Listener监听者之ServletContextListener

1. **package** com.example.listener;

**import** javax.servlet.\*;

/\*\*

\* Web应用上下文监听者

\* **@author** ixenos

\*

\*/

**public** **class** MyServletContextListener **implements** ServletContextListener {

/\*\*

\* 监听上下文初始化时，进行的附加操作

\* 比如：初始化数据库连接的代码

\* （从ServletContext得到上下文初始化参数，

\* 通过初始化参数建立一个数据库连接，把数据库连接**存储为一个属性**，使得Web应用各部分都能访问）

\*/

@Override

**public** **void** contextInitialized(ServletContextEvent sce) {

// **TODO** 初始化数据库连接的代码，然后存储为一个上下文属性

System.***out***.println("HelloWorld");

}

/\*\*

\* 监听上下文撤销时，进行的附加操作

\* 比如：关闭数据库连接

\*/

@Override

**public** **void** contextDestroyed(ServletContextEvent sce) {

// **TODO** 关闭数据库连接的代码

System.***out***.println("GoodbyeWorld");

}

}

2.在DD中这样配置Listener

<!-- Listener -->

<listener>

<listener-class>com.example.listener.MyServletContextListener</listener-class>

</listener>

配合servlet:

<servlet>

<servlet-name>testListener</servlet-name>

<servlet-class>com.example.controller.TestListenerServlet</servlet-class>

</servlet>

<context-param>

<param-name>breed</param-name>

<param-value>popi</param-value>

</context-param>

<servlet-mapping>

<servlet-name>testListener</servlet-name>

<url-pattern>/testLis.do</url-pattern>

</servlet-mapping>

<listener>

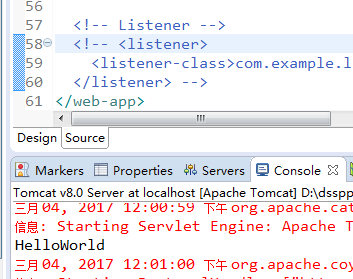
<listener-class>com.example.listener.MyServletContextListener</listener-class>

</listener>

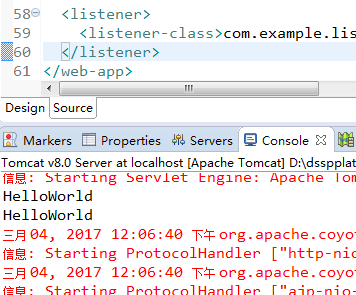
3.要记住，ServletContextListener是ServletContext的Listener！服务整个Web应用。

Tomcat+eclipse在加载一个项目的web.xml的时候，还会加载另外的项目的web.xml,寻找Listener标签的ServletContextListener，即使不是部署运行它！，同样会执行，我通过一句打印测试出了两次调用

屏蔽另一个项目的listener的时候，就只有一个listener



不屏蔽的时候，有两个listener在运行



4.步骤：

1) 容器读取这个应用的DD，获得<listener>和<context-param>

2）**容器为这个应用创建一个新的ServletContext对象**，应用的所有部分都会共享这个上下文（ServletContext的创建早于ServletConfig）

3）容器为每个上下文参数创建一个String名/值对

4）容器将名/值参数的引用交给ServletContext

5）容器创建MyServletContextListener类的一个新的实例

6）容器调用MyServletContextListener的contextInitalized()方法，传入新的ServletContextEvent，这个事件对象含有一个ServletContext引用，所以能通过它调用ServletContext，然后从中得到上下文初始化参数（我们在DD中的<context-name>）

7）监听者向ServletContextEvent要到了一个ServletContext的引用

8）监听者再向这个ServletContext要一个上下文初始化参数“breed”

9）监听者使用这个初始化参数来构造一个新的Dog对象

10）监听者把Dog设置为ServletContext中的一个属性（context.setAttribute(。。。)）

11）容器创建一个新的servlet：利用初始化参数建立一个新的servletConfig，然后为这个ServletConfig提供我们ServletContext的一个引用，然后调用servlet的init()方法

12）Servlet得到一个请求，然后业务代码中向ServletContext请求属性“Dog”ctx.getAttribute(“Dog”) Dog是上下文属性名

13）Servlet在Dog上调用getBreed()，并将结果打印到HttpResponse

5.不止有ServletContextListener这一个监听者（Listener）

只要是servlet生命周期中的重要时刻，总会有一个监听者监听。除了上下文事件外（监听者接收事件），还有监听**上下文属性**、**Servlet请求**和**属性**，以及**HTTP会话属性**和会话属性相关的事件。