# 12.ServletContext

## 12.1.ServletContext概述

代表整个WEB应用的对象

在WEB应用被服务器加载之后, 服务器会立即创建代表当前WEB应用的ServletContext对象, 创建之后ServletContext对象会一直驻留在服务器的内存中, 用来唯一的表示当前WEB应用.直到服务器关闭或者是WEB应用被移出容器时为止, 随着WEB应用的销毁, 代表WEB应用的ServletContext对象也会跟着销毁!!

**获取ServletContext对象**

this.getServletConfig().getServletContext();

或者

this.getServletContext();

## 12.2.ServletContext功能

### 12.2.1.获取web应用的初始化参数

在web.xml文件中可以为每一个Servlet配置初始化参数, 配置的参数只能在当前Servlet内部通过ServletConfig对象来获取, 在别的Servlet中无法获取!

如果现有一段配置信息, 不是属于某一个Servlet, 而是希望在整个WEB应用中都可以获取到, 可以将这段信息配置给整个WEB应用, 再通过代表WEB应用的ServletContext对象来获取这些信息!

配置初始化参数的方式:

<context-param>

<param-name>scparam1</param-name>

<param-value>scvalue1</param-value>

</context-param>

...

获取初始化参数的方式:

getInitParameter();

getInitParameterNames();

**代码示例：**创建包cn.tedu.context，创建servlet：ServletContextDemo1

|  |
| --- |
| 配置初始化参数的方式:  在web.xml中的<web-app>根标签下，加入如下代码：  <context-param>  <param-name>scparam1</param-name>  <param-value>scvalue1</param-value>  </context-param>  <context-param>  <param-name>scparam2</param-name>  <param-value>scvalue2</param-value>  </context-param> |
| 获取初始化参数的方式:  在ServletContextDemo1中加入如下代码：  //1.获取ServletContext对象  //方式一：  ServletContext context = **this**.getServletConfig().getServletContext();  //方式二：  ServletContext context2 = **this**.getServletContext();    //2.获取web应用的初始化参数参数  String v1 = context.getInitParameter("scparam1");  String v2 = context.getInitParameter("scparam2");    System.*out*.println("v1:"+v1);  System.*out*.println("v2:"+v2); |
| 也可以在其他servlet或jsp中通过ServletContext 获取参数。 |

### 12.2.2.作为域对象来使用

如果一个对象具有一个可以被看见的范围, 利用该对象上的map可以在整个范围内实现数据的共享!

ServletContext对象是一个域对象, 利用这个对象上的map就可以在整个web应用内实现资源的共享.

域对象提供的方法:

setAttribute(String name, Object obj);//添加或修改域属性

getAttribute(String name);//通过属性名获取属性值

removeAttribute(String name);//根据属性名删除属性

getAttributeNames();//获取所有域属性名字组成的枚举

域对象的特征:

生命周期: 和WEB应用的命一般长

作用范围: 整个WEB应用

主要功能: 在整个WEB应用范围内实现数据的共享

**代码示例：**

|  |
| --- |
| 在ServletContextDemo1中加入如下代码：  //往ServletContext域中存入一些数据  context.setAttribute("name", "张三");  context.setAttribute("addr", "北京");  context.setAttribute("age", "18");  context.setAttribute("gender", "男"); |
| 在index.jsp中将数据取出来并展示：在index.jsp中，添加如下代码：  注意：需要将编码改为utf-8。  <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"utf-8"*%>  <!DOCTYPE HTML>  <html>  <head>  <title>My JSP 'index.jsp' starting page</title>  <meta http-equiv=*"pragma"* content=*"no-cache"*>  <meta http-equiv=*"cache-control"* content=*"no-cache"*>  <meta http-equiv=*"expires"* content=*"0"*>  </head>  <body>  <%  ServletContext context = **this**.getServletContext();  String name = (String)context.getAttribute("name");  String addr = (String)context.getAttribute("addr");  String age = (String)context.getAttribute("age");  String gender = (String)context.getAttribute("gender");  %>    姓名：<%= name %> <br>  地址：<%= addr %> <br>  年龄：<%= age %> <br>  性别：<%= gender %> <br>    </body>  </html> |
| 注：在jsp中，<% %>中可以写多行java代码  <%= %>中可以写表达式/常量/变量，一般一行或者一个值。 |

### 12.2.3.获取web资源文件

**1.在web开发中, 当加载配置文件需要写一个路径时:**

1. **如果写一个相对路径:**

例如：String path = "conf.properties";

//加载路径为：E:\software\tomcat7.0\bin\conf.properties

在通过相对路径加载资源文件时, 程序会到tomcat的bin目录寻找该配置文件, 但是找不到!!如果是相对路径, 会到程序启动的目录下寻找该文件, 由于WEB应用是运行在tomcat服务器, 所以程序启动的目录就是tomcat服务器的启动目录, 就是bin目录!

1. **如果写一个绝对路径:**

例如：String path = "/conf.properties";

// 加载路径为：E:\conf.properties

在通过绝对路径加载资源文件时, 这时会到WEB应用所在路径的硬盘根目录去寻找该文件, 但是也找不到。

**（3）可以写一个带盘符的绝对路径:**

例如：String path = "E:/software/tomcat7.0/webapps/day11/WEB-INF/classes/conf.properties";

//可以正确加载到配置文件

在通过带盘符的绝对路径寻找文件时, 可以找到文件, 但是这样写路径把路径写死在程序中了,如果后期换一个发布环境, 路径很可能是错误的, 所以这样写路径也不可取!!

**代码示例：**

|  |
| --- |
| 在src下创建conf.properties配置文件，加入配置信息，例如：  driverClass=com.mysql.jdbc.Driver |
| 在context包中创建servlet：ServletContextDemo2，并加入如下代码：  Properties prop = **new** Properties();  //获取conf.properties文件的路径  //相对路径  //String path = "conf.properties";  //绝对路径  //String path = "/conf.properties";  //带盘符的绝对路径  String path = "E:/software/tomcat7.0/webapps/day11/WEB-INF/classes/conf.properties";    //获取path绝对路径  File file = **new** File(path);  System.*out*.println("path:"+file.getAbsolutePath());    //加载配置文件，如果路径有误，则页面会报错  prop.load(**new** FileInputStream(path)); |

**2.可以通过ServletContext对象提供的方法来获取WEB资源文件**

servletContext对象提供的getRealPath方法是基于WEB应用的根路径(/day11)去寻找该配置文件, 所以在getRealPath方法中还需要传入文件相对于WEB应用根路径的路径!

方法为：servletContext.getRealPath(String path);

例如：配置文件路径为：

E:/software/tomcat7.0/webapps/day11/WEB-INF/classes/conf.properties

则方法应写为：

this.getServletContext().getRealPath("/WEB-INF/classes/conf.properties");

**3.也可以通过类加载器来获取WEB资源文件**

类加载器提供的getResource方法是基于WEB应用的类目录(/day11/WEB-INF/classes)去寻找该配置文件, 所以在getResource方法中还需要传入文件相对于classes目录的路径!

方法为：classLoader.getResource(String path).getPath();

例如：配置文件路径为：

E:/software/tomcat7.0/webapps/day11/WEB-INF/classes/conf.properties

则方法应写为：

ServletContextDemo2.class.getClassLoader().getResource("conf.properties").getPath();

**代码示例：**

|  |
| --- |
| 在ServletContextDemo2中，获取路径代码中添加如下代码：  //通过ServletContext对象的getRealPath()方法获取配置文件路径  //String path = this.getServletContext().getRealPath("/WEB-INF/classes/conf.properties");    //通过类加载器获取资源文件  String path = ServletContextDemo2.**class**.getClassLoader().getResource("conf.properties").getPath(); |

## 12.3.作业练习

1. ServletConfig是什么？

答：用来代表Servlet在web.xml中配置信息的对象，可以通过this.getServletConfig()方法来获取此对象，通过这个对象可以获取当前servlet在web.xml中的初始化参数，也可以获取ServletContext对象。

1. 简述ServletContext生命周期？

答：当服务器启动时，会依次加载web应用，每一个web应用加载完成后都会创建一个ServletContext对象唯一代表该web应用，这个对象一直存活，直到web应用移除出容器或服务器关闭时，随着应用销毁，ServletContext对象跟着销毁。

1. ServletContext的作用范围是什么？

答：整个WEB应用