java.net

Java程序设计

第02-02讲 监听器（Listener）

Java课程组

本讲教学目标

- 理解监听器的作用
- 掌握如何编写监听器程序

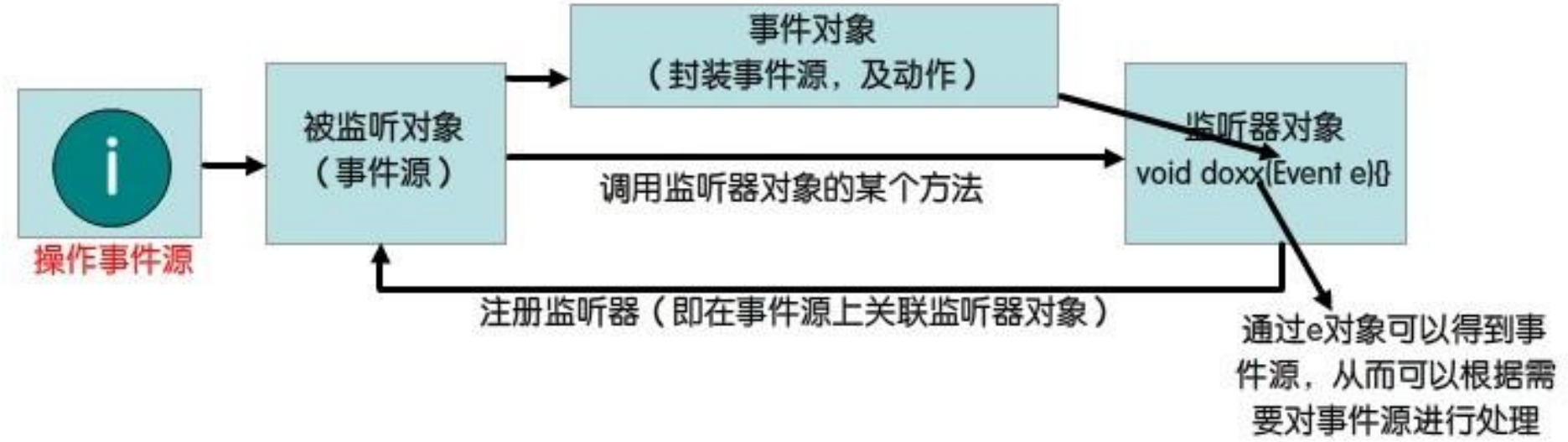
知识回顾/本讲先行知识

- 会话
  - 了解会话管理的基本原理
  - 掌握会话管理的四种技术实现

本讲内容

- 监听器(Linstener)简介
- 与ServletContext相关监听器
- 与HttpSession相关监听器
- 与ServletRequest相关监听器

监听器(Listener)简介



- 监听器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另一个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，监听器某个方法立即被执行

Listener接口与事件对应表

分类	Listener接口	Event类
与ServletContext有关	ServletContextListener	ServletContextEvent
	ServletContextAttributeListener	ServletContextAttributeEvent
与HttpSession有关	HttpSessionListener	HttpSessionEvent
	HttpSessionActivationListener	
	HttpSessionAttributeListener	HttpSessionBindingEvent
	HttpSessionBindingListener	
与ServletRequest有关	ServletRequestListener	ServletRequestEvent
	ServletRequestAttributeListener	ServletRequestAttributeEvent

编写监听器的步骤

- 编写实现类
- 在web.xml中进行部署
- 编写测试页面

本讲内容

- 监听器(Linstener)简介
- 与ServletContext相关监听器
- 与HttpSession相关监听器
- 与ServletRequest相关监听器

全局对象：application范围对象

- 单个Web站点的资源都**共享一个**javax.servlet.ServletContext类的实体。通过该对象可以存取应用程序的全局对象以及初始化阶段的变量
- 全局对象即为Application范围对象，其生命周期从容器启动至容器关闭。初始阶段的变量是指在web.xml中，由<context-param>元素设定的变量，该变量的范围是Application范围

ServletContextListener接口

- 实现了该接口的程序，当JavaWeb应用程序启动时，会自动开始监听工作
- 首先调用contextInitialized()方法接收对应的ServletContextEvent事件
- 当应用从容器中移除时，会自动调用contextDestroyed()方法
- 以上两个方法都会接收到ServletContextEvent事件对象，该对象可以调用getServletContext()方法取得ServletContext对象(全局对象)

ServletContextListener

方法	说明
contextInitialized(ServletContextEvent e)	通知正在收听的对象，应用程序已经被加载及初始化
contextDestroyed(ServletContextEvent e)	通知正在收听的对象，应用程序已经被载出

- ServletContextEvent的主要方法：
  - getServletContext()

ServletContextAttributeListener接口

- 实现该接口的程序，能够监听Application范围的变化，例如：当有对象加入全局范围时，该程序会被调用

- 示例：

ServletContextAttributeListener接口

方法	说明
attributeAdded(ServletContextAttributeEvent e)	若有对象加入Application范围时，通知正在收听的对象
attributeReplaced(ServletContextAttributeEvent e)	若在Application的范围，有对象取代另一个对象，通知正在收听的对象
attributeRemoved(ServletContextAttributeEvent e)	若有对象从Application范围移出时，通知正在收听的对象

- ServletContextAttributeEvent的主要方法
  - getName()
  - getValue()
    - attributeReplaced()方法中，getName()与getValue()是取之前的值

本讲内容

- 监听器(Linstener)简介
- 与ServletContext相关监听器
- 与HttpSession相关监听器
- 与ServletRequest相关监听器

HttpSessionListener

- HttpSessionListener监听Session对象的创建与销毁，当有Session对象产生或销毁时，会自动调用sessionCreated()或sessionDestroyed()两个方法

HttpSessionEvent事件

- HttpSessionListener接口与HttpSessionActivationListener接口都使用HttpSessionEvent事件对象
- HttpSessionEvent类主要的方法：
  - getSession()

HttpSessionActivationListener接口

- 该接口主要用于：同一个Session转移到不同JVM的情形(如：负载均衡，这些JVM可以在同一台机器或分散在网络中的多台机器)
- 当Session被储存起来，并且等待转移至另一个JVM，这段时间称为失效状态(Passivate)，若Session中的属性对象实现HttpSessionActivationListener接口时，Container会自动调用sessionWillPassivate()方法通知该对象的Session已变成失效状态
- 当Session被转移至其他JVM之后，它又成为有效状态(Activate)，此时Container会自动调用sessionDidActivate()方法通知该对象的Session已变成有效状态

接口方法列表

方法	说明
HttpSessionListener接口	
sessionCreated(HttpSessionEvent e)	通知正在收听的对象，Session已经被加载及初始化
sessionDestroyed(HttpSessionEvent e)	通知正在收听的对象， Session已经被载出
HttpSessionActivationListener接口	
sessionDidActivate(HttpSessionEvent e)	通知正在收听的对象，它的Session已经被变为有效状态
sessionWillPassivate(HttpSessionEvent e)	通知正在收听的对象，它的Session已经被变为无效状态

HttpSessionAttributeListener

- HttpSessionAttributeListener会监听Session范围的变化，功能与ServletContextAttributeListener接口类似，包含三个方法
  - attributeAdded()
  - attributeReplaced()
  - attributeRemove()

HttpSessionBindingEvent事件

- HttpSessionBindingEvent事件主要有三个方法
  - getName()
  - getSession()
  - getValue()

HttpSessionBindingListener

- 实现HttpSessionBindingListener接口后，只要有对象加入Session范围或从Session范围中移除时，容器会分别自动调用下面两个方法：
  - valueBound(HttpSessionBindingEvent e)
  - valueUnbound(HttpSessionBindingEvent e)
- HttpSessionBindingListener接口是唯一不需要在web.xml中设定的Listener

HttpSessionBindingListener

- 自定义实现HttpSessionBindingListener接口的类
- 实例化监听器类的对象
- 将该对象添加到Session中

```
protected void doGet(HttpServletRequest request, HttpServletResponse r
    HttpSession session = request.getSession();
    SessionBindingTest bindingTestListener = new SessionBindingTest();
    session.setAttribute("testBinding", bindingTestListener);
    PrintWriter out = response.getWriter();
    out.println("添加SessionBindingTest对象");

    out.println("<P>"+ "修改Session参数</P>");
    session.setAttribute("sessionParam", "sessionParamNew");

    out.println("<P>"+ "删除Session参数</P>");
    session.removeAttribute("testBinding");
```

HttpSessionBindingListener

- HttpSessionAttributeListener使用的事件与HttpSessionBindingListener使用的事件相同： HttpSessionBindingEvent
- HttpSessionAttributeListener与HttpSessionBindingListener的不同在于：
  - 前者监听Web站点所有Session范围的变化
  - 后者只监听Session范围内实现了HttpSessionBindingListener接口的对象的创建与销毁

接口方法列表

方法	说明
HttpSessionBindingListener接口	
valueBound(HttpSessionBindingEvent e)	当实现HttpSessionBindingListener的对象加入session时，会调用该方法
valueUnbound(HttpSessionBindingEvent e)	当实现HttpSessionBindingListener的对象在session中销毁时，调用该方法
HttpSessionAttributeListener接口	
attributeAdded(HttpSessionBindingEvent e)	若有对象加入Session范围时，通知正在收听的对象
attributeReplaced(HttpSessionBindingEvent e)	若在Session的范围，有对象取代另一个对象，通知正在收听的对象
attributeRemoved(HttpSessionBindingEvent e)	若有对象从Session范围移出时，通知正在收听的对象

本讲内容

- 监听器(Linstener)简介
- 与ServletContext相关监听器
- 与HttpSession相关监听器
- 与ServletRequest相关监听器

ServletRequestListener接口

- 当有请求产生或销毁，会自动调用该接口实现的requestInitialized()或requestDestroyed()方法
- 该接口使用ServletRequestEvent事件

ServletRequestListener

方法	说明
requestInitialized(ServletRequestEvent e)	通知正在收听的对象，ServletRequest已经被加载及初始化
requestDestroyed(ServletRequestEvent e)	通知正在收听的对象，ServletRequest已经被载出

- ServletRequestEvent的主要方法：
  - getServletContext()
  - getServletRequest()

ServletRequestAttributeListener

- 该接口监听Request范围的变化，有三个主要方法：
  - attributeAdded()
  - attributeReplaced()
  - attributeRemoved()
- 使用ServletRequestAttributeEvent事件

ServletRequestAttributeListener

方法	说明
attributeAdded(ServletRequestAttributeEvent e)	若有对象加入Request范围时，通知正在收听的对象
attributeReplaced(ServletRequestAttributeEvent e)	若在Request的范围，有对象取代另一个对象，通知正在收听的对象
attributeRemoved(ServletRequestAttributeEvent e)	若有对象从Request范围移出时，通知正在收听的对象

- ServletRequestAttributeEvent主要方法
  - getName()
  - getValue()

监听器的应用

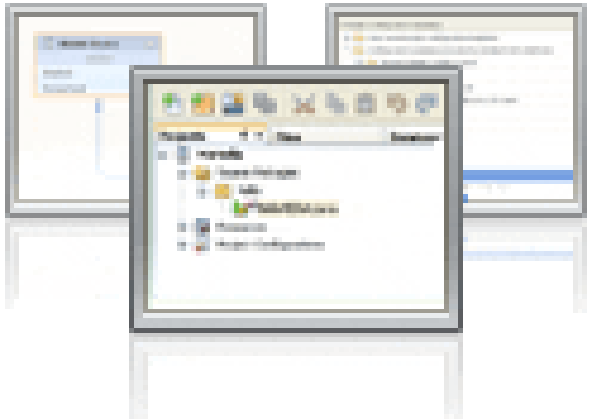
- ServletContext范围的监听器可以进行一些初始化的动作，如：当Web应用启动的时候进行全局配置
- Session范围的监听器对一个会话过程(与客户端关联)中所产生的事件进行响应，可以对客户端信息的变化进行跟踪
- Request范围的监听器可以监听用户的每次请求

小结

- JavaWeb应用中的监听器与事件
- 监听器与事件的编写、使用
- 理解监听器的应用场景

实验

- 实验



本讲结束

- 谢谢大家