**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。

## 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基本原理和技术实现，重点介绍了Session技术。Session技术的基本原理是什么呢？当用户访问服务器的时候，服务器会为每一个用户创建一个Session对象来存储用户信息，它有唯一的ID号，并且会发送给客户端浏览器。客户端浏览器每次访问时，或说请求时，都会带着这个ID,来让服务器找到对应Session对象。所以说，Session是在一个用户的多次请求间共享的，不同的用户有不同Session，有多少用户正在访问，那服务器就至少有多少Session对象存在。为什么说至少呢，这就涉及Session的销毁，这里就不再多说了。

## 作业点评

（0分钟）

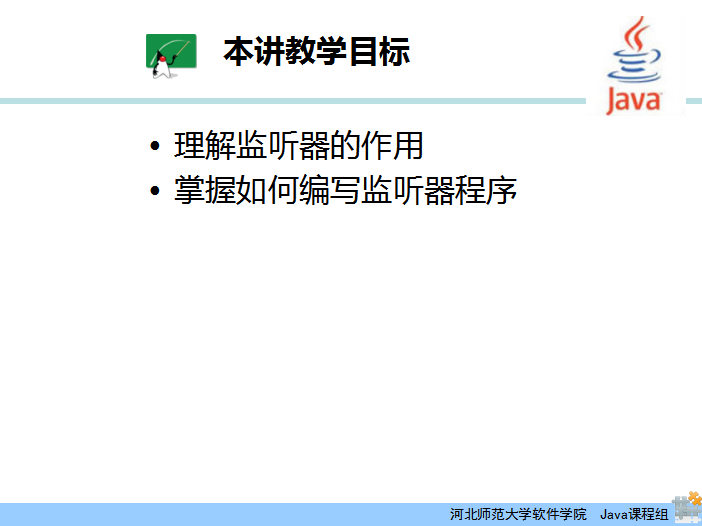
## 导入新课

（1分钟）

经常有应用使用当前在线用户数来展示它的受欢迎程度，比如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，这些功能是怎么实现的呢？从上一章我们了解到，每当一个用户访问服务器时，服务器就会为之创建一个HttpSession，那如果要统计一个服务器的访问量，是不是可以这样做：设计一个工具，这个工具专门去监视Session创建这件事，只要服务器创建一个Session，那这个工具就让代表访问量的变量加一。这样可以做到统计网站的访问量吗？当然可以，而且真实项目中就是这么做的。这个工具，就是这章要讲的监听器。

## 讲授新课

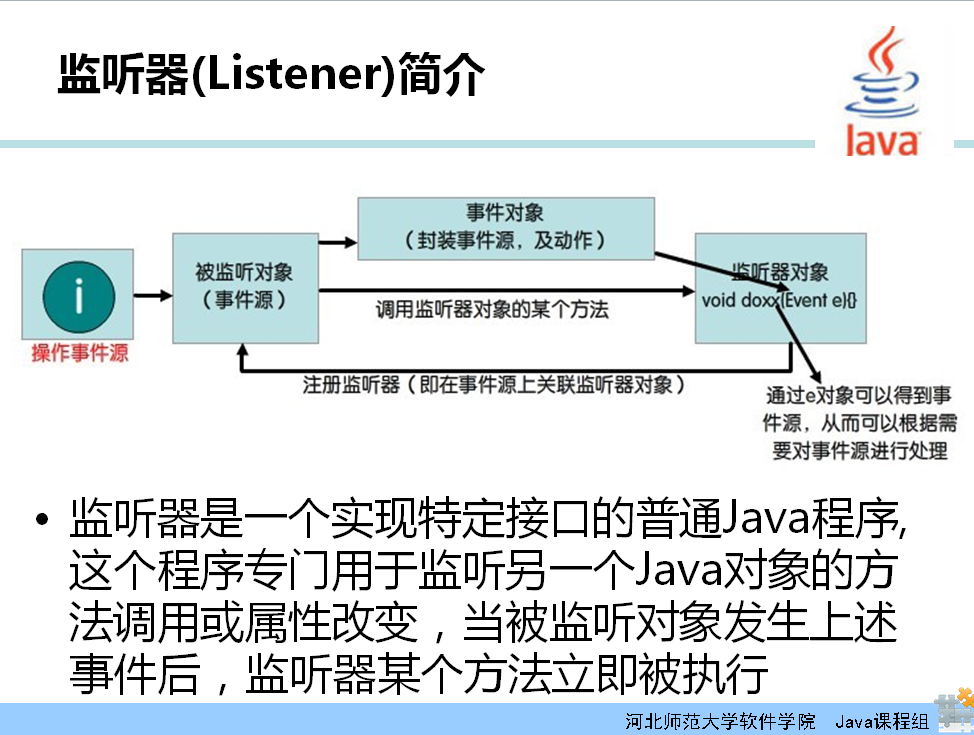
（120分钟）



本章的教学目标是理解监听器的作用，掌握监听接口中方法的含义及使用方法，重点掌握监听器的编程实现。



首先对监听器进行简单介绍。



什么是监听器？举个例子，进停车场时要扫描车牌，记录车辆进入停车场的时间，这个扫描器就是一个监听器。它专门监听车进库这个事件，如果车在这熄火了它不记录的，人进车库也不记录，也就是说只有车进库这个事件发生时，它才会处理。

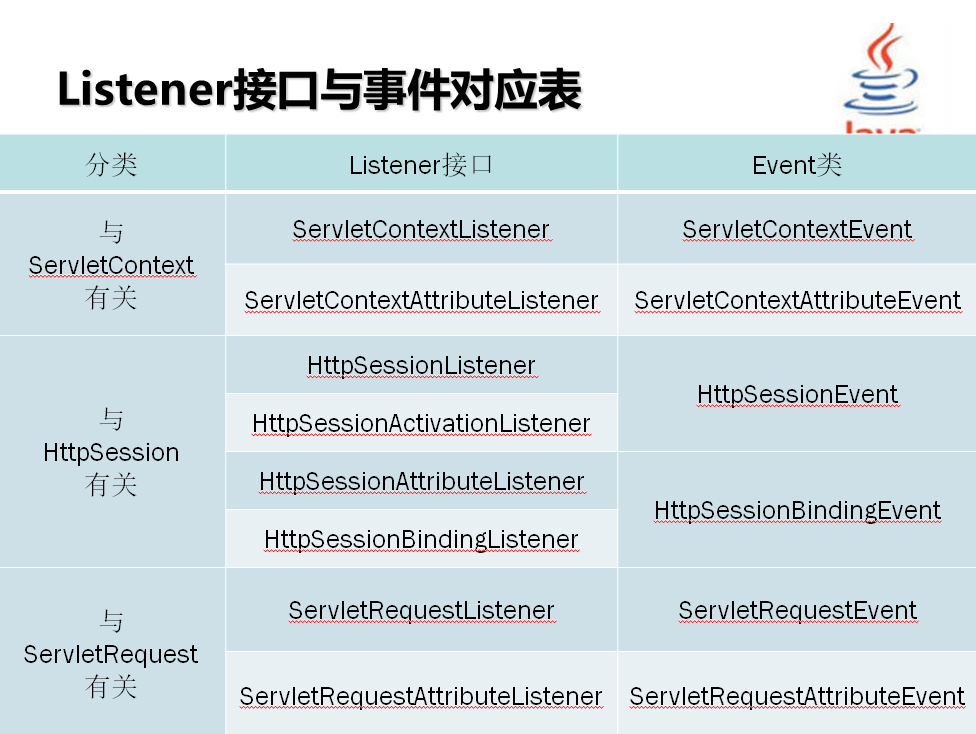
在JavaWeb中，监听器，又称Listener，它的作用是当服务器端发生了某个事件时，调用事件处理程序。为了描述监听器的运行原理，先了解事件监听涉及到的三个组件：

1.事件源，即被监听对象；

2.事件，一个对象发生了一件事就是一个事件，车进库是个事件，车出库是个事件，Session创建也是一个事件，Session销毁也是一个事件；

3.事件监听器,即监听器对象。

当事件源上发生某一个动作时，它会调用事件监听器的一个方法，并在调用该方法时把事件对象传递进去，开发人员在监听器中通过事件对象，就可以拿到事件源，从而对事件源进行操作。也就是说，监听器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中可以被监听的对象，也就是事件源，有三种：

1.ServletContext，即整个Web应用范围的对象，在Web应用中共享一个ServletContext对象，在这个对象里设置的属性，也为Web应用所共享。比如网站访问量，属于整个网站的数据，就可以设置为ServletContext对象的属性，从而让全站共享；

2.HttpSession，用户范围的对象，一个用户的多次请求共享一个Session；

3.ServletRequest，客户端浏览器与服务器一次交互范围的对象。

监听器其实就是实现特定的监听接口的类，专门三个对象及其属性的添加、移除、修改事件。

这八个监听接口和六个事件类也是围绕这三个对象及其属性对象来编程的。从接口和类名上也可以找到规律:

1.ServletContextListener监听接口对应ServletContextEvent事件类，监听ServletContext的创建和销毁事件；

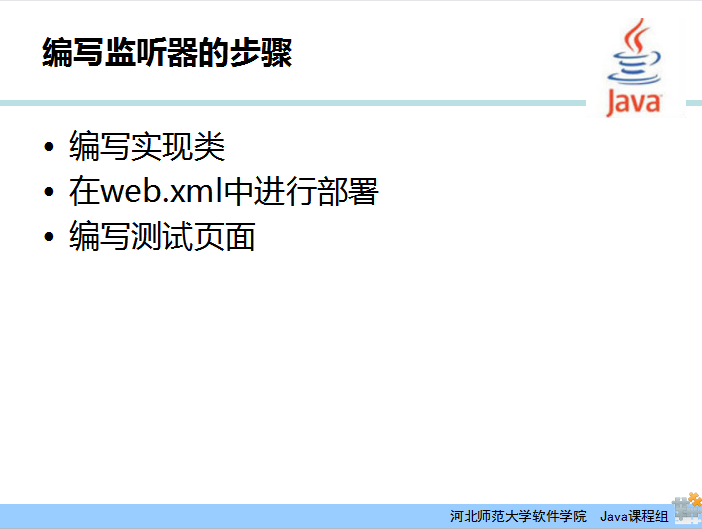
2.ServletContextAttributeListener监听接口依赖ServletContextAttributeEvent事件类，监听ServletContext中属性对象的增加和删除事件；

3.HttpSessionListener监听接口和HttpSessionActivationListener监听接口依赖HttpSessionEvent事件类，分别监听HttpSession对象的创建、销毁事件和失效、有效状态；

4.HttpSessionAttributeListener监听接口和HttpSessionBindingListener监听接口依赖HttpSessionBindingEvent事件类，分别监听HttpSession中属性对象的增加、删除事件和绑定、解绑事件；

5.ServletRequestListener监听接口依赖ServletRequestEvent事件类，监听ServletRequest对象的创建和销毁事件；

6.ServletRequestAttributeListener监听接口依赖ServletRequestAttributeEvent类，监听ServletRequest中属性对象的增加和删除事件。



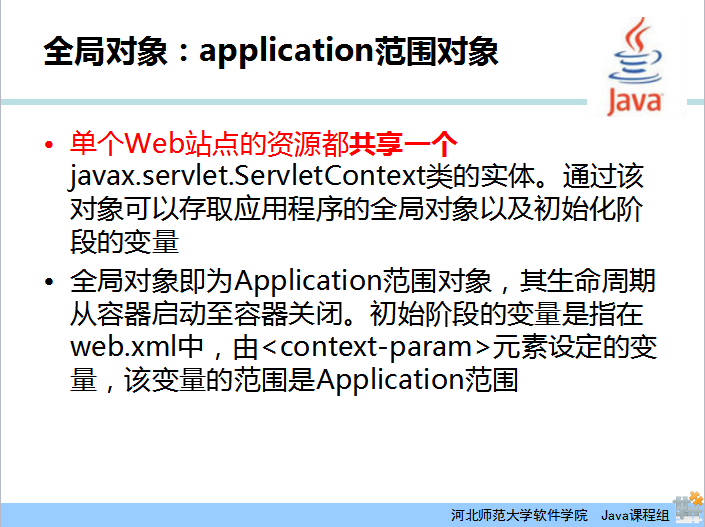
编写一个监听器，首先要编写它的实现类，这个类必须实现某个监听接口。

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。



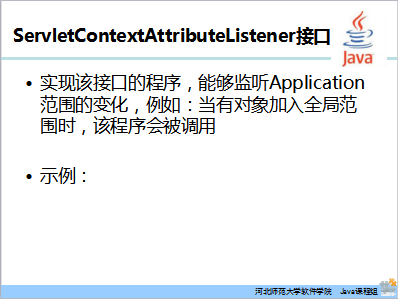
介绍与ServletContext相关监听器之前，先回顾一下ServletContext对象。ServletContext是一个Application范围的全局对象，每个Web应用启动后，Servlet容器都会创建唯一的一个ServletContext类的对象，直到Servlet容器关闭时消失。所以单个Web站点的资源都共享一个ServletContext对象，该对象包含整个Web程序的信息，通过该对象可以存取应用程序的全局对象以及初始化阶段的变量。初始阶段的变量是指在web.xml中，由<context-param>元素设定的变量，该变量的范围是Application范围。



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。



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。

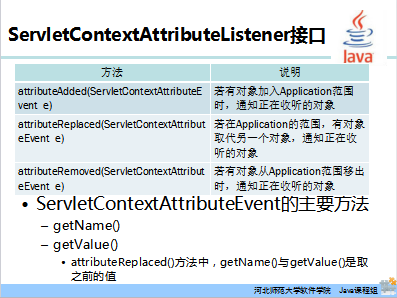
1.编写监听器，部分代码如下：

|  |
| --- |
| 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：

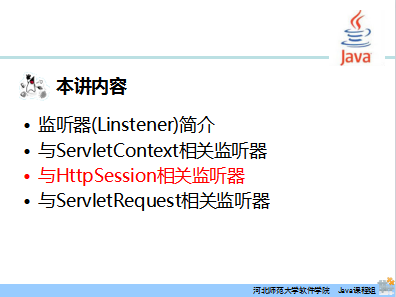
|  |
| --- |
| <listener> <listener-class>onest.listener.ServletContextAttributeLinstenerDemo</listener-class>  </listener> |

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。

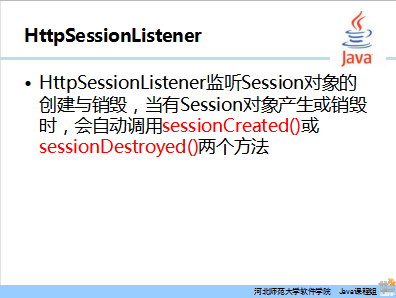


ServletContextAttributeListener接口中attributeAdded()方法的作用是，当有对象加入Application范围时，通知正在收听的对象，也就是ServletContext的setAttribute()方法被调用时，会触发attributeAdded()方法执行；attributeReplaced()方法的作用是，若在Application的范围，有对象取代另一个对象，通知正在收听的对象，也就是ServletContext的setAttribute()方法设置的属性名存在时，这个属性值被替换，那么attributeReplaced()方法中的代码被执行;attributeRemoved()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



HttpSessionListener接口用于监听HttpSession对象的创建和销毁。创建一个HttpSession时，激发sessionCreated()方法，销毁一个HttpSession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HttpSession对象。

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

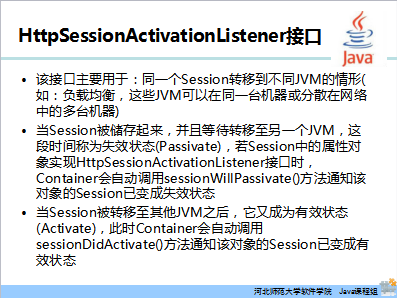
1.编写监听器，部分代码如下：

|  |
| --- |
| 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：

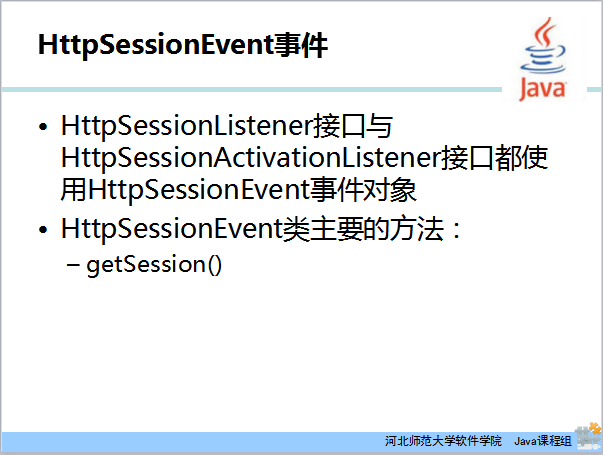
|  |
| --- |
| <listener>  <listener-class>onest.listener.HttpSessionLinstenerDemo</listener-class>  </listener>  <!-- 配置HttpSession对象的销毁时机 -->  <session-config>  <!--配置HttpSession对象的1分钟之后销毁 -->  <session-timeout>1</session-timeout>  </session-config> |

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

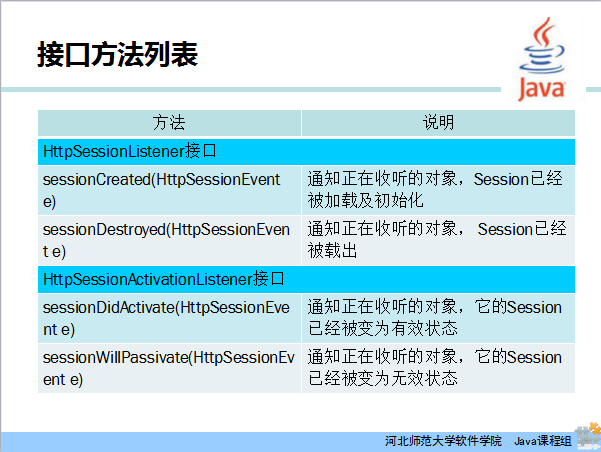


与HttpSessionListener接口依赖同一个Event类的HttpSessionActivationListener接口主要用于在分布式环境中同一个HttpSession转移到不同JVM的情形。比如在同一台计算机中有多个JVM，或这些JVM分散在网络中的多台计算机，当HttpSession在这些JVM中转移时，会用到HttpSessionActivationListener接口。

这个接口主要有两个方法，sessionWillPassivate()方法和sessionDidActivate()方法。将HttpSession转移至另一个JVM时，必须将HttpSession对象在原本的JVM中的所有属性对象序列化，这时HttpSession就变为失效状态(Passivate)，若在这之前HttpSession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HttpSession已变成失效状态。当HttpSession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HttpSession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HttpSession对象的创建和销毁，而后者监听HttpSession对象的有效或无效状态，这两个接口监听的是HttpSession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HttpSession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：

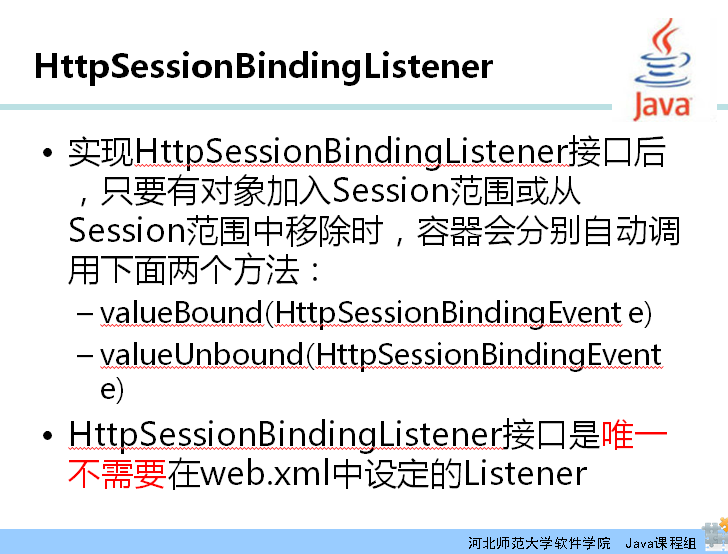
1.sessionCreated()方法，作用是通知正在收听的对象，HttpSession已经被加载及初始化；

2.sessionDestroyed()方法，作用是通知正在收听的对象，HttpSession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HttpSession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HttpSession已经被变为无效状态。



HttpSessionBindingListener接口用于监听HttpSession范围的属性对象。实现此接口的类对象加入HttpSession范围或从HttpSession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HttpSession中来启用。当实现HttpSessionBindingListener接口的对象被添加为HttpSession的属性时，调用valueBound()方法，被移除HttpSession时，调用valueUnbound()方法。



示例：编写一个HttpSessionBindingListenerDemo类，实现HttpSessionBindingListener接口，监听HttpSession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：

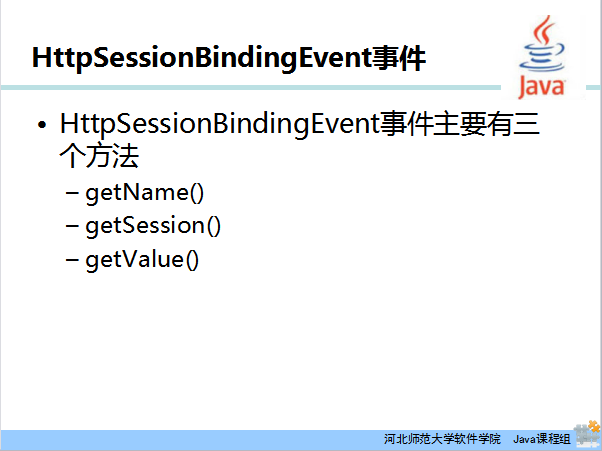
|  |
| --- |
| HttpSession session = request.getSession();  //实例化监听器类的对象  HttpSessionBindingListenerDemo bd = new HttpSessionBindingListenerDemo();  String name = (String)request.getParameter("name");  //将监听器对象添加到Session中  session.setAttribute(name, bd); |

在程序中将监听器对象添加到HttpSession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HttpSession属性对象的变化，功能与ServletContextAttributeListener接口类似，只是范围为HttpSession。接口包含三个方法：attributeAdded()、attributeReplaced()和attributeRemove()，分别监听HttpSession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HttpSession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HttpSession范围的变化，后者只监听HttpSession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HttpSession对象，getName()获取HttpSession属性的名称，getValue()获取HttpSession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HttpSession时，会调用该方法；

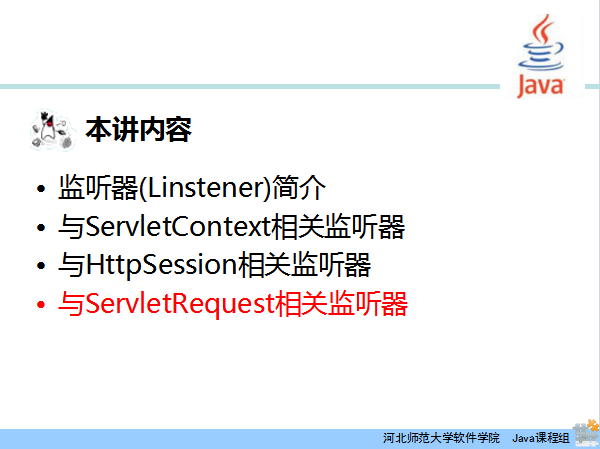
2.valueUnbound()方法，当实现此接口的对象在HttpSession中销毁时，调用该方法。

HttpSessionAttributeListener接口中有：

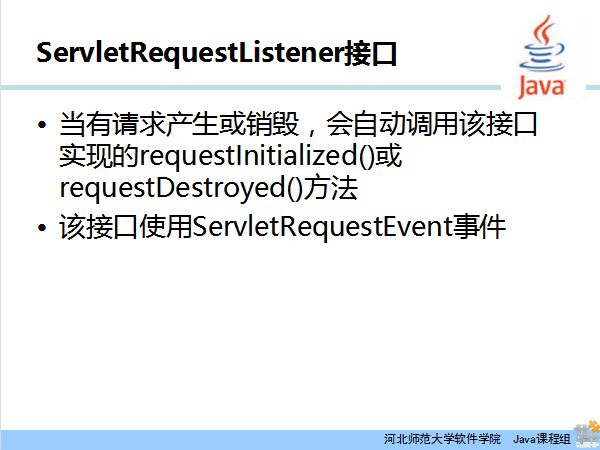
1.attributeAdded()方法，若有对象加入HttpSession范围时，通知正在收听的对象；

2.attributeReplaced()方法，表示在HttpSession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HttpSession范围移出时，通知正在收听的对象。



第三类监听器，与ServletRequest相关的监听器。



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

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

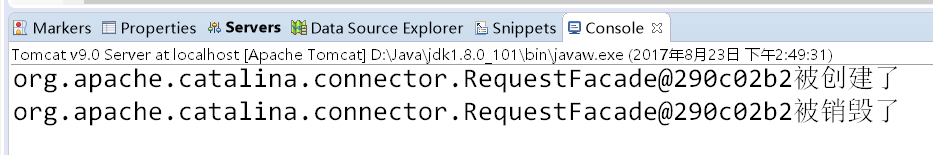
1.编写监听器，部分代码如下：

|  |
| --- |
| 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> |

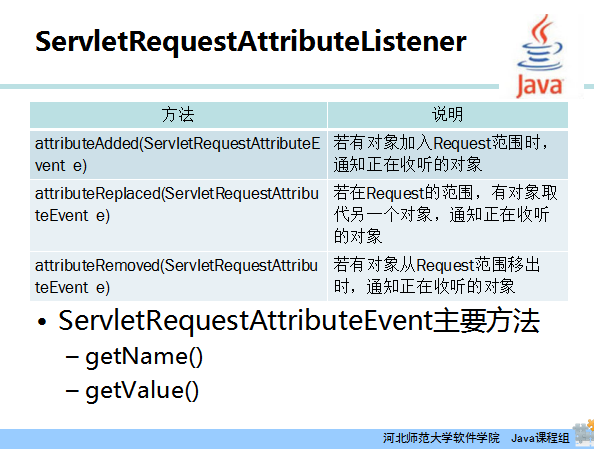
运行结果，如下图所示。



从运行结果中可以看到，用户每一次访问都会创建ServletRequest对象，当访问结束后，ServletRequest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ServletRequest属性的增加，attributeReplaced()方法用于监听ServletRequest属性的替换，attributeRemoved()用于监听ServletRequest属性的移除。这个接口依赖ServletRequestAttributeEvent类。



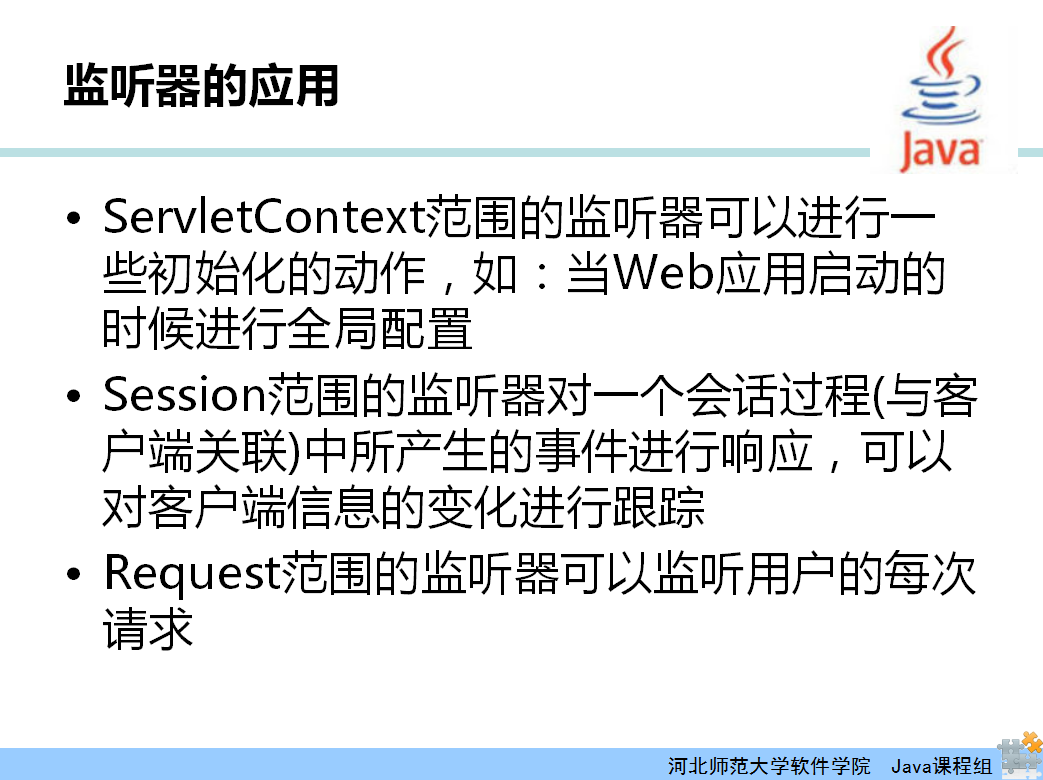
ServletRequestAttributeListener接口中有：

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

2.attributeReplaced()表示若在ServletRequest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ServletRequest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ServletRequest的属性名称，后者用来获取ServletRequest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HttpSession范围的监听接口，HttpSession监听接口、HttpSession状态监听接口、HttpSession属性监听接口和HttpSession绑定监听接口；

3.两个ServletRequest范围的监听接口，ServletRequest监听接口和ServletRequest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；HttpSession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（延伸，保存数据）、用户访问时长或网站的访问量等；ServletRequest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。