContextAttributeListenerTest.jsp 测试页面

```
<% //往 application 域对象中添加属性
```

```
    application.setAttribute("name", "孤傲苍狼");
    //替换 application 域对象中 name 属性的值
    application.setAttribute("name", "gacl");
    //移除 application 域对象中 name 属性
    application.removeAttribute("name");
%>
```

3. ServletRequestAttributeListener 和 HttpSessionAttributeListener 监听器范例：

（1）编写监听器监听 HttpSession 和 HttpServletRequest 域对象的属性值变化情况，代码如下：

```
public class MyRequestAndSessionAttributeListener implements
            HttpSessionAttributeListener, ServletRequestAttributeListener {
public void attributeAdded(ServletRequestAttributeEvent srae) {
        String str =MessageFormat.format(
                "ServletRequest 域对象中添加了属性:{0}，属性值是:{1}"
                ,srae.getName()
                ,srae.getValue());
        System.out.println(str);
    }
public void attributeRemoved(ServletRequestAttributeEvent srae) {
        String str =MessageFormat.format(
                "ServletRequest 域对象中删除属性:{0}，属性值是:{1}"
                ,srae.getName()
                ,srae.getValue());
        System.out.println(str);
    }
public void attributeReplaced(ServletRequestAttributeEvent srae) {
        String str =MessageFormat.format(
                "ServletRequest 域对象中替换了属性:{0}的值"
                ,srae.getName());
```

```
            System.out.println(str);
        }
public void attributeAdded(HttpSessionBindingEvent se) {
            String str =MessageFormat.format(
                    "HttpSession 域对象中添加了属性:{0}，属性值是:{1}"
                     ,se.getName()
                      ,se.getValue());
            System.out.println(str);
        }
public void attributeRemoved(HttpSessionBindingEvent se) {
            String str =MessageFormat.format(
                      "HttpSession 域对象中删除属性:{0}，属性值是:{1}"
                     ,se.getName()
                     ,se.getValue());
            System.out.println(str);
        }
public void attributeReplaced(HttpSessionBindingEvent se) {
            String str =MessageFormat.format(
                      "HttpSession 域对象中替换了属性:{0}的值"
                      ,se.getName());
           System.out.println(str);
      }
 }
```

（2）编写 RequestAndSessionAttributeListenerTest.jsp 测试页面

```
<%
//往 session 域对象中添加属性
session.setAttribute("aa", "bb");
//替换 session 域对象中 aa 属性的值
session.setAttribute("aa", "xx");
```

```
//移除 session 域对象中 aa 属性
session.removeAttribute("aa");
//往 request 域对象中添加属性
request.setAttribute("aa", "bb");
//替换 request 域对象中 aa 属性的值
request.setAttribute("aa", "xx");
//移除 request 域对象中 aa 属性
request.removeAttribute("aa");
%>
```

11.4 监听器实例

1.实例：统计当前在线人数。

说明：运行结果后同时打开三个浏览器查看结果。

11.5 Web3.0 注解

1. 建立一个带注解的 Servlet

```
@WebServlet(
        urlPatterns = { "/aaa" },
        initParams = {
                @WebInitParam(name = "x", value = "1"),
                @WebInitParam(name = "y", value = "2")
})
```

2. 建立一个带注解的 Filter

```
@WebFilter(
        urlPatterns = { "/*" },
        initParams = {
                @WebInitParam(name = "x", value = "1"),
                @WebInitParam(name = "y", value = "2")
        })
```

3. 建立一个带注解的 Listener

@WebListener

public class MyListener implements ServletContextListener,

ServletContextAttributeListener, HttpSessionListener, HttpSessionAttributeListener,

HttpSessionActivationListener, HttpSessionBindingListener, ServletRequestListener,

ServletRequestAttributeListener{}