Web 组件交互

学习目标

- [ ] 能够说出 EL 表达式的作用并能够使用 EL 表达式
- [ ] 能够说出 JSTL 标签库的作用
- [ ] 能够使用 JSTL 标签库的 if foreach choose 标签
- [ ] 能够使用转发和重定向实现页面跳转
- [ ] 能够往三大作用域中存取数据

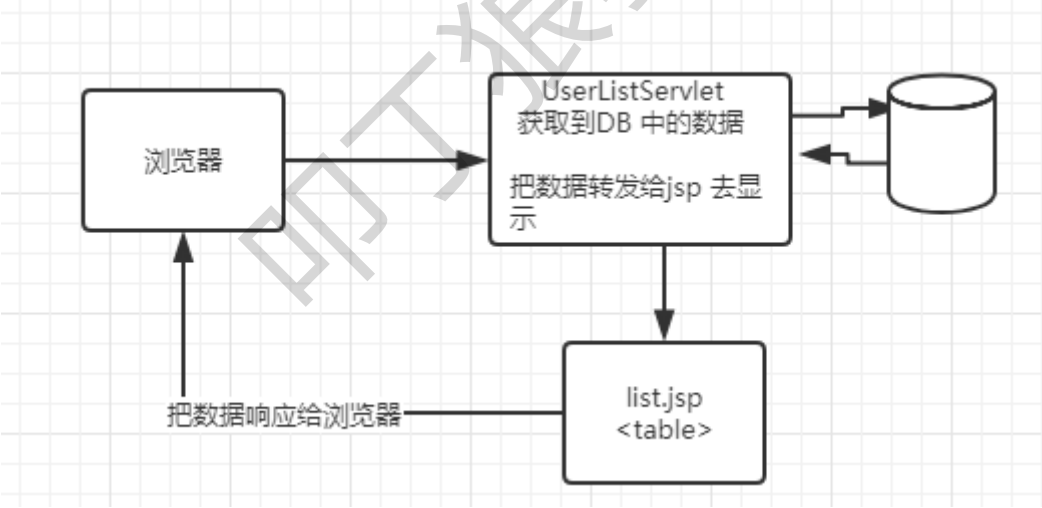
第一章 : 为什么要做跳转和数据共享

为了让 Jsp 和 Servlet 更加的责任分离,去做自己擅长的事情.所以以后的数据获取,

1 Servlet 去数据库中查找

2 Servlet 把数据给 jsp

3 jsp 把数据响应给浏览器



叮当狼教育

第二章 转发和重定向

2.1 转发和重定向的作用

用于web 组件的跳转，从A 组件跳到另一个组件

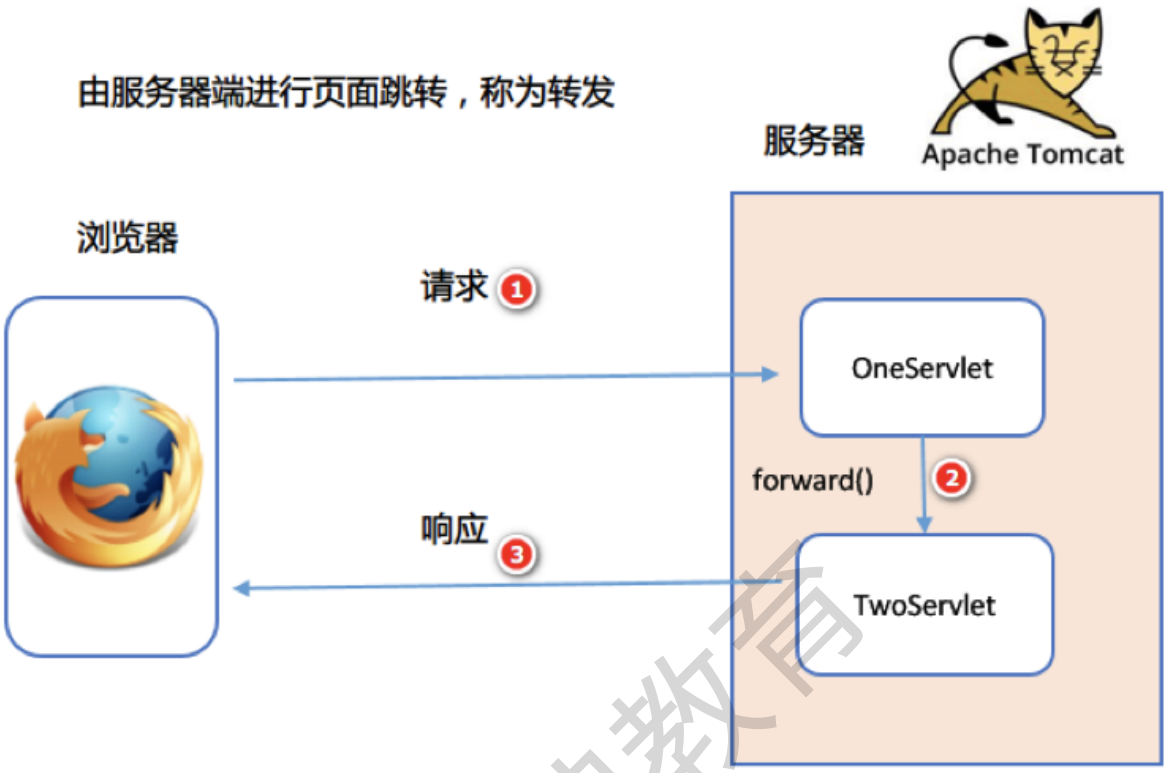
目前学到的 web 组件 : JSP, Servlet

2.2 转发概述

2.2.1 跳转位置

- 在服务器端进行的组件(资源)的跳转

2.2.2 转发原理



2.2.3 转发方法

request 对象与转发相关方法
request.getRequestDispatcher("/要跳转到的地址").forward(request, response);

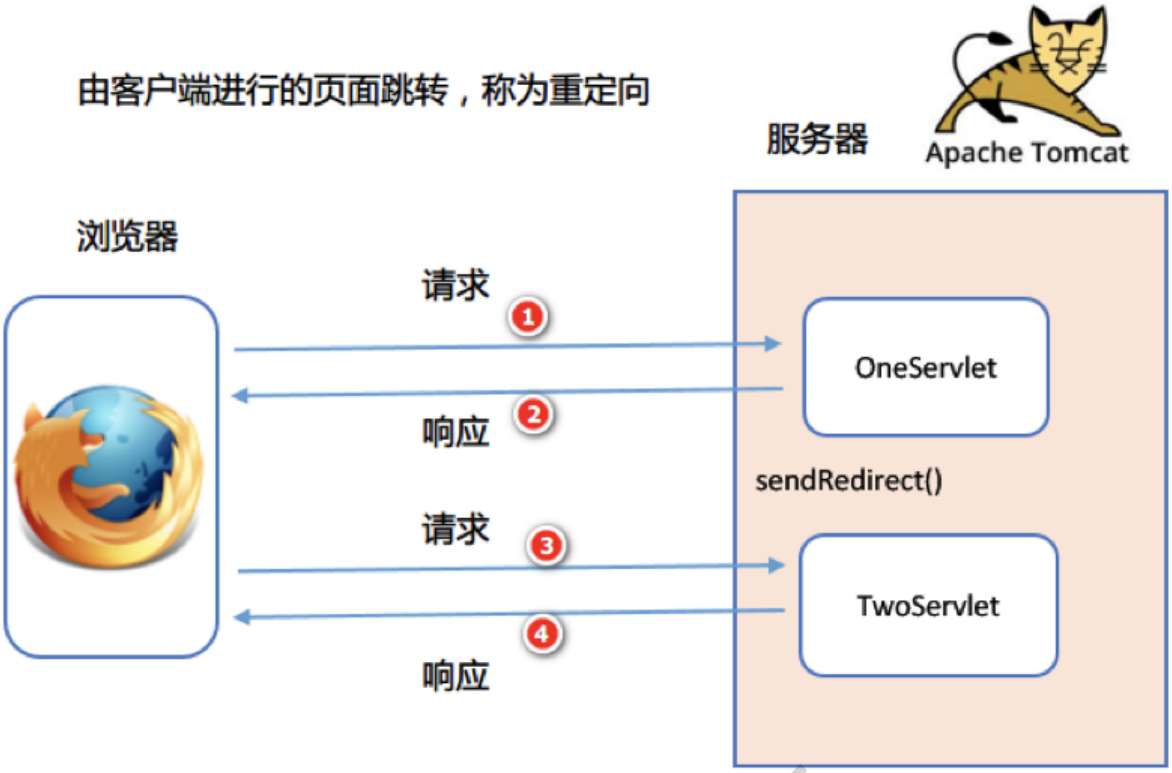
2.3 重定向概述

2.3.1 跳转位置

- 在浏览器端进行的页面(组件/资源)跳转

2.3.2 重定向原理

叮丁狼教育



2.3.2 重定向方法

response对象与重定向相关方法
response.sendRedirect("要跳转的地址")

2.4 转发和重定向代码

2.4.1 需求说明

创建 OneServlet 和 TwoServlet，实现浏览器访问 OneServlet，使用转发或重定向跳转到 TwoServlet

2.4.2 OneServlet 代码

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/one")
public class OneServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String username = req.getParameter("username");
        System.out.println(username);

        // 转发到 twoServlet
         req.getRequestDispatcher("/forward/two").forward(req,resp);
        // 可以跳转到 WEB-INF 中
        // req.getRequestDispatcher("/WEB-INF/hello.jsp").forward(req,resp);
```

叩丁狼教育

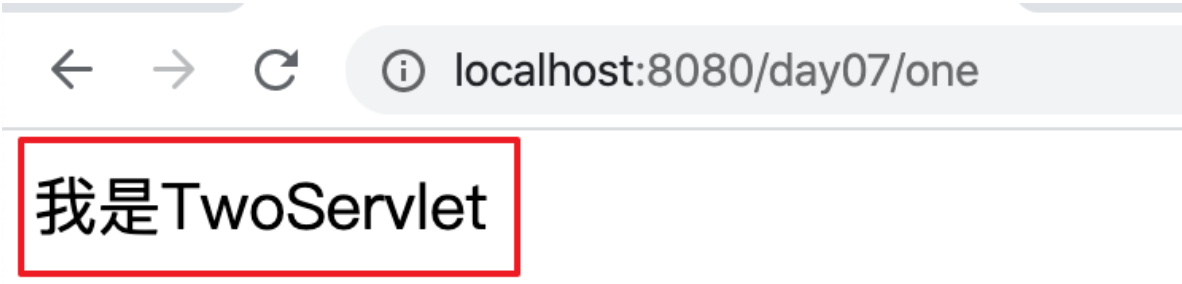
```
        // 不能访问到别人的项目(外域)
        //req.getRequestDispatcher("http://www.baidu.com").forward(req,resp);


        // 重定向到 twoServlet
        // resp.sendRedirect("/day10/redirect/two");
        // 不可以访问 WEB-INF 下的资源
        // resp.sendRedirect("/day10/WEB-INF/hello.jsp");
        // 可以访问其他项目(外域)
        //resp.sendRedirect("http://www.baidu.com");
        System.out.println("后面的代码可以执行到,但是没有意义,以后不会在跳转之后写代
码");
    }
}
```

2.4.3 TwoServlet代码

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

@WebServlet(urlPatterns = "/two")
public class TwoServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // 设置内容类型和编码
        response.setContentType("text/html;charset=utf-8");
        // 获得字符打印流
        PrintWriter out = response.getWriter();
        out.print("我是TwoServlet");
    }
}
```

叩丁狼教育

- 浏览器访问 OneServlet 的效果：



2.5 转发和重定向区别

区别	转发 forward	重定向 redirect
目录	服务端的根目录：`http://localhost:8080/项目地址/` 跳转的path 不用写项目路径(虚拟目录)	浏览器的根目录(端口之后):`http://localhost:8080` 跳转path需加上项目路径(虚拟路劲)
地址栏	地址栏不会发生变化，还是上一个地址	会变化，显示新的地址
跳转位置	在服务端进行跳转	在浏览器端跳转
请求对象(域)	请求域数据不会丢失，因为是同一个请求	请求域数据会丢失，因为不是同一个请求

2.6 常见面试题

- [ ] 什么时候使用转发，什么时候使用重定向？

```
如果需要保留请求域中的数据，使用转发
如果要跳转到 WEB-INF 目录中的页面（资源），使用请求转发
如果要访问其他项目，使用重定向
其他的任选
```

- [ ] 转发或重定向后续的代码是否还会运行？

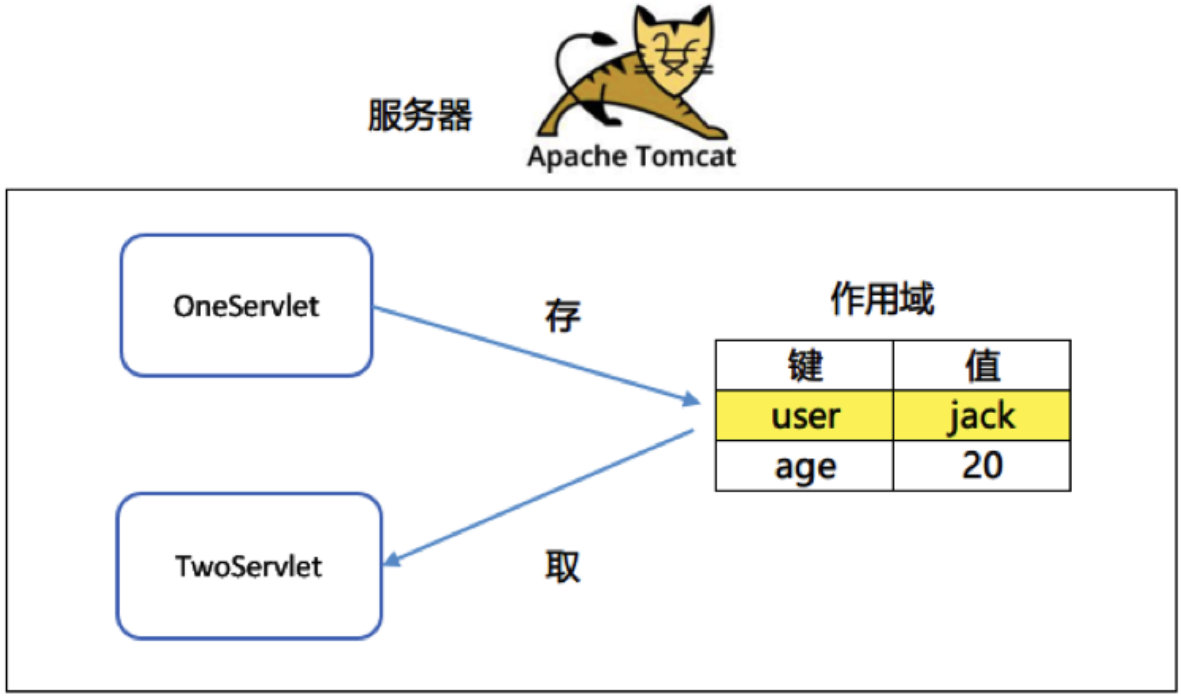
```
无论是重定向还是转发，后续代码都会执行，但一般转发或重定向后面不会有代码，没有执行的意义了，因为不管是转发还是重定向浏览器最终显示的都是跳转后页面的数
```

码丁狼教育

第三章 三大作用域

3.1 什么是作用域

用于 Servlet 之间数据共享的服务器内存区域，作用域结构是一个Map<String, Object>



3.2 作用域类型

作用域	类型	作用范围
请求域	HttpServletRequest	只在同一次请求中起作用
会话域	HttpSession	同一个会话中起作用 浏览器第1次访问服务器直到浏览器关闭 的整个过程称为1次会话
上下文域	ServletContext	同一个应用中起作用 服务器启动直到服务器关闭的整个过程都起作用

尚硅谷教育

3.3 作用域方法

与作用域有关的方法	作用
Object getAttribute("键")	从中得到一个值
void setAttribute("键",Object数据)	向作用域中存储键值对数据
void removeAttribute("键")	删除作用域中的键值对数据

- 哪个作用域对象调用方法就操作对应的作用域数据。

3.4 作用域操作

3.4.1 需求

在 FirstServlet 存储键值到作用域中，转发到另一个 SecondServlet，从 SecondServlet 中取出键值并且输出。

3.4.2 FirstServlet代码

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/scope/first")
public class FirstServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // 往请求域存储键值对
        request.setAttribute("name", "request");
        // 往会话域存储键值对
        request.getSession().setAttribute("name", "session");
        // 往上下文域存储键值对
        request.getServletContext().setAttribute("name", "servletContext");
        // 转发到second
        request.getRequestDispatcher("/scope/second").forward(request,
response);
    }
}
```

3.4.3 SecondServlet代码

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

@WebServlet("/scope/second")
public class SecondServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // 设置内容类型和编码
        response.setContentType("text/html;charset=utf-8");
        // 获得字符打印流
        PrintWriter out = response.getWriter();

        // 从请求域中获取数据
        Object name1 = request.getAttribute("name");
        // 从会话域中获取数据
        Object name2 = request.getSession().getAttribute("name");
        // 从上下文域中获取数据
        Object name3 = request.getServletContext().getAttribute("name");
```

叮丁狼教育

```
            out.print("请求域数据：" + name1 + "<br>");
            out.print("会话域数据：" + name2 + "<br>");
            out.print("上下文域数据：" + name3 + "<br>");
        }
    }
```

- 浏览器访问FirstServlet效果：

作用域范围演示案例

需求: 从 scopeServlet 中获取三个作用域中的数据 num, 如果获取到则加1再存入,如果获取不到则存入 num 的值为1,最后跳转到 ResultServelt 中来做数据的获取和显示.

ScopeServlet.java

```
@WebServlet("/scope")
public class ScopeServlet extends HttpServlet {
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        // 在 ScopeServlet 中获取三个作用域中的数据 num, 如果获取到则加1再存入,
        // 如果获取不到则存入 num 的值为1,跳转到 ResultServelt 中来做数据的获取和显示
        Object strNum = req.getAttribute("num");
        if(strNum == null){
            req.setAttribute("num",1);
        }else{
            Integer num = (Integer)strNum;
            req.setAttribute("num",num + 1);
        }

        // session作用域
        HttpSession session = req.getSession();
        Object sessinoNum = session.getAttribute("num");
        if(sessinoNum == null){
            session.setAttribute("num",1);
        }else{
            Integer num = (Integer)sessinoNum;
            session.setAttribute("num",num + 1);
        }

        // servletContext 作用域
        ServletContext context = super.getServletContext();
        Object ctNum = context.getAttribute("num");
        if(ctNum == null){
            context.setAttribute("num",1);
```

北京狼教育

```
        }else{
            Integer num = (Integer)ctNum;
            context.setAttribute("num",num + 1);
        }

        // 跳转
        req.getRequestDispatcher("/result").forward(req,resp);

    }
}
```

ResultServlet.java

```
@WebServlet("/result")
public class ResultServlet extends HttpServlet {
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        resp.setContentType("text/html;charset=utf-8");
        PrintWriter out = resp.getWriter();

        Object reqNum = req.getAttribute("num");
        Object sessionNum = req.getSession().getAttribute("num");
        Object ctNum = req.getServletContext().getAttribute("num");

        out.println("<br/> request:" + reqNum);
        out.println("<br/> session:" + sessionNum);
        out.println("<br/> servletContext:" + ctNum);

    }
}
```

叮丁狼教育

3.5 如何选择作用域

- 先考虑作用范围小的作用域，如果小作用范围的作用域能满足需求就使用小作用范围的作用域。
- 考虑顺序：请求域-->会话域-->上下文域

第四章 EL表达式

4.1 EL表达式概述

EL 表达式

EL（Expression Language） 是为了使JSP写起来更加简单。表达式语言的灵感来自于 ECMAScript 和 XPath 表达式语言，它提供了在 JSP 中简化表达式的方法，让Jsp的代码更加简化。

- 用于替换 JSP 页面中的脚本表达式，让 JSP 的代码更加简化。

EL概念	EL语法	EL作用
英文名称： Expression Language 中文名称：表达式语言	**${** 变量名或表达式 **}**	1.用于输出**作用域中**变量值 2.用于各种运算：算术，逻辑， 关系，三元运算等

- EL 与 JSP 脚本表达式的区别

区别	JSP脚本表达式	EL
语法	<%=m%>	${m}
输出哪里的值	输出的是脚本变量 <% int m= 5; %>	输出的是作用域中的值 <% request.setAttribute("m", 5); %>

- 案例：分别输出脚本变量和作用域中变量值

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%
    int m=5;
    // 向请求域中添加一个变量
    request.setAttribute("num", 6);
%>
输出m:
<%=m%><br>
输出作用域中值:
<%= request.getAttribute("num")%><br>

使用 EL 表达式输出:
${num}

</body>
</html>
```

叮丁狼教育

4.2 EL 表达式获取不同类型数据

> 从 EL 中取值的前提：数据必须放在作用域中。

4.2.1 示例执行效果

4.2.2 示例代码

```
package cn.wolfcode.entity;

@Getter
@Setter
@ToString
public class Emp {
    private String name;
    private BigDecimal salary;
}
```

```
<%@ page import="cn.wolfcode.entity.Emp" %>
<%@ page import="java.util.Arrays" %>
```

```jsp
<%@ page import="java.util.List" %>
<%@ page import="java.util.HashMap" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>EL得到不同类型的值</title>
</head>
<body>
    <%
        // 将员工对象放在页面域
        Emp emp = new Emp("西施", 10000);
        // 放到页面域中
        pageContext.setAttribute("emp", emp);
    %>

    <h2>得到JavaBean的属性值</h2>
    <%-- 本质上是调用 get方法 --%>
    姓名：${emp.name}， 工资：${emp.salary}
    <hr>

    <%
        List<String> names = Arrays.asList("孙悟空", "猪八戒","沙僧");
        // 放在请求域中
        request.setAttribute("names",names);
    %>

    <h2>取出集合List中元素</h2>
    ${names[0]}     ${names[1]}

    <%
        int[] nums = {100,200,300,54};
        // 数组放在请求域
        request.setAttribute("nums", nums);
    %>
    <h2>取出数组中元素</h2>
    ${nums[0]}    ${nums[3]}

    <%
        Map<String, String> map = new HashMap<>();
        map.put("no1", "牛魔王");
        map.put("no2", "红孩儿");
        map.put("no3-no4", "铁扇公主");
        //放在请求域
        request.setAttribute("map",map);
    %>
    <h2>取出 Map 中元素</h2>
    <%--map对象.键名--%>
    ${map.no1}     ${map.no2}   ${map["no3-no4"]}
</body>
</html>
```

叩丁狼教育

4.2.3 注意事项

- JavaBean的属性名或map的键名中如果有特殊字符的写法

变量名["键名"]
可以使用双引号或单引号，如：${map["no3-no4"]}

4.3 EL 中使用表达式

4.3.1 算术表达式

算术运算符	说明	范例	结果
+	加	${1+1}	2
-	减	${2-1}	1
*	乘	${1*1}	1
/或 div	除	${5 div 2} 或 ${5 /2 }	2.5
%或mod	取余	${5 mod 2} 或 ${5%2}	1

> Tip：在EL表达式中执行算术运算，只要字符串是数值都会执行算术运行。

```
<%
request.setAttribute("a", "20");
request.setAttribute("b", "20");
%>
${a+b}
${a*b}
${a/b}
${a%b}
${1 + 1} , ${2 - 1}
```

4.3.2 比较表达式

关系运算符	说明	范例	结果
==或 eq	等于(equal)	${1 eq 1}	true
!= 或 ne	不等于(not equal)	${1 != 1}	false
< 或 lt	小于(Less than)	${1 lt 2}	true
<=或 le	小于等于(Less than or equal)	${1 <= 1}	true
> 或 gt	大于(Greater than)	${1 > 2}	false
>=或 ge	大于等于(Greater than or equal)	${1 >=1}	true

4.3.3 逻辑表达式

逻辑运算符	说明	范例	结果
&& 或 and	交集(与)	${true and false}	false
|| 或 or	并集(或)	${true || false }	true
! 或 not	非	${not true}	false

4.3.4 三元运算

${逻辑判断?真的值:假的值}

4.3.5 判空表达式

判空,判断的是内容是否为空,不是对象是否为空.

${empty 变量名} 变量一定要放在作用域中
1) 如果变量名在作用域中为空，不存在，返回 true
2) 如果变量名为空串，返回 true
3) 变量名是一个集合，如果集合中没有元素，返回 true

4.3.6 示例代码：使用上面所有的运算符

```
<%@ page import="java.util.ArrayList" %>
<%@ page import="java.util.List" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
```

```
    <title>Title</title>
</head>
<body>
    <h2>算术运算符</h2>
        ${1+1}<br>
        ${2-1}<br>

    <h2>比较运算符</h2>
        ${1 eq 1}<br>
        ${1 != 1}<br>
        ${1 lt 2}<br>

    <h2>逻辑运算符</h2>
        ${true and false}<br>
        ${true || false }<br>
        ${not true}<br>

    <h2>三元运算</h2>
        ${5>6?"男":"女"}

    <h2>判空表达式</h2>
        ${empty aaa} <br>
        <%
            pageContext.setAttribute("bbb","");
            // 集合中没有元素
            List<String> names = new ArrayList<>();
            names.add("唐僧");
            pageContext.setAttribute("ccc", names);
        %>
        ${empty bbb}<br>

        ${empty ccc}<br>
</body>
</html>
```

叮丁狼教育

4.4 EL 从四个作用域中取值

EL 获取数据的方式是从 4 个作用域对象中，从小到大的去获取，如果需要指定作用域获取数据，可使用以下 EL 的内置对象来指定。

4.4.1 pageContext 对象

- JSP 本质是 Servlet，但比 Servlet 多了一个作用域：**页面域**，在 JSP 中有四大作用域。
- 什么是页面域：只在一个JSP页面中起作用，不同的JSP之间不能实现数据的共享，比请求域范围还要小。
- 页面域对象名：pageContext，是JSP其中一个内置对象名

PageContext 操作有关的方法	说明
void setAttribute(String key, Object value)	向页面域中添加键和值
Object getAttribute(String key)	从页面域中得到值
void removeAttribute(String key)	删除四个域中同名的键
Object findAttribute(String key)	自动从四个作用域中去查某个键， 从小到大的范围来查找，如果找到就停止。 如果没有找到，返回null

示例代码：

```
<%
    // 向页面域中添加一个字符串
    pageContext.setAttribute("name", "页面域");
    // 请求域
    request.setAttribute("name", "请求域");
    // 会话域
    session.setAttribute("name", "会话域");
    // 上下文域
    application.setAttribute("name", "上下文域");
%>
自动查找作用域获取：${name}
```

从小到大注释作用域设置数据代码，可看到 EL 表达式在往下获取数据。

叮当狼教育

4.4.2 EL 指定域获取数据

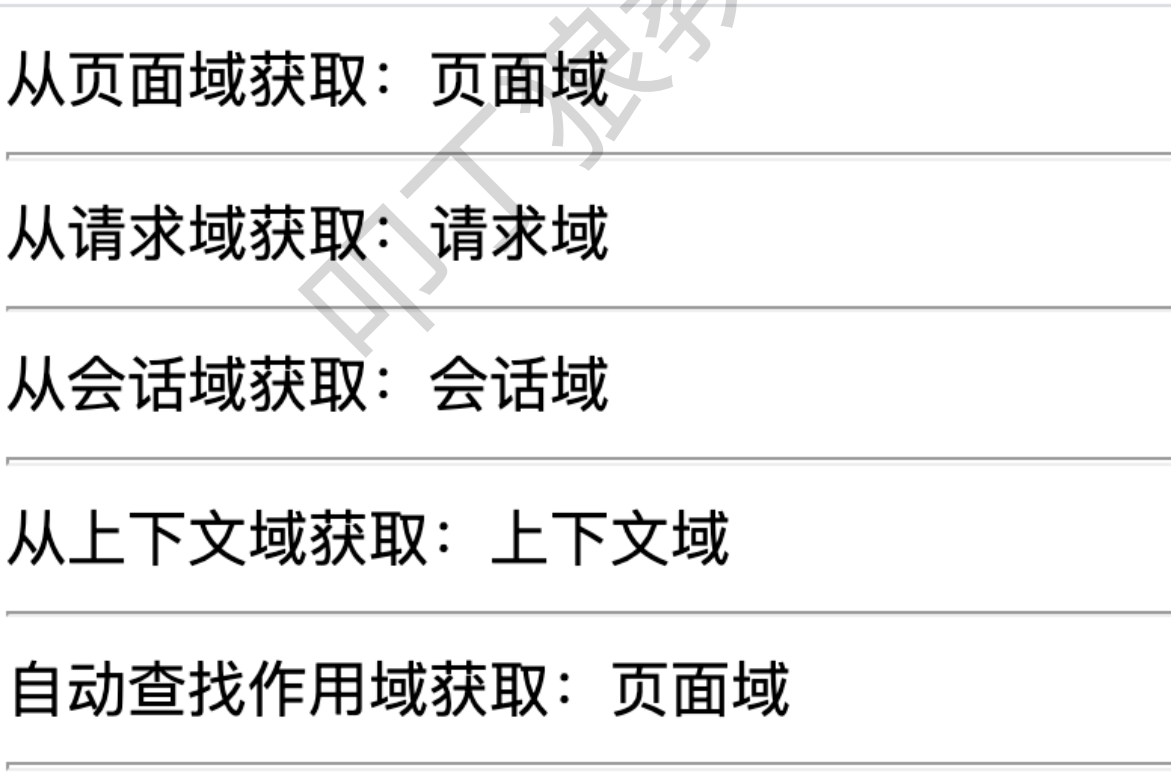
作用域	EL的写法
页面域	${pageScope.键名}
请求域	${requestScope.键名}
会话域	${sessionScope.键名}
上下文域	${applicationScope.键名}

- 示例代码

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>EL指定域获取数据</title>
</head>
<body>
    <%
        // 向页面域中添加一个字符串
```

```
        pageContext.setAttribute("name", "页面域");
        // 请求域
        request.setAttribute("name", "请求域");
        // 会话域
        session.setAttribute("name", "会话域");
        // 上下文域
        application.setAttribute("name", "上下文域");
    %>

    从页面域获取：${pageScope.name}
    <hr>

    从请求域获取：${requestScope.name}
    <hr>

    从会话域获取：${sessionScope.name}
    <hr>

    从上下文域获取：${applicationScope.name}
    <hr>
        自动查找作用域获取：${name}
    <hr>
</body>
</html>
```

- 执行效果



尚硅谷教育

第五章 JSTL 标签库

JSTL 引入代码

```
@WebServlet("/emp")
```

```java
public class EmployeeListServlet extends HttpServlet{
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        // 模拟从数据库中去读取到员工列表,真实项目开发,把以下代码换为从数据库中去读取即可
        List<Employee> list = new ArrayList<Employee>();
        list.add(new Employee("小狼",new BigDecimal("1000")));
        list.add(new Employee("小码",new BigDecimal("1000")));
        list.add(new Employee("小明",new BigDecimal("1000")));

        // 把数据放到作用域中
        req.setAttribute("list",list);
        // 跳转到 jsp 中来显示列表
        req.getRequestDispatcher("/list.jsp").forward(req,resp);
    }
}
```

```jsp
    ${list[0].username} -> ${list[0].salary}<br/>
    ${list[1].username} -> ${list[1].salary}<br/>
    ${list[2].username} -> ${list[2].salary}<br/>
```

目前使用 EL 只能一个一个元素使用索引去获取,又因为数据库的数据条数是不固定,所以需要使用循环来获取.

但是目前只能使用 Java 脚本来做循环,但是在jsp 页面中不要使用Java脚本,所以需要学 新的标签库(JSTL 标签库)

码丁狼教育

5.1 JSTL 概述

5.1.1 为什么使用 JSTL

1. 因为 JSP 页面主要用于显示的，最好不要写 Java 代码 <%%>，如果不用 JSTL 就得写 Java 代码。
2. 对于页面设计人员来说，使用 Java语言操作动态数据是比较困难的，而采用标签和表达式语言相对容易一些， JSTL 的使用为页面设计人员和后台开发人员的分工协作提供了便利。
3. 终极目标：为了简化 JSP 页面的设计。

5.1.2 什么是 JSTL

JSTL 编辑 讨论 上传视频

本词条由"科普中国"科学百科词条编写与应用工作项目 审核 。

JSTL（Java server pages standarded tag library，即JSP标准标签库）是由JCP（Java community Proces）所制定的标准规范，它主要提供给Java Web开发人员一个标准通用的标签库，并由Apache的Jakarta小组来维护。开发人员可以利用这些标签取代JSP页面上的Java代码，从而提高程序的可读性，降低程序的维护难度。

- **JSTL 是 JSP 标准标签库，利用标签库的标签可以取代 JSP 页面上的 Java 代码，为页面设计人员和后台开发人员的分工协作提供了便利。**

5.1.3 JSTL 常用标签库

标签库名	URI字符串	作用	前缀
核心标签库	`http://java.sun.com/jsp/jstl/core`	用于页面逻辑控制 如：if、forEach	c
格式化标签库	`http://java.sun.com/jsp/jstl/fmt`	用于执行格式操作 如：格式化日期字符串	fmt

5.1.4 JSTL 使用步骤

1. 导入 jar 包，每个标签底层其实都是用 Java 代码实现的，复制 jar 包到 **web/WEB-INF/lib** 目录
   也可以在 tomcat 目录下 webapps\examples\WEB-INF\lib 去获取 JSTL 的两个包

2. **导入 jar (File -> project structure -> Modules) (必须)**
3. 创建 JSP 页面，使用 taglib 的指令引用标签库

```
<%--prefix表示前缀，固定为c，uri 标识 --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

4. 在 JSP 页面中就可以使用标签库定义好的标签，比如：`<c:if>`

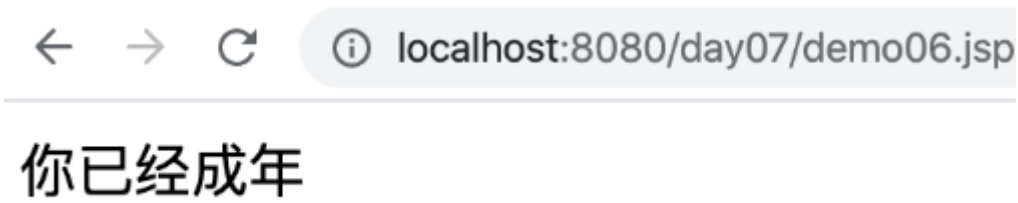
5.2 核心标签库常用标签

5.2.1 `<c:if>` 标签

- 作用： 用于页面上单条件判断。

属性名	是否支持EL	属性类型	属 性 描 述
test	支持，必须是 EL	boolean	EL 中条件为真则执行标签体内容 注意：没有对应 else 标签

- 示例：如果用户提交的 age 值大于18，则显示你已经成年，否则显示未成年。



- 代码

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%--prefix表示前缀，固定为c，uri 标识 --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
```

叮丁狼教育

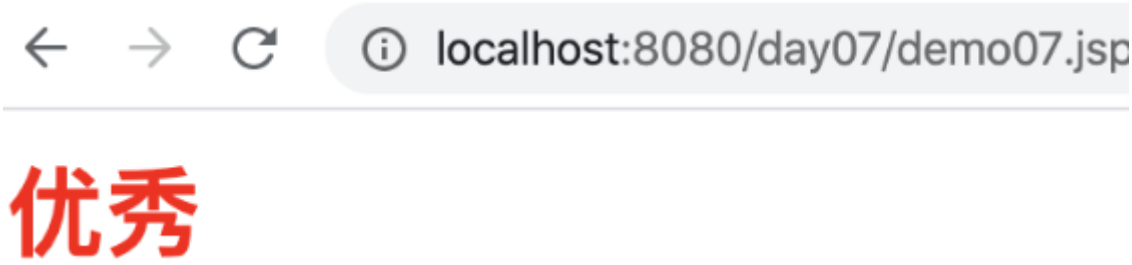
```
    <title>if标签</title>
</head>
<body>
    <%
        // 设置 age 到请求作用域
        request.SetAttribute("age",20)
    %>
    <c:if test="${age >=18}">
        你已经成年
    </c:if>
    <c:if test="${age < 18}">
        未成年，请绕行
    </c:if>
</body>
</html>
```

5.2.2 <c:choose>标签

- 作用：用于多分支判断(多条件判断)

标签名	作用
choose	类似于 java 中swtich, choose只是一个容器，包含下面两个元素
when	可以出现多个 用于每个判断条件，类似于 switch 中 case。有一个 test 属性，与 if 功能相同
otherwise	如果上面所有的条件都不满足，执行 otherwise 内容。类似于 switch 中 default

- 示例：接收用户提交的分数，根据分数输出等级
    - 80~100优秀
    - 60~80及格
    - 0~60不及格
    - 其它输出分数不正确。



- 代码

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>choose标签</title>
</head>
<body>
    <%
```

叮当狼教育

```
        // 设置 age 到请求作用域
        request.SetAttribute("score",80)
    %>
    <c:if test="${not empty score}">
        <c:choose>
            <c:when test="${score >=80 && score<=100}">
                <h2 style="color: red">优秀</h2>
            </c:when>
            <c:when test="${score >=60 && score<80}">
                <h2 style="color: green">良好</h2>
            </c:when>
            <c:when test="${score >=0 && score<60}">
                <h2 style="color: dodgerblue">不及格</h2>
            </c:when>
            <c:otherwise>
                <h2 style="color: orange">分数不正确</h2>
            </c:otherwise>
        </c:choose>
    </c:if>
</body>
</html>
```

5.2.3 `<c:forEach>`标签

- 作用：用于遍历集合或数组

属性名	是否支持EL	属性类型	属性描述
items	true	数组或集合	使用 EL 表达式，代表集合或数组
var	false	String	var 的变量名代表集合中的每一个元素
varStatus	false	String	代表每个元素的状态对象， 一共有4个属性，属性的含义见下表

- varStatus属性

属性	数据类型	含义
index	int	当前遍历到的这个元素索引号，从 0 开始
count	int	遍历到当前为止，一共遍历了多少个元素,从1 开始
first	boolean	如果当前遍历的是第1个元素，则返回true
last	boolean	如果当前遍历的是最后1个元素，则返回true

- 示例：遍历集合

  学生信息：包含编号，姓名，性别，成绩。在 Servlet 中得到所有的学生信息，把信息放到请求域中，转发到JSP 页面，在 JSP 页面上使用 forEach 标签输出所有的学生信息。

叮咚狼教育

学生信息列表

序号	学号	姓名	性别	成绩
1	1000	嫦娥	女	65
2	2000	婉儿	女	85
3	3000	安琪拉	女	90
4	4000	蜘蛛精	女	87
5	5000	孙悟空	男	85

- Student 代码

```
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class Student {
    private int id;
    private String name;
    private boolean gender; // true 男 false 女
    private double score;
}
```

- Servlet 代码

```
@WebServlet("/list")
public class ListStudentServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // 创建集合存储学生信息，假装从数据库查询的学生信息
        List<Student> stus = new ArrayList<Student>();
        stus.add(new Student(1000,"嫦娥",false,65));
        stus.add(new Student(2000,"婉儿",false,85));
        stus.add(new Student(3000,"安琪拉",false,90));
        stus.add(new Student(4000,"蜘蛛精",false,87));
        stus.add(new Student(5000,"孙悟空",true,85)));

        // 将集合存储请求域中
        request.setAttribute("stus", stus);

        // 转发到JSP页面显示数据
        request.getRequestDispatcher("/list.jsp").forward(request, response);
    }
}
```

- JSP代码

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
```

叩丁狼教育

```
    <title>学生信息表</title>
    <style>
        tr {
            text-align: center;
        }
    </style>
</head>
<body>
<table align="center" border="1" cellspacing="0" cellpadding="0" width="80%">
    <caption>学生信息列表</caption>
    <tr>
        <th>序号</th>
        <th>学号</th>
        <th>姓名</th>
        <th>性别</th>
        <th>成绩</th>
    </tr>

    <%--
        forEach标签：用于JSP页面遍历集合和数组
            items属性：设置要遍历的集合或数组：一般从作用域中获取
            var属性：设置一个变量名：用来接收遍历到的每一个元素

            varStatus属性：设置一个变量名：记录当前遍历元素的状态(状态对象)
                index 属性：当前遍历元素的在集合中的索引值：从0开始
                count 属性：遍历到当前元素为止已经遍历了多少个元素,从 1 开始
    --%>
    <c:forEach items="${stus}" var="stu" varStatus="status">
        <!-- 给偶数行设置背景颜色 -->
        <tr style="background-color:${status.count % 2 == 0 ? 'gray;' : ''}">
            <td>${status.count}</td>
            <td>${stu.id}</td>
            <td>${stu.name}</td>
            <td>${stu.gender? "男":"女"}</td>
            <td>${stu.score}</td>
        </tr>
    </c:forEach>
</table>
</body>
</html>
```

5.3 格式化标签库常用标签

5.3.1 `<fmt:formatDate>`标签(了解)

- 作用：用于对日期进行格式化

属性名	属性类型	属 性 描 述
test	Date	要格式化的日期对象
pattern	String	指定日期格式

- 示例代码

叮咚狼教育

```jsp
<%@ page import="java.util.Date" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%-- 引入标签库 --%>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<html>
<head>
    <title>formatDate标签</title>
</head>
<body>
    <%
        request.setAttribute("date", new Date());
    %>
    当前时间:
    <fmt:formatDate value="${date}" pattern="yyyy-MM-dd HH:mm:ss">
</fmt:formatDate>
</body>
</html>
```

小结

1. 理解为什么需要做跳转和数据共享
2. 掌握请求转发和重定向的使用
3. 理解 请求转发和重定向的区别(验证)
4. 掌握 请求转发和重定向的选用问题
5. 理解作用域的作用以及三大作用域的作用范围
6. 掌握三大作用域如何共享数据(API:1 如何获取作用域对象,2 如何通过作用域对象去共享数据)
7. 理解 EL 表达式的作用 : 从四个作用域中从小到大去获取数据
8. 掌握 使用 EL 表达式去获取各种类型的数据
9. 理解 JSTL 标签库的作用
10. 掌握 使用 JSTL 标签库的步骤
11. 掌握 JSTL 标签库的常用标签 (条件判断标签,循环标签)
12. 了解 JSTL 标签库的格式化标签(Date 格式化)

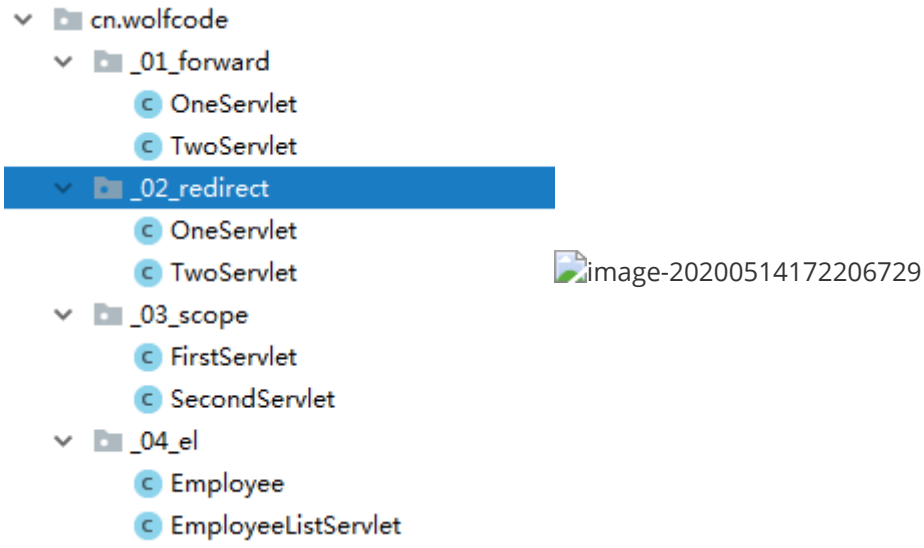


image-20200514172206729

拓展: 使用 Servlet 去数据库中读取数据(Servlet中调用dao 的方法获取数据),把数据展现在jsp中

叮丁狼教育