一.监听器

1.监听某一个事件的发生, 状态的改变。

2.监听器的内部机制: 其实就是接口回调。

3.Servlet一共有8个监听器。按作用分为三大类:监听三个作用域的创建和销毁、监听三个作用域属性状态的变更和监听httpSession里面存值状态的变更。

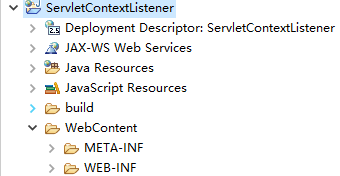
4.被监听的三个作用域分别是: request(HttpServletRequest)、session(HttpSession)和application(ServletContext)。

二.监听三个作用域的创建和销毁

1.ServletContextListener监控ServletContext的创建和销毁。

(1).项目被服务器(Tomcat服务器)加载的时候创建ServletContext; 项目被服务器(Tomcat服务器)移除或关闭服务器的时候销毁ServletContext。

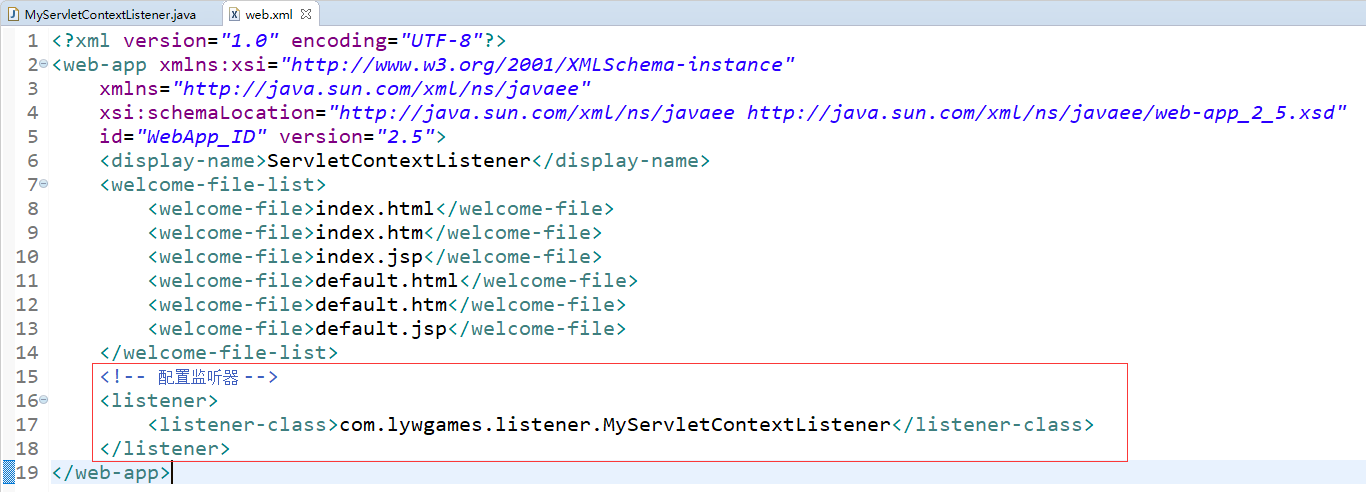
(2).创建名为ServletContextListener的Web工程



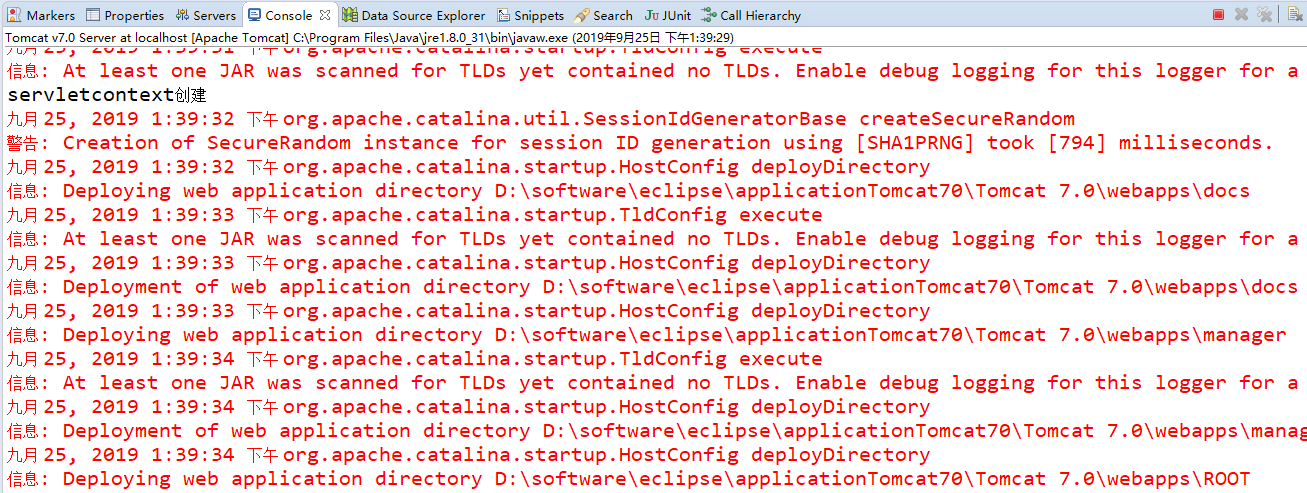
(3).编写MyServletContextListener.java实现ServletContextListener接口

|  |
| --- |
| package com.lywgames.listener;  import javax.servlet.ServletContextEvent;  import javax.servlet.ServletContextListener;  public class MyServletContextListener implements ServletContextListener {  /\*\*  \* 项目被服务器(Tomcat服务器)加载的时候调用  \*/  @Override  public void contextInitialized(ServletContextEvent sce) {  System.out.println("servletcontext创建");  }  /\*\*  \* 项目被服务器(Tomcat服务器)移除的时候调用  \*/  @Override  public void contextDestroyed(ServletContextEvent sce) {  System.out.println("servletcontext销毁");  }  } |

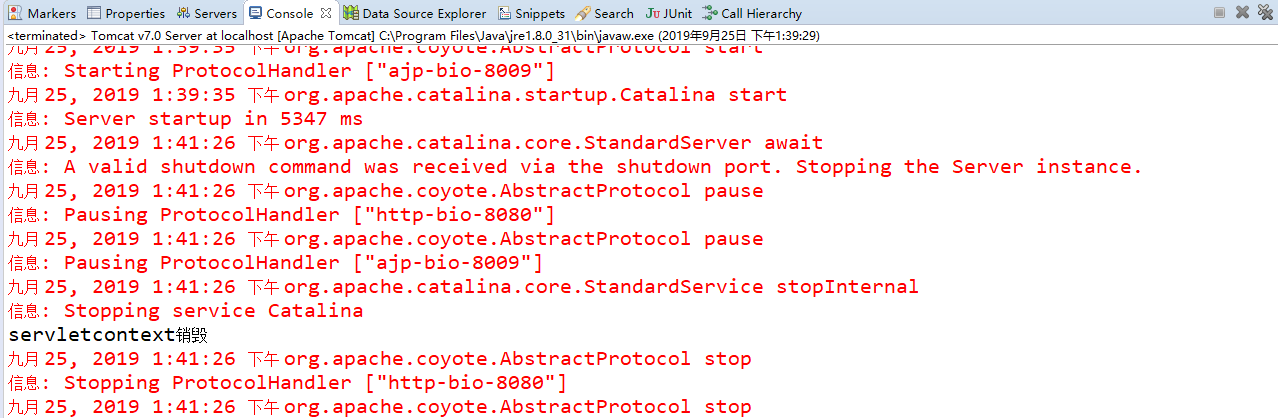
(4).在web.xml里配置监听器



(5).运行该项目, 监控到servletcontext被创建了



(6).关闭服务器, 监控到servletcontext被销毁了

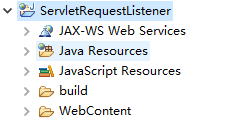


3.ServletRequestListener监控request的创建和销毁。

(1).客户端(浏览器)访问服务器上的资源的时候就会有request的创建; 服务器对客户端做出响应的时候就会有request的销毁。

(2).客户端(浏览器)访问服务器上的任何资源都会有request的创建, 包括html、jsp和servlet。

(3).创建名为ServletRequestListener的Web工程



(4).编写MyServletContextListener.java实现ServletRequestListener接口

|  |
| --- |
| package com.lywgames.listener;  import javax.servlet.ServletRequestEvent;  import javax.servlet.ServletRequestListener;  public class MyServletRequestListener implements ServletRequestListener {  /\*\*  \* 有客户端访问服务器资源的时候调用  \*/  @Override  public void requestInitialized(ServletRequestEvent sre) {  System.out.println("有request请求");  }  /\*\*  \* 服务器对客户端做出响应的时候调用  \*/  @Override  public void requestDestroyed(ServletRequestEvent sre) {  System.out.println("有request销毁了");  }  } |

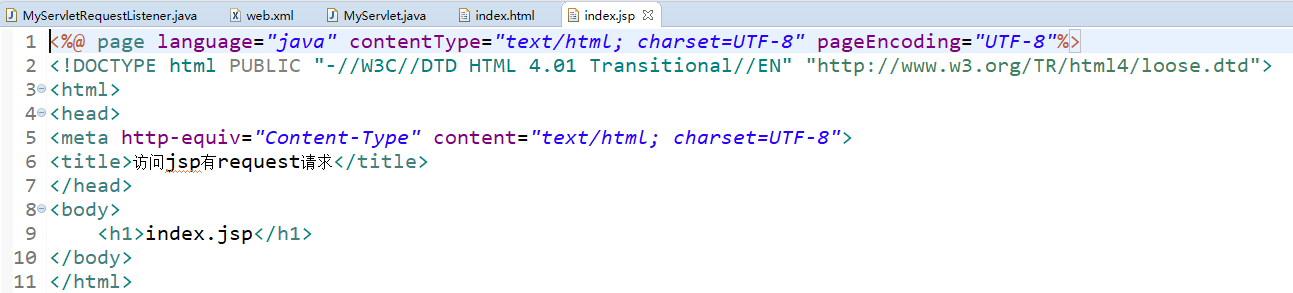
(6).创建一共MyServlet.java类



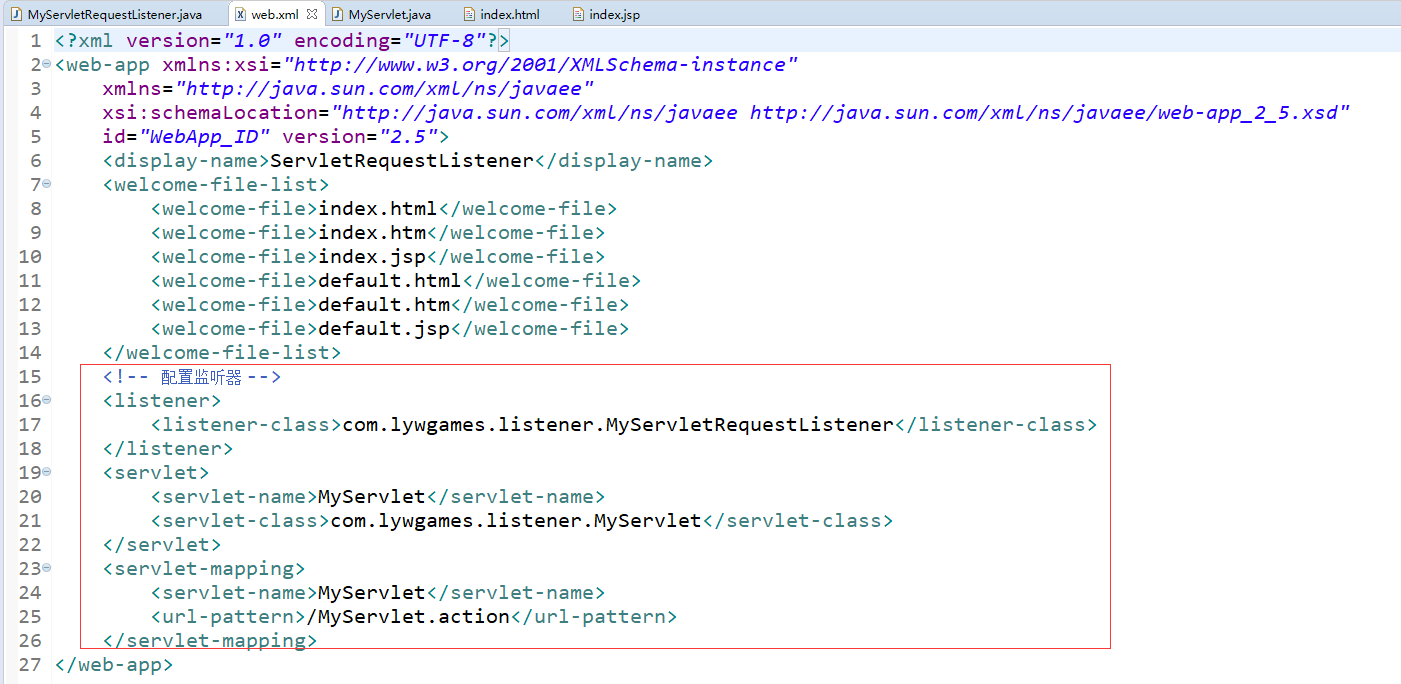
(7).创建index.html



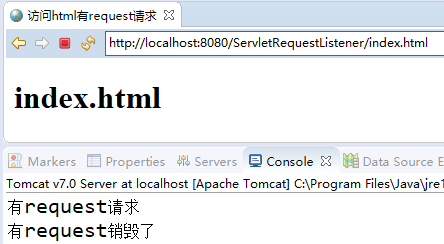
(8).创建index.jsp



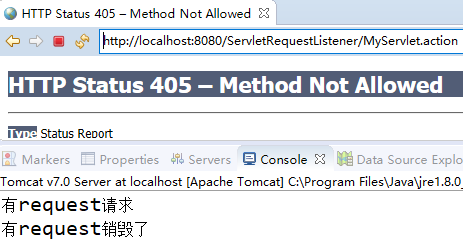
(9).在web.xml里配置监听器和Servlet



(10).部署项目, 访问index.html、index.jsp和MyServlet都有request的创建和销毁



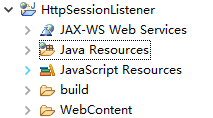




4.HttpSessionListener监控session的创建和销毁。

(1).只要有getSession()方法的调用, 就会有session的创建。访问jsp的时候, 因为jsp页面有session=true(默认也是true)的设置, 所以访问jsp就会有session的创建; 如果设置session=false, 就不会有session的创建。访问Servlet的时候, 如果Servlet里有getSession()的调用, 则会创建session; 如果没有getSession()的调用, 则不会创建session。访问html没有session的创建。超过30分钟或者非正常关闭服务器, 则会销毁session; 正常关闭服务器会序列化session。

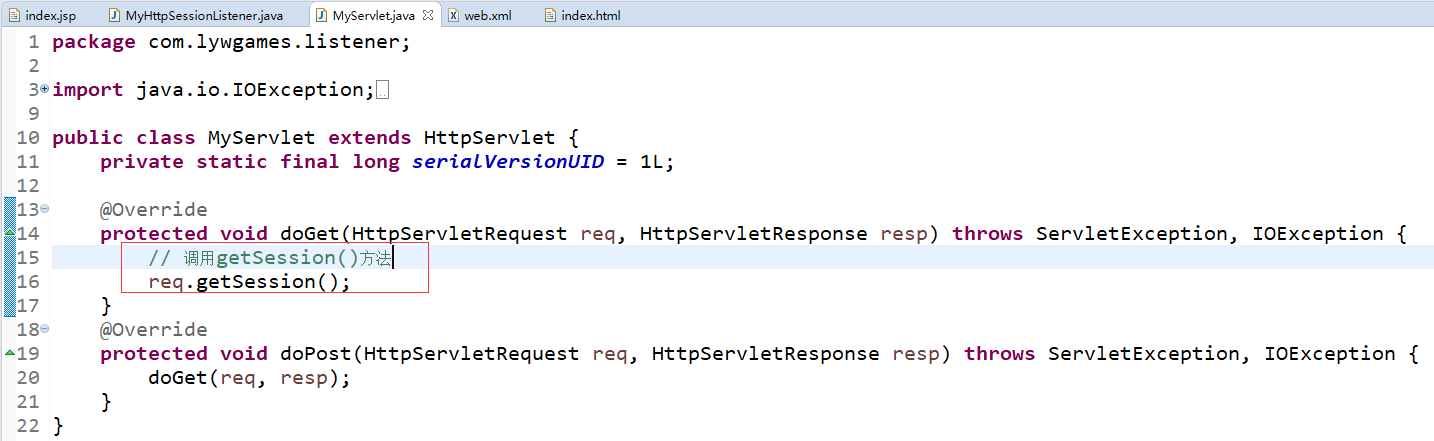
(2).创建名为HttpSessionListener的Web工程



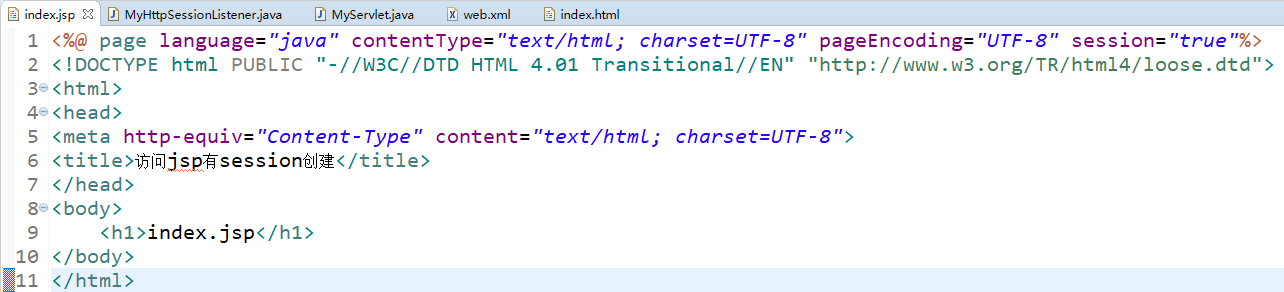
(3).编写MyHttpSessionListener.java实现HttpSessionListener接口

|  |
| --- |
| package com.lywgames.listener;  import javax.servlet.http.HttpSessionEvent;  import javax.servlet.http.HttpSessionListener;  public class MyHttpSessionListener implements HttpSessionListener {    @Override  public void sessionCreated(HttpSessionEvent hse) {  System.out.println("创建了session");  }  @Override  public void sessionDestroyed(HttpSessionEvent hse) {  System.out.println("销毁了session");  }  } |

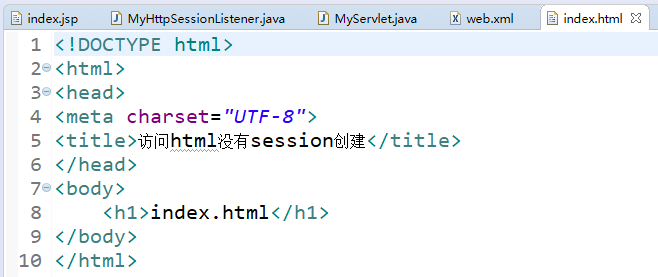
(4).创建MyServlet并调用getSession()方法



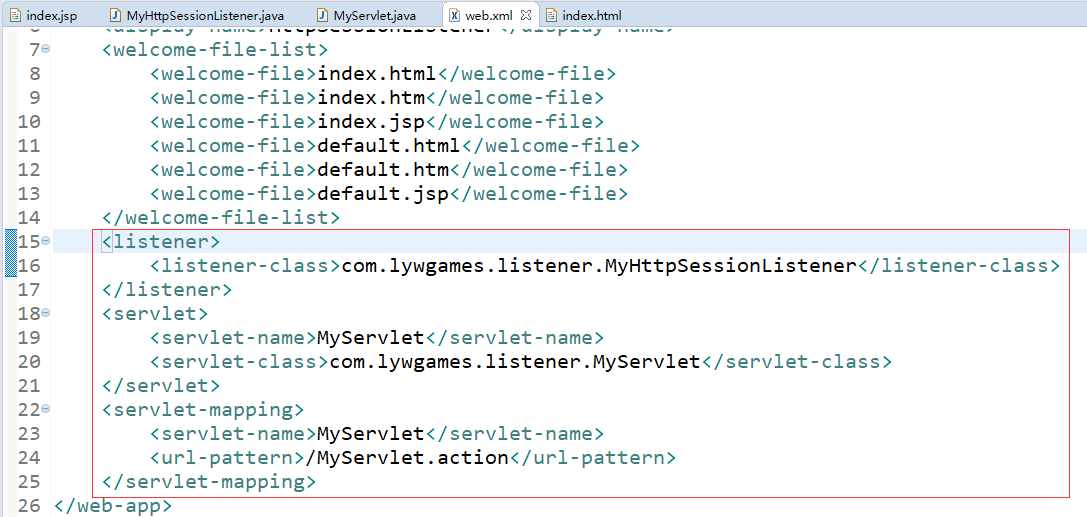
(5).创建index.jsp



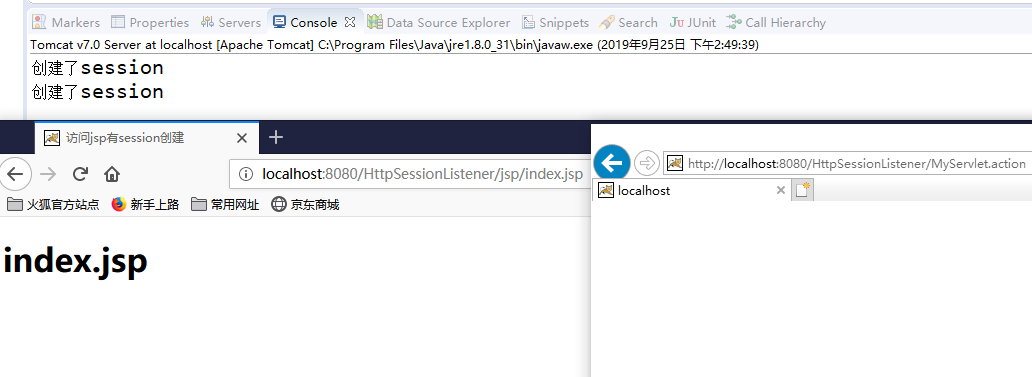
(6).创建index.html



(7).在web.xml里配置监听器和Servlet



(8).部署项目, 访问index.jsp和MyServlet都有session的创建



5.ServletContextListener的作用

(1).在servletcontext创建的时候, 完成自己想要的初始化工作。

6..HttpSessionListener可以用来统计在线人数。

三.监听三个作用域中值的添加、替换和移除

1.ServletContextAttributeListener监听servletContext作用域中值的添加、替换和移除。

2.ServletRequestAttributeListener监听request作用域中值的添加、替换和移除。

3.HttpSessionAttributeListener监听session作用域中值的添加、替换和移除。

四.监听httpSession里面存值状态的变更(这种监听器不用我们注册)

1.HttpSessionBindingListener监听对象与session绑定和解除绑定的动作。

2.HttpSessionActivationListener用于监听现在session的值是钝化(序列化)还是活化(反序列化)的动作。

(1).钝化(序列化): 把内存中的数据, 存储到硬盘上。

(2).活化(反序列化): 把硬盘中的数据读取到内存中。

(3).session中的值可能会很多, 并且我们有很长一段时间不使用这个内存中的值, 那么可以考虑把session的值可以存储到硬盘上[钝化], 等下一次在使用的时候, 在从硬盘上提取出来[活化]。

(4).配置session的钝化时间

--在tomcat里面 conf/context.xml 里面配置: 对所有的运行在这个服务器的项目生效。

--在conf/Catalina/localhost/context.xml 配置: 对localhost生效。

--在自己的web工程项目中的 META-INF/context.xml: 只对当前的工程生效。

(5).maxIdleSwap: 钝化时间, directory: 钝化路径。

