02-02\_监听器

# 授课内容

授课科目：JavaEE

授课内容：监听器

授课类型：讲授+实践

授课时间：3+1学时

主讲教师：解卫静

# 教学目标要求

## 能力目标：

1. 培养学生掌握编写监听器程序的技能，并能够使用监听器解决实际问题；

## 知识目标：

1. 培养学生理解监听器的作用；
2. 培养学生掌握ServletContext相关监听器、HttpSession相关监听器、ServletRequest相关监听器的使用过程和配置方法。

# 课件分析

概 述：首先阐述监听器的作用，介绍监听接口、事件，并讲解编写监听器的步骤，重点介绍与ServletContext、HttpSession、ServletRequest相关监听器的使用原理和编程实现。

教学重点：监听器的编程实现。

教学难点：监听器的编程实现。

# 教学方法

讲授法、案例法、实践教学法、类比法等。

# 教学过程

## 课前补充

（0分钟）

## 上节回顾

（2分钟）

上一章介绍了会话管理的实现过程，可以大致描述为：当服务器接收到客户端浏览器的首次请求时，服务器会初始化一个会话并分配给该会话一个标识符，服务器将此标识符响应给客户端浏览器，在以后的请求中，客户端浏览器将此标识符包含在请求中，服务器根据此标识符将请求与对应的会话联系起来。还讲解了会话管理的技术实现，包括隐藏域、Cookie、URL重写和Session。

## 作业点评

（0分钟）

## 导入新课

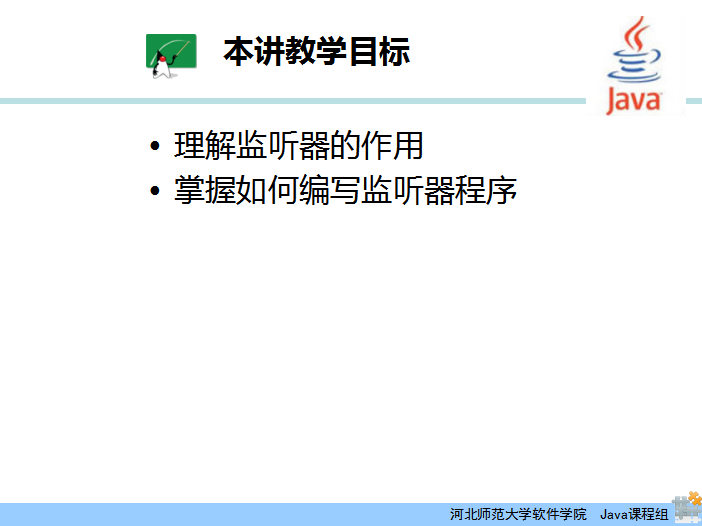
（2分钟）

在使用Session实现用户登录功能时提到，当客户端浏览器关闭时，存储Session ID的Cookie消失，与之对应的Session对象不会再使用但仍然存在，在销毁之前它占的内存就浪费了。如果要避免这种浪费，可以对客户端浏览器进行监听，从而实现当浏览器关闭时销毁对应的Session对象。

另外，大家熟悉的通讯软件QQ，曾经实时在线人数突破过两个亿，游戏里也经常统计当前在线玩家数量，还有各种网站，会显示网站的访问量是多少等等，这些都会用到本章要讲的监听器。

## 讲授新课

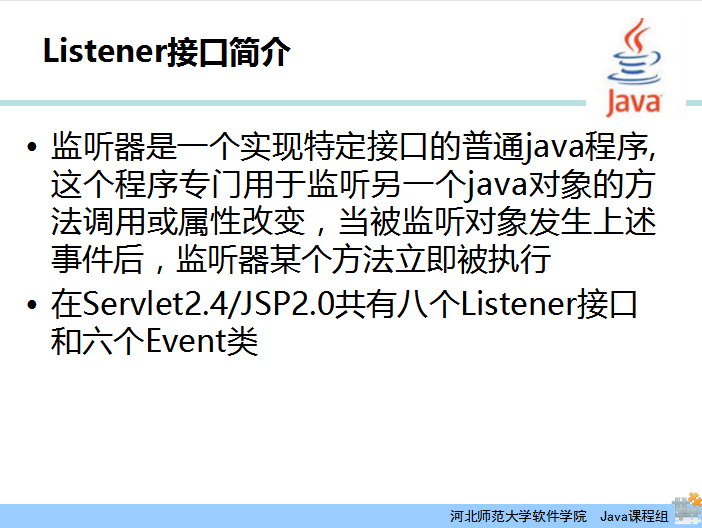
（120分钟）



本章的教学目标是理解ServletContext相关监听器、Session相关监听器、Request相关监听器的作用，掌握监听接口中方法的含义及使用方法，重点掌握这三类监听器的编程实现。



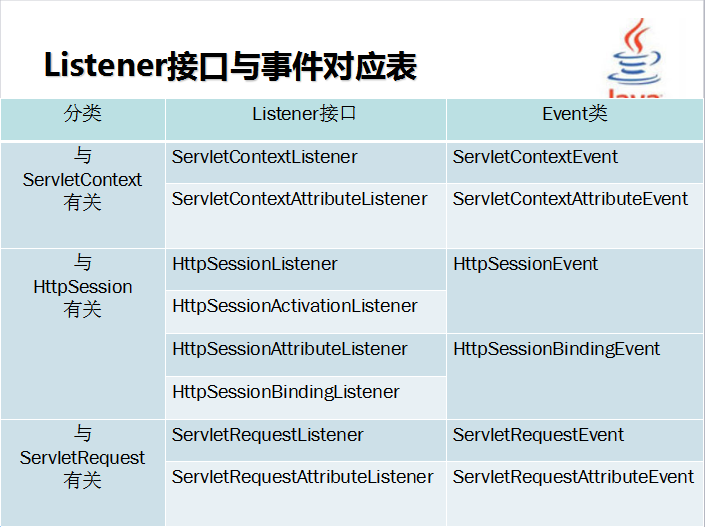
首先对监听器进行简单介绍。



监听器，又称Listener，专门用于监听其他对象的事件发生或状态改变，当被监听的对象创建、销毁或属性发生改变时，监听器会立即采取相应的行动。监听器其实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。

JavaWeb中的监听器是Servlet规范中一种实现特定的Lisener接口的类，用于监听web应用程序中的ServletContext,Session和Request对象的创建和销毁事件，以及监听这些对象的属性发生修改的事件。

在Servlet规范中共有八个Listener接口和六个Event类。



实现这些接口的监听器监听的对象分别为ServletContext，Session和Request，根据监听对象的不同，把监听器划分为三种类型：

1.监听对象自身的创建和销毁的事件监听器。例如:

ServletContextListener监听ServletContext的创建和销毁事件；

HttpSessionListener监听Session的创建和销毁事件；

ServletRequestListener监听Request的创建和销毁事件。

2.监听对象中的属性的增加和删除的事件监听器。例如：

ServletContextAttributeListener监听ServletContext中属性的增加和删除事件；

HttpSessionAttributeListener监听Session中属性的增加和删除事件；

ServletRequestAttributeListener监听Request中属性的增加和删除事件。

3.监听绑定到Session范围中的某个对象的状态的事件监听器。例如：

HttpSessionActivationListener监听Session中属性的失效状态与有效状态；

HttpSessionBindingListener监听被绑定的Session对象的创建和销毁事件。

与接口对应的Event类，用来封装被监听的对象。例如：

ServletContextListener接口依赖的ServletContextEvent类，封装的是ServletContext对象；

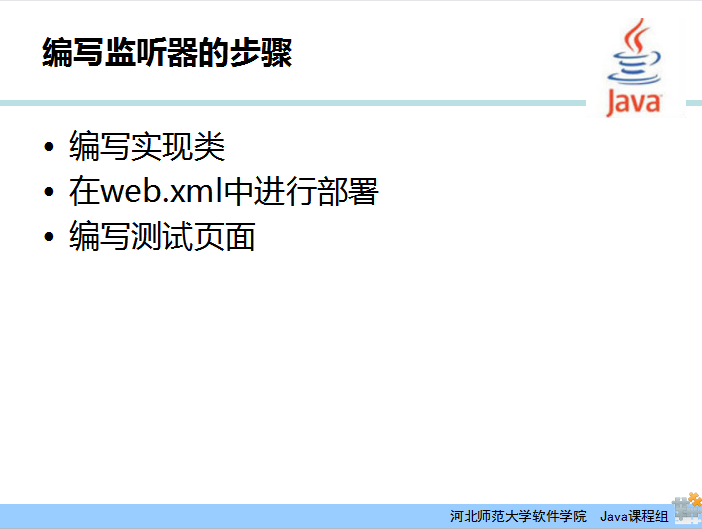
HttpSessionListener接口和HttpSessionActivationListener接口依赖的HttpSessionEvent类，封装的是Session对象；

ServletRequestListener接口依赖的ServletRequestEvent类，封装的是Request对象；

ServletContextAttributeListener接口依赖的ServletContextAttributeEvent类，封装的是ServletContext中的属性；

HttpSessionAttributeListener接口和HttpSessionBindingListener接口依赖的HttpSessionBindingEvent类，封装的是Session中的属性；

ServletRequestAttributeListener接口依赖的ServletRequestAttributeEvent类，封装的是Request中的属性。



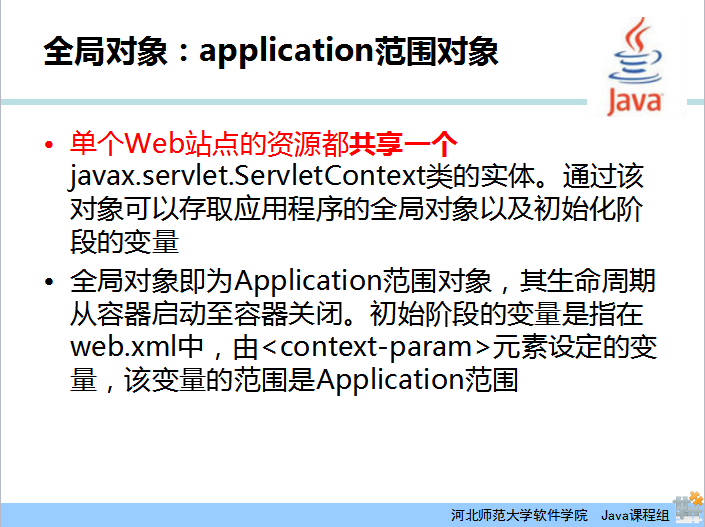
编写一个监听器，首先要编写它的实现类，这个类必须实现某个监听接口。

然后在web.xml中进行部署，即注册监听器。因为要想监听某个对象，那么必须将监听器注册到这个对象上才能够实现对它的行为动作进行监听，在JavaWeb中，监听器的注册是在web.xml文件中进行配置或使用注解。

最后编写一个测试页面，对监听器进行测试。



下面详细介绍第一类监听器，与ServletContext相关的监听器。



介绍与ServletContext相关监听器之前，先回顾一下ServletContext对象。ServletContext是一个Application范围的全局对象，每个Web应用启动后，Servlet容器都会创建唯一的一个ServletContext类的对象，直到Servlet容器关闭时消失。所以单个Web站点的资源都共享一个ServletContext对象，该对象包含整个Web程序的信息，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。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接口用于监听ServletContext对象的创建和销毁事件。如果一个类实现了ServletContextListener接口，那么当JavaWeb应用程序启动时，会执行此监听器中contextInitialized()方法，当应用从Servlet容器中移除时，会自动调用监听器的contextDestroyed()方法。实现ServletContextListener接口的监听器的这两个方法，都会接收ServletContextEvent事件作为参数，在这里就是ServletContext的创建或销毁事件，可以通过该事件的getServletContext()方法获取ServletContext对象，从而实现对ServletContext对象的监听。



ServletContextListener接口中contextInitialized()方法的作用是通知正在收听的对象，应用程序已经被加载及初始化，contextDestroyed()方法的作用是通知正在收听的对象，应用程序已经被载出。也就是，应用程序被加载及初始化时，ServletContext被创建，同时监听器执行contextInitialized()方法；应用程序被载出后，ServletContext被销毁，执行监听器中的contextDestroyed()方法。两个方法接收到的ServletContextEvent对象中的主要方法就是getServletContext()，用来获取ServletContext对象。

示例：编写一个ServletContextListenerDemo类，实现ServletContextListener接口，监听ServletContext对象的创建和销毁。

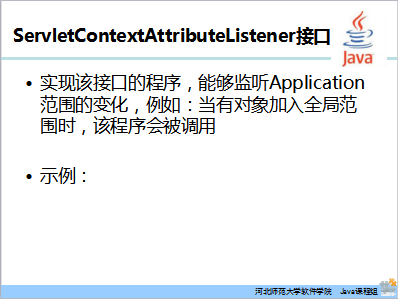
1.编写监听器，部分代码如下：

|  |
| --- |
| public class ServletContextListenerDemo implements ServletContextListener {  public void contextInitialized(ServletContextEvent arg0) {  System.out.println("当Web应用启动时会执行该方法，我们可以在这里做一些初始化工作");  //向Application范围添加两个全局变量，分别统计在线人数和在线会员数  arg0.getServletContext().setAttribute("onlineNum", 0);  arg0.getServletContext().setAttribute("onlineMemberNum", 0);  }  public void contextDestroyed(ServletContextEvent arg0) {  System.out.println("当Web应用停止时会执行该方法，我们可以在这里做一些清理工作");  }  } |

2.在web.xml文件中注册监听器，代码如下：

|  |
| --- |
| <listener>  <listener-class>onest.listener.ServletContextListenerDemo</listener-class>  </listener> |

经过以上两个步骤，便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ServletContextListenerDemo监听器就可以监听ServletContext对象了。



ServletContextAttributeListener接口，用于监听ServletContext属性的添加、移除和替换，即监听Application范围的变化，例如：当有对象加入全局范围时，实现了该接口的监听器会被调用。

示例：编写一个ServletContextAttributeListenerDemo类，实现ServletContextAttributeListener接口，监听ServletContext对象中属性的添加和移除。

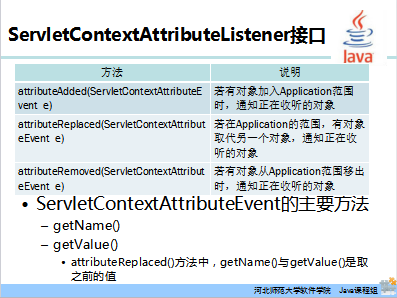
1.编写监听器，部分代码如下：

|  |
| --- |
| public class ServletContextAttributeLinstenerDemo implements ServletContextAttributeListener {  public void attributeAdded(ServletContextAttributeEvent scab) {  System.out.println("当ServletContext属性增加时会执行该方法");  }  public void attributeRemoved(ServletContextAttributeEvent scab) {  System.out.println("当ServletContext属性移除时会执行该方法");  }  public void attributeReplaced(ServletContextAttributeEvent scab) {  System.out.println("当ServletContext属性改变时会执行该方法");  }  } |

2.在web.xml文件中注册监听器，代码如下：

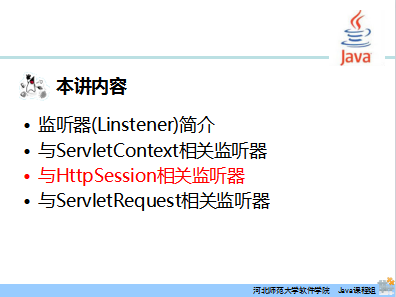
|  |
| --- |
| <listener>  <listener-class>onest.listener.ServletContextAttributeLinstenerDemo</listener-class>  </listener> |

经过以上两个步骤，就可以实现服务器在启动时，自动把在web.xml中配置的监听器注册到ServletContext对象上，并且ServletContextAttributeLinstenerDemo监听器对ServletContext对象中的属性进行监听。

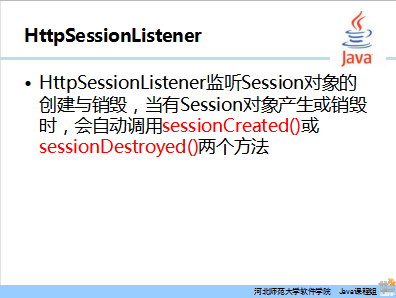


ServletContextAttributeListener接口中attributeAdded()方法的作用是，当有对象加入Application范围时，通知正在收听的对象，也就是ServletContext的setAttribute()方法被调用时，会触发attributeAdded()方法执行；attributeReplaced()方法的作用是，若在Application的范围，有对象取代另一个对象，通知正在收听的对象，也就是ServletContext的setAttribute()方法设置的属性名存在时，这个属性值被替换，那么attributeReplaced()方法中的代码被执行;attributeRemoved()方法的作用是，若有对象从Application范围移出，通知正在收听的对象，即ServletContext的removeAttribute()方法被调用时，执行attributeRemoved()方法中的代码。

以上的三个方法都需要传入参数ServletContextAttributeEvent对象，ServletContextAttributeEvent类的主要方法有getName()和getValue()，getName()可以获取ServletContext对象中参数的名称,getValue()可以获取ServletContext对象中参数的值。在attributeReplaced()方法中，getValue()获取到的是被替换前的参数值。



第二类监听器，与Session相关的监听器。



HttpSessionListener接口用于监听Session对象的创建和销毁。创建一个Session时，激发sessionCreated()方法，销毁一个Session时，激发sessionDestroyed()方法。这两个方法都需要传入参数HttpSessionEvent对象，HttpSessionEvent类中主要的方法是getSession()，用来获取Session对象。

示例：编写一个HttpSessionListenerDemo类，实现HttpSessionListener接口，监听Session对象的创建和销毁。要求Session对象创建时，在线人数增加，Session对象销毁时，在线人数减少。

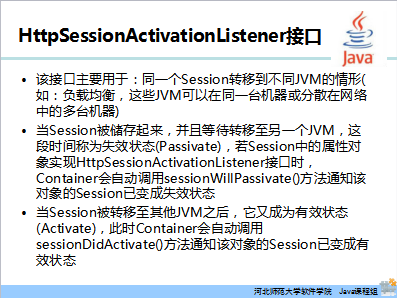
1.编写监听器，部分代码如下：

|  |
| --- |
| public void sessionCreated(HttpSessionEvent arg0) {  int onlineNum = (Integer)arg0.getSession().getServletContext().getAttribute("onlineNum");  arg0.getSession().getServletContext().setAttribute("onlineNum",++onlineNum);  } |

2.在web.xml文件中注册监听器，代码如下：

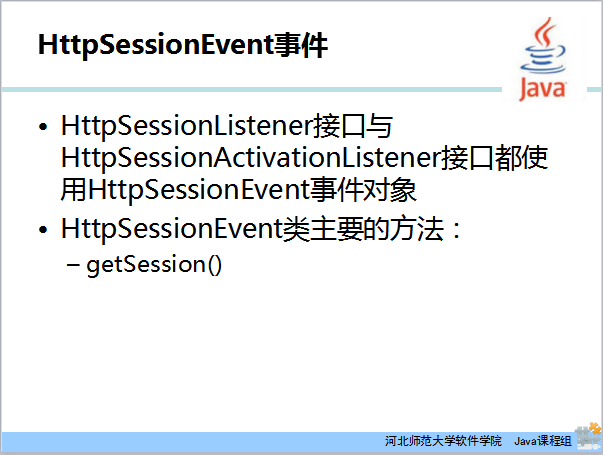
|  |
| --- |
| <listener>  <listener-class>onest.listener.HttpSessionLinstenerDemo</listener-class>  </listener>  <!-- 配置HttpSession对象的销毁时机 -->  <session-config>  <!--配置HttpSession对象的1分钟之后销毁 -->  <session-timeout>1</session-timeout>  </session-config> |

访问Servlet时，Session对象就会创建，sessionCreated()方法中的代码被执行，在线人数增加一。

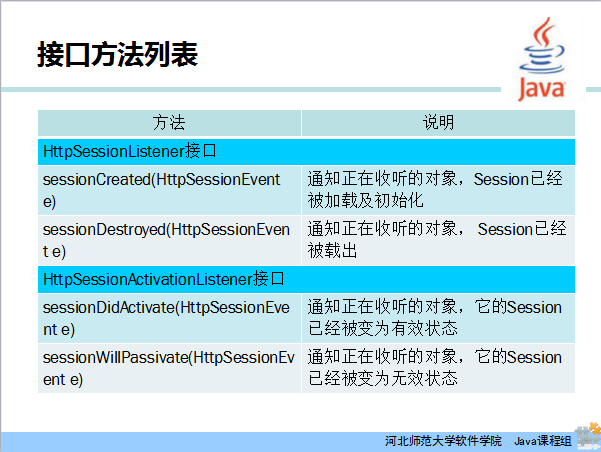


与HttpSessionListener接口依赖同一个Event类的HttpSessionActivationListener接口主要用于在分布式环境中同一个Session转移到不同JVM的情形。比如在同一台计算机中有多个JVM，或这些JVM分散在网络中的多台计算机，当Session在这些JVM中转移时，会用到HttpSessionActivationListener接口。

这个接口主要有两个方法，sessionWillPassivate()方法和sessionDidActivate()方法。将Session转移至另一个JVM时，必须将Session对象在原本的JVM中的所有属性对象序列化，这时Session就变为失效状态(Passivate)，若在这之前Session属性对象实现了HttpSessionActivationListener接口，那么Servlet容器会自动调用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，转为有效状态(Activate)，此时Servlet容器会自动调用sessionDidActivate()方法通知该对象的Session已变成有效状态。



HttpSessionListener接口与HttpSessionActivationListener接口都使用HttpSessionEvent对象作为方法的参数，前者监听Session对象的创建和销毁，而后者监听Session对象的有效或无效状态，这两个接口监听的是Session对象本身，所以它们依赖的HttpSessionEvent类中主要的方法是getSession()，用来获取Session对象。



这个表里罗列了HttpSessionListener接口和HttpSessionActivationListener接口中的方法。这些方法都使用HttpSessionEvent对象作为参数，其中HttpSessionListener接口中有：

1.sessionCreated()方法，作用是通知正在收听的对象，Session已经被加载及初始化；

2.sessionDestroyed()方法，作用是通知正在收听的对象，Session已经被载出。

HttpSessionActivationListener接口中有：

1. sessionDidActivate()方法，作用是通知正在收听的对象，它的Session已经被变为有效状态；

2.sessionWillPassivate()方法的作用是通知正在收听的对象，它的Session已经被变为无效状态。



HttpSessionBindingListener接口用于监听Session范围的属性对象。实现此接口后，只要有对象加入session范围或从session范围中移除，容器就会自动调用监听器的valueBound()方法或valueUnbound()方法。在所有监听接口中，只有HttpSessionBindingListener接口是唯一一个不需要在web.xml中设定的监听接口，它通过被添加到Session中来启用。当实现HttpSessionBindingListener接口的对象被添加为Session的属性时，调用valueBound()方法，被移除Session的属性时，调用valueUnbound()方法。



示例：编写一个HttpSessionBindingListenerDemo类，实现HttpSessionBindingListener接口，监听Session绑定对象的创建和销毁。

1.编写监听器，部分代码如下：

|  |
| --- |
| public class HttpSessionBindingListenerDemo implements HttpSessionBindingListener {  public void valueBound(HttpSessionBindingEvent event) {  System.out.println(event.getName()+"上线");  }  public void valueUnbound(HttpSessionBindingEvent event) {  System.out.println(event.getName()+"下线");  }  } |

2.编写监听器绑定程序，代码如下：

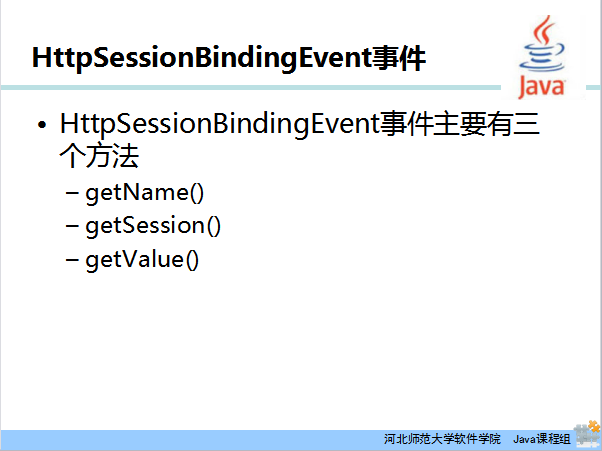
|  |
| --- |
| HttpSession session = request.getSession();  //实例化监听器类的对象  HttpSessionBindingListenerDemo bd = new HttpSessionBindingListenerDemo();  String name = (String)request.getParameter("name");  //将监听器对象添加到Session中  session.setAttribute(name, bd); |

在程序中将监听器对象添加到Session中后，valueBound()方法被调用，测试结果会显示登录人上线。



HttpSessionAttributeListener接口用于监听Session属性对象的变化，功能与ServletContextAttributeListener接口类似，只是范围为Session。接口包含三个方法：attributeAdded()、attributeReplaced()和attributeRemove()，分别监听Session属性值的增加、改变和移除。

HttpSessionAttributeListener接口使用的事件与HttpSessionBindingListener接口监听的都是Session属性对象，所以二者使用的事件相同，都是HttpSessionBindingEvent。不同点在于，前者监听Web站点所有Session范围的变化，后者只监听Session范围内实现了此接口对象的创建与销毁。



HttpSessionAttributeListener接口与HttpSessionBindingListener接口使用的HttpSessionBindingEvent事件，主要包含三个方法，即getSession()、getName()和getValue()，getSession()获取Session对象，getName()获取Session属性的名称，getValue()获取Session属性的值。



这个表列举了HttpSessionBindingListener接口和HttpSessionAttributeListener接口中的主要方法。HttpSessionBindingListener接口中有：

1.valueBound()方法，当实现此监听接口的对象加入Session时，会调用该方法；

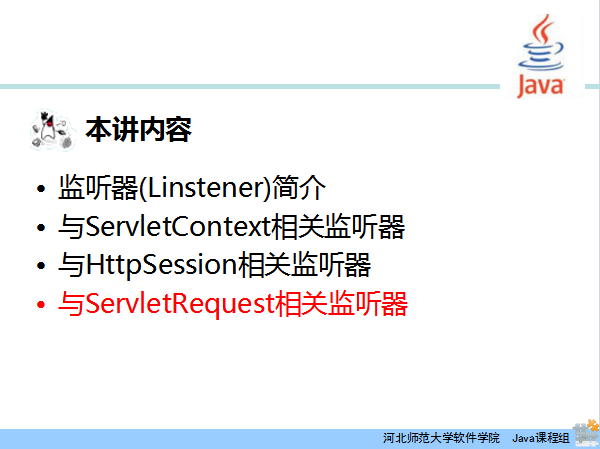
2.valueUnbound()方法，当实现此接口的对象在Session中销毁时，调用该方法。

HttpSessionAttributeListener接口中有：

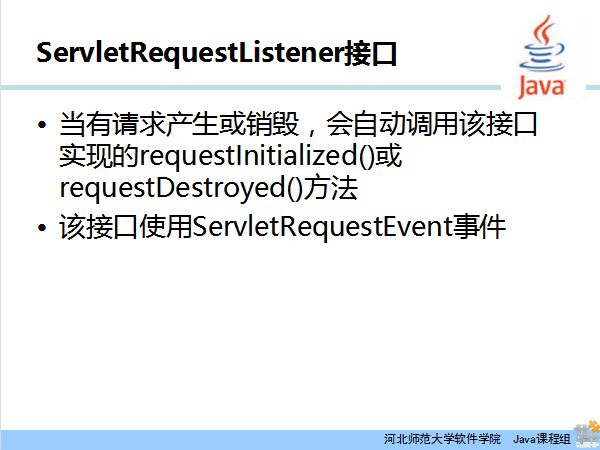
1.attributeAdded()方法，若有对象加入Session范围时，通知正在收听的对象；

2.attributeReplaced()方法，表示在Session的范围，有对象取代另一个对象时，通知正在收听的对象；

3.attributeRemoved()方法，表示若有对象从Session范围移出时，通知正在收听的对象。



第三类监听器，与Request相关的监听器。



ServletRequestListener接口用于监听Request对象的创建和销毁。当Request对象被创建时，调用监听器的requestInitialized()方法，当Request对象被销毁时，调用监听器的requestDestroyed()方法。Request对象创建和销毁的时机是，用户每一次访问都会创建Request对象；当前访问结束，Request对象就会销毁。



当请求产生时，调用的requestInitialized()方法，表示通知正在收听的对象，Request已经被加载及初始化；当请求销毁时，调用requestDestroyed()方法，表示通知正在收听的对象，Request已经被载出。这两个方法接收的参数ServletRequestEvent，主要包含两个方法，getServletContext()方法和getServletRequest()方法。前者表示获取ServletContext对象，后者表示获取Request请求对象。

示例：编写一个ServletRequestListenerDemo类，实现ServletRequestListener接口，监听Request对象的创建和销毁。

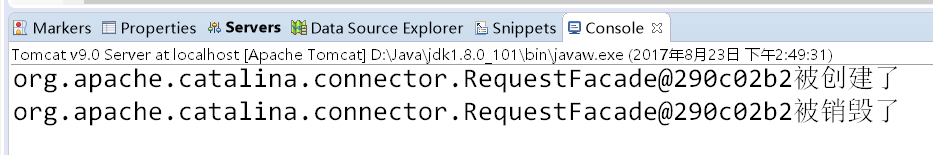
1.编写监听器，部分代码如下：

|  |
| --- |
| public class ServletRequestListenerDemo implements ServletRequestListener {  public void requestDestroyed(ServletRequestEvent sre) {  System.out.println(sre.getServletRequest()+"被销毁了");  }  public void requestInitialized(ServletRequestEvent sre) {  System.out.println(sre.getServletRequest()+"被创建了");  }  } |

2.在web.xml文件中注册监听器，代码如下：

|  |
| --- |
| <listener>  <listener-class>onest.listener.ServletRequestListenerDemo</listener-class>  </listener> |

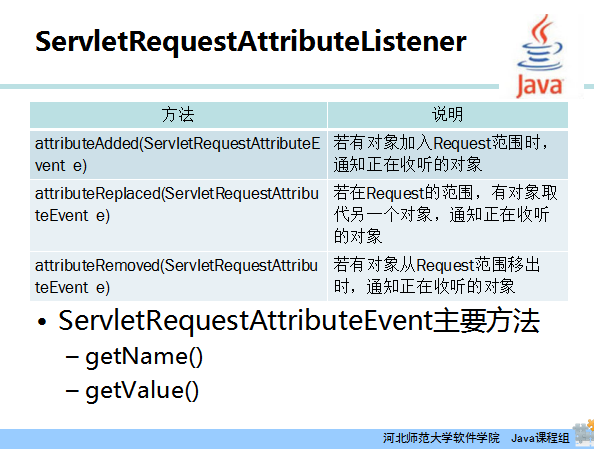
运行结果，如下图所示。



从运行结果中可以看到，用户每一次访问都会创建Request对象，当访问结束后，Request对象就会销毁。



ServletRequestAttributeListener接口用于监听Reqeust范围属性对象的变化，主要有三个方法，attributeAdded()方法用于监听Request属性的增加，attributeReplaced()方法用于监听Request属性的替换，attributeRemoved()用于监听Request属性的移除。这个接口依赖ServletRequestAttributeEvent类。



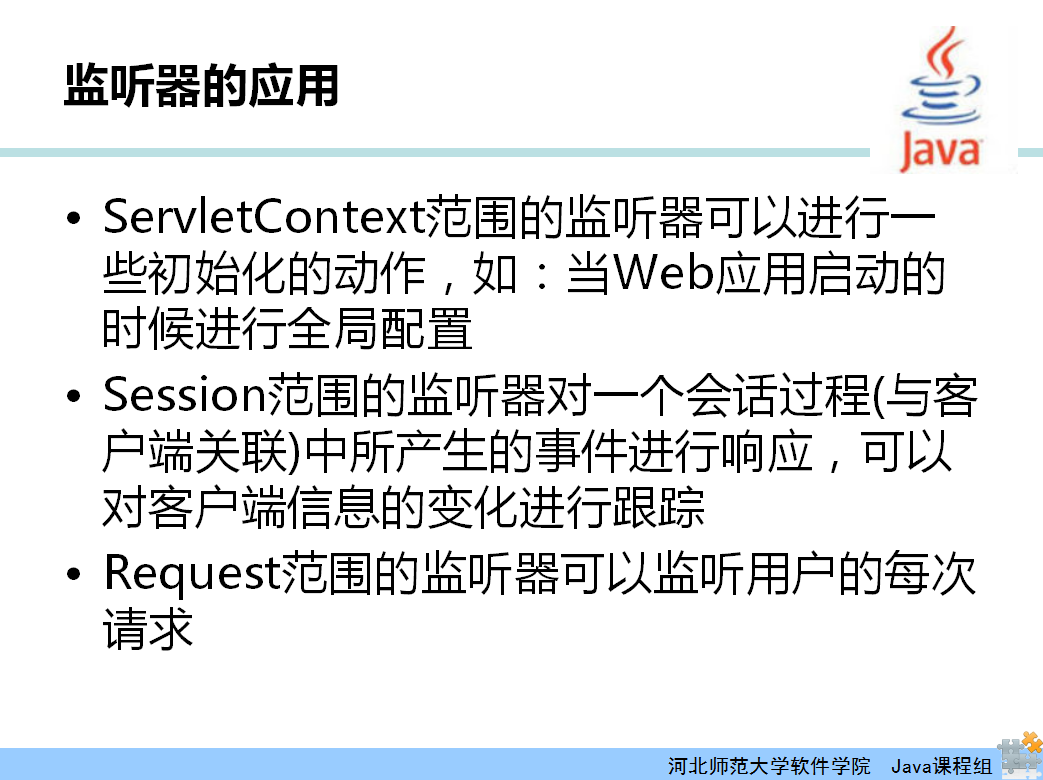
ServletRequestAttributeListener接口中有：

1.attributeAdded()方法，表示若有对象加入Request范围时，通知正在收听的对象；

2.attributeReplaced()表示若在Request的范围，有对象取代另一个对象，通知正在收听的对象；

3.attributeRemoved()方法表示若有对象从Request范围移出时，通知正在收听的对象。

这三个方法都接收ServletRequestAttributeEvent对象作为参数，ServletRequestAttributeEvent类中主要方法有getName()方法和getValue()方法，前者用来获取Request的属性名称，后者用来获取Request的属性值。



介绍了以上三类共八个Listener接口：

1.两个ServletContext范围的监听接口，ServletContext监听接口和ServletContext属性监听接口；

2.四个Session范围的监听接口，Session监听接口、Session状态监听接口、Session属性监听接口和Session绑定监听接口；

3.两个Request范围的监听接口，Request监听接口和Request属性接口。

在编写监听器时，除了要实现某个监听接口外，还要在web.xml中进行配置，或者不配置web.xml而是使用注解，即在监听器上标注@WebListener，则容器在部署应用程序时，会实例化类并注册给应用程序。

这些接口监听的对象不同，应用的场景也有所不同。ServletContext范围的监听器，可以进行一些初始化的动作，当Web应用启动的时候进行全局配置，如设置缓存功能的定时器；Session范围的监听器，可以对一个会话过程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，如统计当前在线人数、用户访问时长或网站的访问量等；Request范围的监听器,可以监听用户的每一次请求，如统计某个新闻的点击量，用户的每次访问都要计数。

## 课时小结

（10分钟）

1. 与ServletContext相关监听器的编写和使用。
2. 与HttpSession相关监听器的编写和使用。
3. 与ServletRequest相关监听器的编写和使用。
4. 不同监听器的作用和应用场景。

# 布置作业

（5分钟）

1. 完成监听器的实验手册。
2. 使用监听器实现在线人数的统计。