# ServletConfig接口详解

## ServletConfig是什么？

jakart.servlet.ServletConfig。显然，ServletConfig是servlet规范的一员，ServletConfig是一个接口。

## 是谁实现了ServletConfig接口？

获取一个ServletConfig以后打印输出可以看到：org.apache.catalina.core.StandardWrapperFacade实现了这个接口。

结论：tomcat服务器实现了ServletConfig接口。

思考：如果把tomcat服务器换成jetty服务器，输出ServletConfig的时候，还会是这个接口吗？

显然不是，jetty服务器有自己的实现ServletConfig接口的类，内容可能类似，但是包名一定不同。

## 多个Servlet对象是否会共享同一个Servlet对象？

不是。每个Servlet对象都有一个自己的ServletConfig对象。（Servlet和ServletConfig是一对一的关系）

## ServletConfig对象时谁在什么时候创建的？

由web服务器（比如tomcat）创建了ServletConfig对象。

web服务器在创建Servlet对象的时候，同时会创建一个ServletConfig对象。

## ServletConfig接口是做什么用的？

config是configuration的缩写，所以是**Servlet对象的配置信息对象**。

## ServletConfig中到底包装了什么信息？

ServletConfig包装的内容主要是对应的Servlet对象在web.xml文件中的对应的servlet标签内容。

tomcat解析web.xml文件，将其中的servlet标签以及内部信息自动包装到ServletConfig中。

## ServletConfig接口中有哪些函数？

web.xml文件中是可以对Servlet配置初始化参数的（parameter）。

通过以下格式，写在<servlet>标签下：

<init-param>

<param-name>参数名</param-name>

<param-value>参数值</param-value>

</init-param>

servlet标签中的内容会被自动封装到ServletConfig对象中，通过ServletConfig中的两个函数，可以获取到web.xml中的初始化参数信息：getInitParameter() 和 getInitParameterNames()。

## 使用GenericServlet代替手动创建ServletConfig

由于GenericServlet实现了ServletConfig接口，所以包含了ServletConfig中所有需要的函数，这样，在继承了GenericServlet之后，就可以直接调用父类中封装的函数，通过this指针调用ServletConfig中的函数。

下面是一段演示代码：

// 实际上，因为GenericServlet实现了ServletConfig接口，所以继承了GenericServlet后，可以直接用this调用上述函数。  
Enumeration<String> paramNames = this.getInitParameterNames();  
while (paramNames.hasMoreElements()) {  
 String paramName = paramNames.nextElement();  
 String paramValue = this.getInitParameter(paramName);  
 System.out.println("配置信息名：" + paramName + "；配置信息值" + paramValue + "<br/>");  
}

# ServletContext接口详解

## 获取ServletContext

两种方法获取ServletContext：

1、通过servletConfig对象可以获取servletContent：·servletConfig.getServletContext()·

2、或者，继承了GenericServlet以后也可以直接：·this.getServletContext()·

## 什么是ServletContext

ServletContext是一个接口，是Servlet规范中的一员。

web服务器使用·org.apache.catalina.core.ApplicationContextFacade·实现了ServletContext接口。

ServletContext对象是web服务器创建的，对于一个webapp来说，ServletContext对象只有一个（可以理解成**ServletContext保存了整个web.xml文件保存的信息**）。

ServletContext对象在服务器关闭的时候销毁。

ServletContext被称为**Servlet上下文/环境对象**。

tomcat是一个web容器，存放了多个webapp，每个webapp都有一个ServletContext对象，用于管理该webapp中的ServletContext。

## ServletContext中的常用函数

*参考帮助文档*，下面是常用的几个函数：

### 获取配置信息

1. String getInitParameter(String name)
2. Enumeration<String> getInitParameterNames();

在web.xml文件中可以设置初始化参数：

<!-- 配置上下文初始化参数。 -->  
<context-param>  
 <param-name>param1</param-name>  
 <param-value>val1</param-value>  
</context-param>

通过Servlet可以获取这些值：

// 通过上下文对象获取初始化参数值。  
Enumeration<String> initParameterNames = servletContext.getInitParameterNames();  
while (initParameterNames.hasMoreElements()) {  
 String name = initParameterNames.nextElement();  
 String initParameter = servletContext.getInitParameter(name);  
 writer.print("<br/>" + initParameter);  
}

**全局配置和局部配置**

对于全局有效的配置，要写在web.xml中的全局位置，对于局部的（就是每个webapp自己的配置）就需要写在<servlet>标签下面。

### 关于路径的几个函数

·ServletContext.getContextath()·用于获取应用的根路径（非常重要），这个函数可以动态获取程序根路径。

·servletContext.getRealPath("/index.html");·用于获取文件的绝对路径。

注意：路径中的“/”表示的是web文件夹的根，即使不加“/”，也会默认从web文件夹开始查找。

### 记录项目日志

通过ServletContext可以记录日志，日志一般会记录到CATALINA\_HOME/logs目录下。

注意：因为使用的是idea，所以不会把日志保存在tomcat文件夹下面的，而是在idea指定的文件架下面产生的。具体路径参考idea启动tomcat的时候提供的CATALINA\_BASE值。

**tomcat服务器的logs文件架下面都有哪些文件**？

一共有三种文件：

1. 服务器端的java程序运行的控制台信息
2. ServletContext对象的log函数存储的文件
3. 访问日志，保存访问记录

### 关于Attribute的函数

ServletContext还有一个称呼叫做**应用域**。后面还有其他的域，比如，请求域、会话域等。

**如果所有的Servlet用户共享同一个数据，并且该数据数据量不多且很少被修改，则可以考虑放入ServletContext的应用域中**。

为什么是所有用户共享的？为什么数据量要小？为什么需要很少修改的数据？

1. ServletContext只有一个，不是共享的数据在这里没有意义。
2. 因为数据量比较大的话，会过于占用堆内存，并且这个对象的声明周期比较长，服务器关闭的时候才会被销毁，从而会影响服务器的性能。
3. 因为所有用户共享的数据如果涉及到修改操作，必然会涉及到线程并发带来的安全问题，所以放在ServletContext中的数据一般都是只读的。

应用域相当于一个缓存，缓存中的数据，下次要用的时候不需要从数据库中获取，从而提高了执行效率。

**如何向ServletContext应用域中存取和删除数据**？

使用以下函数即可：

// 向ServletContext应用域中存数据  
// 准备数据  
User user = new User("user\_name", "123");  
// 存数据  
servletContext.setAttribute("key1", user); // 存入数据  
// 获取AServlet中保存的数据  
Object user\_obj = servletContext.getAttribute("key1"); // 读取数据  
writer.print("<br/>" + user\_obj); // 打印数据

# 以后不会再用Servlet类直接继承GenericServlet类

因为我们使用的是BS结构的系统，这种系统是基于HTTP超文本传输协议的，在Servlet规范中提供了一个类叫做HttpServlet，是为HTTP专门准备的Servlet类，我们写的Servlet将会继承HTTPServlet。（因为他是HTTP协议专用的，所以使用起来处理HTTP协议更加方便）。

但是需要知道他的继承结构：

jakarta.Servet接口有一个实现类叫做GenericServlet（抽象类），在jakarta.http包下有另一个抽象类HttpServlet，继承了GenericServlet类。

以后编写的Servlet要继承HttpServlet。