

* 学习目标

* 能够掌握Servlet映射配置

- * `/hello-->/action/* --> *.do`

- * `load-on-startup -->` 服务器启动的时候-->Servlet就会创建

* 能够理解Servlet的线程安全问题

- * 由于Servlet是单实例的：访问成员变量的时候，会有线程安全问题

- * 尽量写局部变量，避免写成员变量

- * 线程安全技术：加锁，CAS，volatile

* 能够掌握ServletConfig的使用

- * `serlvet` 配置初始化参数

* 能够掌握ServletContext的使用

- * 提供一些方法：Servlet 与 Servlet 通讯

- * `contextPath,serverInfo`,相对路径获得绝对路径，通过文件名获得MIME类型，获得输入流

- * 获得配置参数

- * 请求转发，设置属性

* 回顾

* 编写简单Web服务器

- * `ServerSocket`---获得请求路径---写协议---写内容--释放资源---多线程

* Tomcat

- * Apache 服务器：J2EE定下的规范，Java Servlet,JSP,EL,WebSocket,...

* 下载，安装，idea使用

- * 导出war包，在tomcat部署war包，设置默认首页，默认端口，默认应用，域名

* HTTP 协议：超文本传输协议，应用层

- * 特点：简单快速，灵活，无状态，无连接，支持C/S,B/S

- * URL和URI

- * <http://localhost:8080/web/index.html>

- * Http主要协议的内容

- * request method , url , status code:200,404,503

- * accept,accept-encoding,...content-type

- *

- * Servlet

- * 服务端小程序，处理HTTP协议相关的请求和响应

- * 创建一个Servlet

- * 直接实现Servlet接口

- * 继承 GenericServlet类--service

- * 继承 HttpServlet --- doGet,doPost

- * 配置：

- XML：

- ```
<servlet>
```

- ```
    <servlet-name>abc</servlet-name>
```

- ```
 <servlet-class>HelloWorld</servlet-class>
```

- ```
</servlet>
```

- ```
<servlet-mapping>
```

- ```
    <servlet-name>abc</servlet-name>
```

- ```
 <url-pattern>/test</url-pattern>
```

- ```
</servlet-mapping>
```

- 注解：

- ```
@WebServlet(value="/hello")
```

- \* Servlet 的生命周期

- \* 构造器--init--service(多次调用)--destroy

- \* 单实例

- \* 能够掌握Servlet映射配置

\* Servlet的细节

\* 映射细节-配置多个映射路径

```
<servlet-mapping>
 <servlet-name>hello5</servlet-name>
 <url-pattern>/hello5</url-pattern>
</servlet-mapping>
<servlet-mapping>
 <servlet-name>hello5</servlet-name>
 <url-pattern>/hello567</url-pattern>
</servlet-mapping>
```

都是访问同一Servlet

```
1 <servlet-mapping>
2 <servlet-name>hello5</servlet-name>
3 <url-pattern>/hello5</url-pattern>
4 </servlet-mapping>
5 <servlet-mapping>
6 <servlet-name>hello5</servlet-name>
7 <url-pattern>/hello567</url-pattern>
8 </servlet-mapping>
```

\* 映射细节-通配符\* (代表任意字符串)

\*.do

```
<servlet>
 <servlet-name>hello2</servlet-name>
 <servlet-class>com.etc.servlet.HelloServlet2</servlet-class>
</servlet>
<servlet-mapping>
 <servlet-name>hello2</servlet-name>
 <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

访问Servlet的时候，之后后缀是.do都可以：xx.do,abc.do

```
1 <servlet>
2 <servlet-name>hello2</servlet-name>
3 <servlet-class>com.etc.servlet.HelloServlet2</servlet-class>
4 </servlet>
```

```

5 <servlet-mapping>
6 <servlet-name>hello2</servlet-name>
7 <url-pattern>*.do</url-pattern>
8 </servlet-mapping>

```

```

/*
</servlet-mapping>
<servlet>
 <servlet-name>hello2</servlet-name>
 <servlet-class>com.etc.servlet.HelloServlet2</servlet-class>
</servlet>
<servlet-mapping>
 <servlet-name>hello2</servlet-name>
 <url-pattern>/*</url-pattern>
</servlet-mapping>

```

在上下文下，输入任意路径，都可以访问，但是优先级比较低

```

1 <servlet>
2 <servlet-name>hello2</servlet-name>
3 <servlet-class>com.etc.servlet.HelloServlet2</servlet-class>
4 </servlet>
5 <servlet-mapping>
6 <servlet-name>hello2</servlet-name>
7 <url-pattern>/*</url-pattern>
8 </servlet-mapping>

```

/action/\*

```

<servlet-mapping>
 <servlet-name>hello3</servlet-name>
 <url-pattern>/action/*</url-pattern>
</servlet-mapping>

```

/action/xx, 都可以访问

\* 匹配规则细节(优先级)

\* 优先级：从高到低

\* 绝对匹配-->/开头匹配 --> 扩展名方式匹配

/hello      /\*      \*.do

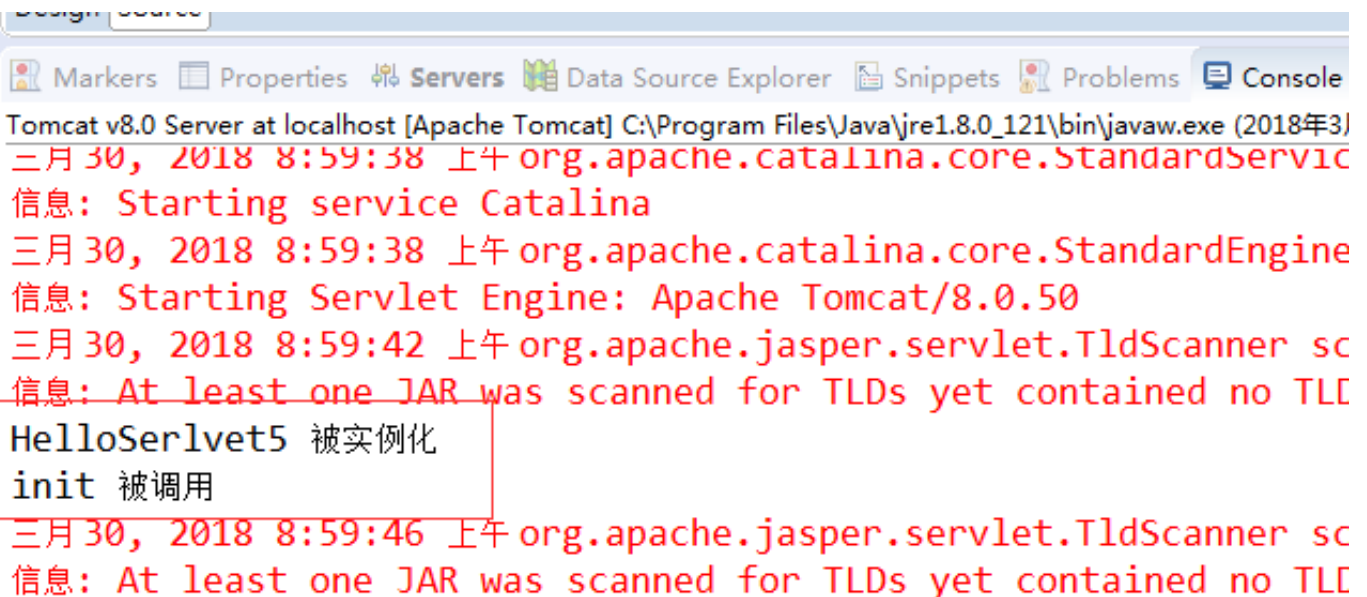
\* load-on-startup

```
</servlet-mapping>
```

```
<servlet>
 <description></description>
 <display-name>hello5</display-name>
 <servlet-name>hello5</servlet-name>
 <servlet-class>com.etc.servlet.HelloServlet5</servlet-class>
 <load-on-startup>1</load-on-startup>
</servlet>
```

大于或等于0的时候，Servlet在tomcat启动时候，会被实例化并且调用init的方法

```
1 <servlet>
2 <description></description>
3 <display-name>hello5</display-name>
4 <servlet-name>hello5</servlet-name>
5 <servlet-class>com.etc.servlet.HelloServlet5</servlet-class>
6 <load-on-startup>1</load-on-startup>
7 </servlet>
```



Tomcat v8.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0\_121\bin\javaw.exe (2018年3月30, 2018 8:59:38 上午 org.apache.catalina.core.StandardServiceManager: Starting service Catalina  
三月 30, 2018 8:59:38 上午 org.apache.catalina.core.StandardEngine: Starting Servlet Engine: Apache Tomcat/8.0.50  
三月 30, 2018 8:59:42 上午 org.apache.jasper.servlet.TldScanner: Scanning for TLDs  
信息: At least one JAR was scanned for TLDs yet contained no TLDs  
HelloServlet5 被实例化  
init 被调用  
三月 30, 2018 8:59:46 上午 org.apache.jasper.servlet.TldScanner: Scanning for TLDs  
信息: At least one JAR was scanned for TLDs yet contained no TLDs

能够理解Servlet的线程安全问题

\* Servlet单实例，多线程访问存在线程安全的问题。

解决方案：

\* 尽量避免使用成员变量，改用局部变量

\* 加锁

\* 能够掌握ServletConfig的使用



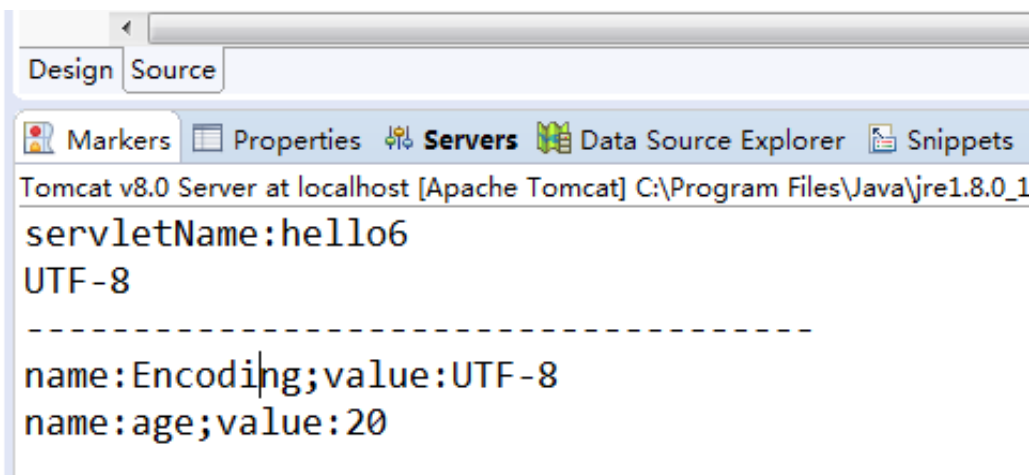
```
1 * 获得ServletConfig对象
2 * 方式一
3 private ServletConfig config;
4 @Override
5 public void init(ServletConfig config) throws ServletException {
6 super.init(config);
7 this.config=config;
8 }
9 * 方式二
10 * getServletConfig()
11
12 * 配置文件
13 <servlet>
14 <description></description>
15 <display-name>hello6</display-name>
16 <servlet-name>hello6</servlet-name>
17 <servlet-class>com.lg.servlet.HelloServlet6</servlet-class>
18 <init-param>
19 <param-name>Encoding</param-name>
20 <param-value>UTF-8</param-value>
21 </init-param>
22 <init-param>
23 <param-name>age</param-name>
24 <param-value>20</param-value>
25 </init-param>
```

```

26 </servlet>
27 <servlet-mapping>
28 <servlet-name>hello6</servlet-name>
29 <url-pattern>/hello6</url-pattern>
30 </servlet-mapping>
31
32 * 代码获取
33 protected void doGet(HttpServletRequest request, HttpServletResponse response)
34 //1 获取Servlet的名字
35 String servletName = config.getServletName();
36 System.out.println("servletName:"+servletName);
37 //2 根据名字获得参数信息
38 String param = config.getInitParameter("Encoding");
39 System.out.println(param);
40 System.out.println("-----");
41 //3 获取参数名字的集合，再通过这些名字获取参数
42 Enumeration<String> names = config.getInitParameterNames();
43 while(names.hasMoreElements()) {
44 String name=names.nextElement();
45 String value=config.getInitParameter(name);
46 System.out.println("name:"+name+";"+"value:"+value);
47 }
48 response.getWriter().append("Served at: ").append(request.getContextPat
49 }
50

```

测试结果：



```

Design Source
Markers Properties Servers Data Source Explorer Snippets
Tomcat v8.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_1
servletName:hello6
UTF-8

name:Encoding;value:UTF-8
name:age;value:20

```

## \* 能够掌握ServletContext的使用



```
1 * 案例一：
2 protected void doGet(HttpServletRequest request, HttpServletResponse response)
3 // 1 获取上下文
4 ServletContext context=getServletContext();
5 //2 获取上下文路径,获取上下文名字,获取服务器的信息
6 String contextPath = context.getContextPath();
7 String serverInfo=context.getServerInfo();
8 System.out.println(contextPath);
9 System.out.println(serverInfo);
10 response.getWriter().append("Served at: ").append(request.getContextPat
11 }
12 * 案例二
13 //3 根据相对路径获取绝对路径,传递一个文件路径获取MIME类型,根据路径(相对路径)获取文件输
14 String path="/WEB-INF/web.xml";
15 // 根据相对路径获取绝对路径
```

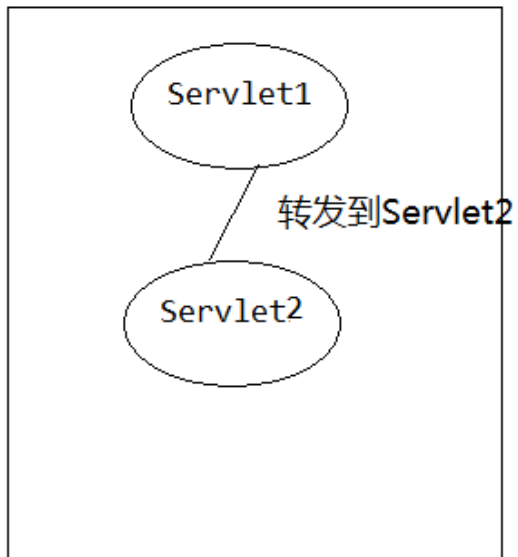


```

16 String absolutePath=context.getRealPath(path);
17 String path2=context.getRealPath("/hello.json");
18 System.out.println(absolutePath);
19 //传递一个文件路径获取MIME类型
20 String mimeType = context.getMimeType(absolutePath);
21 String mimeType2 = context.getMimeType(path2);
22 System.out.println(mimeType);//application/xml
23 System.out.println(mimeType2);//application/json
24 InputStream is = context.getResourceAsStream(path);// 而且这个路径是相对
25 IOUtils.copy(is, System.out);
26
27 * 案例三:
28 * 全局的配置
29 <context-param>
30 <param-name>bookName</param-name>
31 <param-value>Java in thinking</param-value>
32 </context-param>
33 <context-param>
34 <param-name>test</param-name>
35 <param-value>helloworld</param-value>
36 </context-param>
37 * 代码
38 String contextParam = context.getInitParameter("bookName");
39 System.out.println(contextParam);
40 Enumeration<String> paramNames = context.getInitParameterNames();
41 while(paramNames.hasMoreElements()) {
42 String name=paramNames.nextElement();
43 String value=context.getInitParameter(name);
44 System.out.println(name+":"+value);
45 }
46
47 案例四:
48 //1设置在Servlet1某个属性
49 context.setAttribute("age", 20);
50 //2 Servlet1 转发跳转Servlet2
51 context.getRequestDispatcher("/hello8").forward(request, response);
52
53 //3 在Servlet2获取这个属性
54 ServletContext context = getServletContext();
55 int age = (int) context.getAttribute("age");

```

```
56 System.out.println("age:"+age);
57 response.getWriter().append("age:"+age);
```



ServletContext

- 1 Servlet1 跳转到Servlet2
- 2 在Servlet1设置属性
- 3 在Servlet2中获取这个属性