# Listener(监听器)

## 简介

* Listener用于监听JavaWeb程序中的事件。
* 例如：ServletContext、HttpSession、ServletRequest的创建、修改和删除。
* Listener和我们之前学习的JS中的事件处理机制类似。在JS中，当我们触发某个事件后（例如：点击一个按钮）程序会调用一个响应函数来处理事件。同样的，在JavaWeb中，我们可以为某些事件来设置监听器，当事件被触发时，监听器中的指定方法将会被调用。

## 观察者模式

* Listener的原理是基于观察者模式的，所谓观察者模式简单来说，就是当被观察者的特定事件被触发（一般这某些方法被调用）后，会通知观察者（调用观察者的方法），观察者可以在自己的方法中来对事件做一些处理。
* 在我们的JavaWeb中，观察者就是Listener，而被观察者可能有三个ServletContext、HttpSession、ServletRequest。而事件指的就是这些对象的创建、修改和删除等。

## 监听器的分类

### 监听对象的创建与销毁

* ServletContextListener
  + 作用：监听ServletContext对象的创建与销毁
  + 方法：
    - public void contextInitialized ( ServletContextEvent sce );
      * ServletContext创建时调用
    - public void contextDestroyed ( ServletContextEvent sce );
      * ServletContext销毁时调用
  + ServletContextEvent对象
    - 作用：public ServletContext getServletContext ()
      * 获取ServletContext对象
* HttpSessionListener
  + 作用：监听HttpSession对象的创建与销毁
  + 方法：
    - public void sessionCreated ( HttpSessionEvent se );
      * HttpSession对象创建时调用
    - public void sessionDestroyed ( HttpSessionEvent se );
      * HttpSession对象销毁时调用
    - HttpSessionEvent对象
      * 作用：public HttpSession getSession ()
        + 获取当前HttpSession对象
* ServletRequestListener
  + 作用：监听ServletRequest对象的创建与销毁
  + 方法：
    - public void requestInitialized ( ServletRequestEvent sre );
      * ServletRequest对象创建时调用
    - public void requestDestroyed ( ServletRequestEvent sre );
      * ServletRequest对象销毁时调用
    - ServletRequestEvent对象
      * 作用：
        + public ServletRequest getServletRequest ();

获取当前的ServletRequest对象。

* + - * + public ServletContext getServletContext ();

获取当前项目的ServletContext对象。

#### 创建与销毁监听器的使用：

* 三种创建与销毁的监听器使用起来基本一致。
* 下边来编写一个ServletContext的监听器：
  + 步骤：
    - 创建一个类实现ServletContextListener

|  |
| --- |
| **public** **class** MyServletContextListener **implements** ServletContextListener {  @Override  **public** **void** contextInitialized(ServletContextEvent sce) {  System.*out*.println("哈哈，我是ServletContext，我出生了");  }  @Override  **public** **void** contextDestroyed(ServletContextEvent sce) {  System.*out*.println("~~~~(>\_<)~~~~，我是ServletContext，我要死了");  }  } |

* + - 在web.xml文件中注册监听器

|  |
| --- |
| <listener>  <listener-class>com.atguigu.web.listener.MyServletContextListener</listener-class>  </listener> |

* 由于ServletContext对象在服务器启动时创建，停止时销毁。所以启动服务器时我们会发现contextInitialized()方法被调用，服务器停止时contextDestroyed()方法被调用。
* 其他两个监听器和该监听器使用方法一样，不再多说。

### 2. 监听对象的属性变化

* ServletContextAttributeListener
  + 作用：监听ServletContext中属性的创建、修改和销毁
  + 方法：
    - public void attributeAdded(ServletContextAttributeEvent scab);
      * 向ServletContext中添加属性时调用
    - public void attributeRemoved(ServletContextAttributeEvent scab);
      * 从ServletContext中移除属性时调用
    - public void attributeReplaced(ServletContextAttributeEvent scab);
      * 当ServletContext中的属性被修改时调用
  + ServletContextAttributeEvent对象
    - 作用：
      * public String getName()
        + 获取修改或添加的属性名
      * public Object getValue()
        + 获取被修改或添加的属性值
      * public ServletContext getServletContext ()
        + 获取当前WEB应用的ServletContext对象
* HttpSessionAttributeListener
  + 作用：监听HttpSession中属性的创建、修改和销毁
  + 方法：
    - public void attributeAdded ( HttpSessionBindingEvent se );
      * 向HttpSession中添加属性时调用
    - public void attributeRemoved(HttpSessionBindingEvent se);
      * 从HttpSession中移除属性时调用
    - public void attributeReplaced(HttpSessionBindingEvent se);
      * 当HttpSession中的属性被修改时调用
  + HttpSessionBindingEvent对象
    - 作用：
      * public String getName()
        + 获取修改或添加的属性名
      * public Object getValue()
        + 获取被修改或添加的属性值
      * public HttpSession getSession ()
        + 获取当前的HttpSession对象
* ServletRequestAttributeListener
  + 作用：监听ServletRequest中属性的创建、修改和销毁
  + 方法：
    - public void attributeAdded (ServletRequestAttributeEvent srae );
      * 向ServletRequest中添加属性时调用
    - public void attributeRemoved(ServletRequestAttributeEvent srae);
      * 从ServletRequest中移除属性时调用
    - public void attributeReplaced(ServletRequestAttributeEvent srae);
      * 当ServletRequest中的属性被修改时调用
  + ServletRequestAttributeEvent对象
    - 作用：
      * public String getName()
        + 获取修改或添加的属性名
      * public Object getValue()
        + 获取被修改或添加的属性值
      * public ServletRequest getServletRequest ()
        + 获取当前的ServletRequest对象

#### 对象属性变化监听器的使用：

* 同样三种对象属性变化监听器使用方式类似，下边以request属性监听器为例。
  + 创建一个类实现ServletRequestAttributeListener接口

|  |
| --- |
| **public** **class** ReqAttrListener **implements** ServletRequestAttributeListener {  @Override  **public** **void** attributeAdded(ServletRequestAttributeEvent srae) {  System.*out*.println("request域中添加一个属性"+srae.getName()+"="+srae.getValue());  }  @Override  **public** **void** attributeRemoved(ServletRequestAttributeEvent srae) {  System.*out*.println("request域中移除一个属性"+srae.getName()+"="+srae.getValue());  }  @Override  **public** **void** attributeReplaced(ServletRequestAttributeEvent srae) {  System.*out*.println("request域中一个属性被修改了"+srae.getName()+"="+srae.getValue());  }  } |

* + 在web.xml文件中注册监听器

|  |
| --- |
| <listener>  <listener-class>com.atguigu.web.listener.ReqAttrListener</listener-class>  </listener> |

* + 如此当我们操作request域中的属性时，对应方法将会被调用。

### 3. 监听Session内的对象

* HttpSessionBindingListener
  + 作用：监听某个对象在session域中的创建与移除。
  + 方法：
    - public void valueBound(HttpSessionBindingEvent event)
      * 该类的实例被放到Session域中时调用
    - public void valueUnbound(HttpSessionBindingEvent event)
      * 该类的实例从Session中移除时调用
  + HttpSessionBindingEvent对象
    - 作用：
      * public HttpSession getSession ()
        + 获取HttpSession对象
      * public String getName()
        + 获取操作的属性名
      * public Object getValue()
        + 获取操作的属性值
* 使用：要监听哪一个类，直接使该类实现HttpSessionBindingListener接口即可。

|  |
| --- |
| **public** **class** Student **implements** HttpSessionBindingListener {  @Override  **public** **void** valueBound(HttpSessionBindingEvent event) {  //doSomeThing  }  @Override  **public** **void** valueUnbound(HttpSessionBindingEvent event) {  //doSomeThing  }  } |

* HttpSessionActivationListener
  + 作用：监听某个对象在session中的序列化与反序列化。
  + 方法：
    - public void sessionWillPassivate(HttpSessionEvent se)
      * 该类实例和Session一起钝化到硬盘时调用
    - public void sessionDidActivate(HttpSessionEvent se)
      * 该类实例和Session一起活化到内存时调用
  + HttpSessionEvent对象
    - 作用：
      * public HttpSession getSession ()
        + 获取HttpSession对象
* 使用：要监听哪一个类，直接使该类实现HttpSessionActivationListener接口即可。

|  |
| --- |
| **public** **class** Student **implements** HttpSessionActivationListener , Serializable {  **private** **static** **final** **long** *serialVersionUID* = 1L;  @Override  **public** **void** sessionWillPassivate(HttpSessionEvent se) {  // **TODO** Auto-generated method stub  }  @Override  **public** **void** sessionDidActivate(HttpSessionEvent se) {  // **TODO** Auto-generated method stub  }  } |

* + 注意：这里为了是Student对象可以正常序列化到硬盘上，还需要让类实现java.io.Serializable接口