# Context-param

Context-param元素含有一对参数名和参数值，用作应用的ServletContext上下文初始化参数。参数名在整个web应用中必须是唯一的。

<context-param>的作用:

web.xml的配置中<context-param>配置作用

1. 启动一个WEB项目的时候,容器(如:Tomcat)会去读它的配置文件web.xml.读两个节点: <listener></listener> 和 <context-param></context-param>

2.紧接着,容器创建一个ServletContext(上下文),这个WEB项目所有部分都将共享这个上下文.

3.容器将<context-param></context-param>转化为键值对,并交给ServletContext.

4.容器创建<listener></listener>中的类实例,即创建监听.

5.在监听中会有contextInitialized(ServletContextEvent args)初始化方法,在这个方法中获得ServletContext = ServletContextEvent.getServletContext();

context-param的值 = ServletContext.getInitParameter("context-param的键");

6.得到这个context-param的值之后,你就可以做一些操作了.注意,这个时候你的WEB项目还没有完全启动完成.这个动作会比所有的Servlet都要早.

换句话说,这个时候,你对<context-param>中的键值做的操作,将在你的WEB项目完全启动之前被执行.

7.举例.你可能想在项目启动之前就打开数据库.

那么这里就可以在<context-param>中设置数据库的连接方式,在监听类中初始化数据库的连接.

8.这个监听是自己写的一个类,除了初始化方法,它还有销毁方法.用于关闭应用前释放资源.比如说数据库连接的关闭.

如:

<!-- 加载spring的配置文件 -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/applicationContext.xml,/WEB-INF/action-servlet.xml,/WEB-

INF/jason-servlet.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

又如: --->自定义context-param,且自定义listener来获取这些信息

<context-param>

<param-name>urlrewrite</param-name>

<param-value>false</param-value>

</context-param>

<context-param>

<param-name>cluster</param-name>

<param-value>false</param-value>

</context-param>

<context-param>

<param-name>servletmapping</param-name>

<param-value>\*.bbscs</param-value>

</context-param>

<context-param>

<param-name>poststoragemode</param-name>

<param-value>1</param-value>

</context-param>

<listener>

<listener-class>com.laoer.bbscs.web.servlet.SysListener</listener-class>

</listener>

public class SysListener extends HttpServlet implements ServletContextListener {

private static final Log logger = LogFactory.getLog(SysListener.class);

public void contextDestroyed(ServletContextEvent sce) {

//用于在容器关闭时,操作

}

//用于在容器开启时,操作

public void contextInitialized(ServletContextEvent sce) {

String rootpath = sce.getServletContext().getRealPath("/");

System.out.println("-------------rootPath:"+rootpath);

if (rootpath != null) {

rootpath = rootpath.replaceAll("\\\\", "/");

} else {

rootpath = "/";

}

if (!rootpath.endsWith("/")) {

rootpath = rootpath + "/";

}

Constant.ROOTPATH = rootpath;

logger.info("Application Run Path:" + rootpath);

String urlrewrtie = sce.getServletContext().getInitParameter("urlrewrite");

boolean burlrewrtie = false;

if (urlrewrtie != null) {

burlrewrtie = Boolean.parseBoolean(urlrewrtie);

}

Constant.USE\_URL\_REWRITE = burlrewrtie;

logger.info("Use Urlrewrite:" + burlrewrtie);

其它略之....

}

}

/\*最终输出

-------------rootPath:D:\tomcat\_bbs\webapps\BBSCS\_8\_0\_3\

2009-06-09 21:51:46,526 [com.laoer.bbscs.web.servlet.SysListener]-[INFO]

Application Run Path:D:/tomcat\_bbs/webapps/BBSCS\_8\_0\_3/

2009-06-09 21:51:46,526 [com.laoer.bbscs.web.servlet.SysListener]-[INFO]

Use Urlrewrite:true

2009-06-09 21:51:46,526 [com.laoer.bbscs.web.servlet.SysListener]-[INFO]

Use Cluster:false

2009-06-09 21:51:46,526 [com.laoer.bbscs.web.servlet.SysListener]-[INFO]

SERVLET MAPPING:\*.bbscs

2009-06-09 21:51:46,573 [com.laoer.bbscs.web.servlet.SysListener]-[INFO]

Post Storage Mode:1

\*/

context-param和init-param区别

web.xml里面可以定义两种参数：

(1)application范围内的参数，存放在servletcontext中，在web.xml中配置如下：

<context-param>

<param-name>context/param</param-name>

<param-value>avalible during application</param-value>

</context-param>

(2)servlet范围内的参数，只能在servlet的init()方法中取得，在web.xml中配置如下：

<servlet>

<servlet-name>MainServlet</servlet-name>

<servlet-class>com.wes.controller.MainServlet</servlet-class>

<init-param>

<param-name>param1</param-name>

<param-value>avalible in servlet init()</param-value>

</init-param>

<load-on-startup>0</load-on-startup>

</servlet>

在servlet中可以通过代码分别取用：

package com.wes.controller;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

public class MainServlet extends HttpServlet ...{

public MainServlet() ...{

super();

}

public void init() throws ServletException ...{

System.out.println("下面的两个参数param1是在servlet中存放的");

System.out.println(this.getInitParameter("param1"));

System.out.println("下面的参数是存放在servletcontext中的");

System.out.println(getServletContext().getInitParameter("context/param"));

}

}

第一种参数在servlet里面可以通过getServletContext().getInitParameter("context/param")得到

第二种参数只能在servlet的init()方法中通过this.getInitParameter("param1")取得.

## contextConfgiLocation

Web.xml中的contextConfgiLocation的作用

如果在web.xml里给该Listener指定要加载的xml，如：

Xml代码如下：

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

classpath:app.xml

</param-value>

</context-param>

则会去加载相应的xml，而不会去加载/WEB-INF/下的applicationContext.xml。

但是，如果没有指定的话，默认会去/WEB-INF/下加载applicationContext.xml。

在spring的项目中，基本上都会在web.xml通过该配置来初始化各个spring的配置文件。

当然还要加上以下配置：

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

## log4jRefreshInterval

值为60000表示 开一条watchdog线程每60秒扫描一下配置文件的变化

<context-param>

<param-name>log4jConfigLocation</param-name>

<param-value>WEB-INF/log4j.properties</param-value>

</context-param>

<context-param>

<param-name>log4jRefreshInterval</param-name>

<param-value>60000</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>

</listener>