

# day31-el&jstl&综合案例

## 教学目标

- 1. 能够说出el表达式的作用
- 2. 能够使用el表达式获取javabean的属性
- 3. 能够使用jstl标签库的if标签
- 4. 能够使用jstl标签库的foreach标签
- 5. 能够使用三层架构模式完成显示用户案例

## 案例一,显示所有用户案例

### 一，案例需求

显示所有联系人							
编号	姓名	性别	年龄	籍贯	QQ	邮箱	操作
1	张三	男	11	广州	766335435	766335435@qq.com	<div>修改删除</div>
2	李四	男	12	上海	243424242	243424242@qq.com	<div>修改删除</div>
3	王五	女	13	广州	474574574	474574574@qq.com	<div>修改删除</div>
4	赵六	男	14	北京	987069697	987069697@qq.com	<div>修改删除</div>
5	钱七	女	15	广州	412132145	412132145@qq.com	<div>修改删除</div>
<div>添加联系人</div>							

### 二,技术分析

#### 1,EL表达式

##### 1.1.EL概述

Expression Language:表达式语言,jsp2.0之后内置在jsp里面

目的：为了使JSP写起来更加简单。(代替脚本 <% %>)

##### 1.2.EL语法

`${el表达式}`

##### 1.3.EL表达式的用途

1.获取数据. 获取的是域对象中存储的数据

2.EL执行运算

##### 1.4.EL获取数据

###### 1.4.1获取简单数据类型数据

语法:\${requestScope|sessionScope|applicationScope.属性名};

快捷写法:\${属性名},

属性名就是存在域对象里面的key

```
<body>
    <%
        request.setAttribute("rKey", "rrr");
        session.setAttribute("sKey", "sss");
        /*application就是Servlet里面的ServletContext*/
        application.setAttribute("aKey", "aaa");

    %>

    获得request里面存的数据:<br/>
    老方式:<%=request.getAttribute("rKey")%><br/>
    el方式: ${requestScope.rKey}<br/>
    <hr/>

    获得session里面存的数据:<br/>
    老方式:<%=session.getAttribute("sKey")%><br/>
    el方式: ${sessionScope.sKey}<br/>
    <hr/>

    获得application里面存的数据:<br/>
    老方式:<%=application.getAttribute("aKey")%><br/>
    el方式:${applicationScope.aKey}
    <hr/>

    简单方式的写法:<br/>
    <!--${rKey}: 依次从最小的域往最大的域找,能找到就返回-->
    获得request里面的数据:${rKey}<br/>
    获得session里面的数据:${sKey}<br/>
    获得servletContext里面的数据:${aKey}<br/>

</body>
```

#### 1.4.2 获取数组

语法:\${数组属性名[index]};数组属性名就是存入域对象里面的key

#### 1.4.3 获取list

语法:\${list属性名[index]};list属性名就是存入域对象里面的key

#### 1.4.4 获取Map

语法:\${map属性名.键},map属性名就是存入域对象里面的key

```

<%
String[] array = {"aaa", "bbb", "ccc"};
request.setAttribute("a", array);

List<String> list = new ArrayList<String>();
list.add("aaa");
list.add("bbb");
list.add("ccc");
session.setAttribute("l", list);

Map<String, String> map = new HashMap<String, String>();
map.put("akey", "aaa");
map.put("bkey", "bbb");
map.put("ckey", "ccc");
request.setAttribute("m", map);

%>

获得数组里面的第2个值:<br/>
老方式:<%=((String[])request.getAttribute("a"))[1]%><br/>
el方式: ${a[1]} <br/>
<hr/>
获得list里面的第1个值:<br/>
老方式:<%=((List<String>)session.getAttribute("l")).get(0)%><br/>
el方式: ${l[0]} <br/>
<hr/>
获得map里面的akey对应值:<br/>
老方式:<%=((Map<String, String>)request.getAttribute("m")).get("akey")%><br/>
el方式: ${m.akey} <br/>

```

#### 1.4.5 获取bean

语法:\${bean的属性名(也就是存入的key值).javabean属性}

依赖getxxx()方法; eg: getPassword()---去掉get-->Password()----首字母小写---->password

```

<%
User user = new User("zs", "123456", 18);
request.setAttribute("u", user);

%>

获得密码:<br/>
老方式:<%=((User)request.getAttribute("u")).getPassword()%><br/>
el方式: ${u.password} <br/>
<hr/>

```

#### []和.方式的区别

只要是能用.的都可以使用[]

带下标（数组，list）要用[]

有特殊字符的要用[]

- 获取的是三个域范围的值，存入三个域中
- 能获取到则获取,获取不到返回" "字符串 ,不是返回null
- `${域中属性名}`:依次从requestScope|sessionScope|applicationScope中查找指定的属性  
若找到,立即返回,且结束该次查找  
若找不到返回""
- 若属性名中出现了"."+"+"-"等特殊的符号的时候,快捷获取的方式不好使,必须使用以下方式:  
`${xxxScope["属性名"]}`

特殊情况:

```
<%
    request.setAttribute("a.b.c.d","rrr");

    Map<String,String> map = new HashMap<String,String>();
    map.put("a.akey","aaa");
    map.put("bkey","bbb");
    map.put("ckey","ccc");
    request.setAttribute("m",map);

%>

${a.b.c.d}
${requestScope['a.b.c.d']}<br/>

#{m['a.akey']}
```

<br/>

#### 1.4.EL执行运算

运算符	说明	范例	结果
+	加	$\{17+5\}$	22
-	减	$\{17-5\}$	12
*	乘	$\{17*5\}$	85
/或 div	除	$\{17/5\}$ 或 $\{17 \text{ div } 5\}$	3
%或 mod	取余	$\{17\%5\}$ 或 $\{17 \text{ mod } 5\}$	2
==或 eq	等于	$\{5==5\}$ 或 $\{5 \text{ eq } 5\}$	true
!=或 ne	不等于	$\{5!=5\}$ 或 $\{5 \text{ ne } 5\}$	false
<或 lt	小于	$\{3<5\}$ 或 $\{3 \text{ lt } 5\}$	true
>或 gt	大于	$\{3>5\}$ 或 $\{3 \text{ gt } 5\}$	false
<=或 le	小于等于	$\{3<=5\}$ 或 $\{3 \text{ le } 5\}$	true
>=或 ge	大于等于	$\{3>=5\}$ 或 $\{3 \text{ ge } 5\}$	false
&&或 and	并且	$\{\text{true}\&\text{false}\}$ 或 $\{\text{true and false}\}$	false
!或 not	非	$\{!\text{true}\}$ 或 $\{\text{not true}\}$	false
或 or	或者	$\{\text{true}  \text{false}\}$ 或 $\{\text{true or false}\}$	true
empty	是否为空	$\{\text{empty ""}\}$ ，可以判断字符串、数据、集合的长度是否为 0，为 0 返回 true。empty 还可以与 not 或!一起使用。 $\{\text{not empty ""}\}$	true

#### 1.4.1算数运算

+, -, \*, /

- +不能拼接字符串.

#### 1.4.2逻辑运算

< > = <= != ==

#### 1.4.3关系运算

&& || !

#### 1.4.4非空判断(重点)

empty，判断一个对象是否为null,判断集合长度是否为0，判断一个字符串是否为""

not empty

语法:  $\{\text{empty 属性名}\}$ ; 属性名 就是域对象里面的key值

```

<body>
    <%
        //request.setAttribute("a",10);
        User user = null;
        request.setAttribute("u",user);

        User user1 = new User();
        request.setAttribute("u1",user1);

        List list = new ArrayList();
        request.setAttribute("l",list);

        List list02 = null;
        request.setAttribute("l2",list02);

    %>

    <!--a就是从域里面获得的a
        ${a+19}
        ${a>20}
    --%>
    <!--1.判断对象(user是)否为null--%>
    ${empty u} <!--true--%>
    ${empty u1} <!--false--%>

    ${not empty u}<!--false--%>
    <hr/>
    <!--2.判断一个集合的长度是否为0--%>
    ${empty l} <!--true--%>
    ${empty l2} <!--true--%>

</body>

```

注意的地方:

+只能做加法运算,不能拼接字符串

## 2.JSTL标签库

### 2.1.JSTL概述

JSTL (JSP Standard Tag Library, JSP标准标签库)是一个不断完善的开放源代码的JSP标签库,是由apache的jakarta小组来维护的。JSTL只能运行在支持JSP1.2和Servlet2.3规范的容器上,如tomcat 4.x。在JSP 2.0中也是作为标准支持的

目的: 为了简化在jsp页面上展示数据 遍历数据 判断数据



### 2.2JSTL五大标签库

标签库功能描述	标签库的 URI	建议前缀
核心标签库	http://java.sun.com/jsp/jstl/core	c
XML 标签库	http://java.sun.com/jsp/jstl/xml	x
国际化/格式化标签库	http://java.sun.com/jsp/jstl/fmt	fmt
数据库标签库	http://java.sun.com/jsp/jstl/sql	sql
EL 自定义函数	http://java.sun.com/jsp/jstl/functions	fn

jstl的标签内容有很多，现在目前还常用的标签只有核心标签库里面的 if、choose、foreach 标签，接下来我们一个一个学习。

## 2.3. 核心标签库

### 2.3.1 使用步骤:

- 导入jar包  `jstl.jar`  
 `standard.jar`
- 在页面上导入核心标签库 `<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>`

### 2.3.2 if 标签

表 8.5 `<c:if>` 标签的属性

属性名	是否支持 EL	属性类型	属性描述
test	true	boolean	决定是否处理标签体中的内容的条件表达式
var	false	String	用于指定将 test 属性的执行结果保存到某个 Web 域中的某个属性的名称
scope	false	String	指定将 test 属性的执行结果保存到哪个 Web 域中

- 语法

```
<c:if test="el表达式${..}" [var="给之前的表达式的结果起个名字"] [scope="将结果保存在那个域中 默认page"]>
</c:if>
```

- 实例

```

<%request.setAttribute("a", 3); %>

<c:if test="${a>1}" var="i" scope="page">
    a大于1
</c:if>

<c:if test="${a<=1}">
    a不大于1
</c:if>

${i }

```

### 2.3.2choose标签

- 实例

```

<%
    request.setAttribute("a", "前端");
%>

<c:choose>
    <c:when test="${a == 'java' }">
        java
    </c:when>

    <c:when test="${a == 'Android' }">
        Android
    </c:when>

    <c:when test="${a == 'IOS' }">
        IOS
    </c:when>

    <c:otherwise>
        其它
    </c:otherwise>
</c:choose>

```

### 2.3.3foreach标签



表 8.6 &lt;c:forEach&gt;标签的属性

属性名	是否支持 EL	属性类型	属性描述
var	false	String	指定将当前迭代到的元素保存到 page 这个 Web 域中的属性名称
items	true	任何支持的类型	将要迭代的集合对象
varStatus	false	String	指定将代表当前迭代状态信息的对象保存到 page 这个 Web 域中的属性名称
begin	true	int	如果指定 items 属性, 就从集合中的第 begin 个元素开始进行迭代, begin 的索引值从 0 开始编号; 如果没有指定 items 属性, 就从 begin 指定的值开始迭代, 直到 end 值时结束迭代
end	true	int	参看 begin 属性的描述
step	true	int	指定迭代的步长, 即迭代因子的迭代增量

- 简单的使用:

```

<!--
    var ---- 声明变量
    begin ---- 初始化
    end----- 结束条件
    step---- 步长
-->
<c:forEach var="i" begin="2" end="10" step="2" >
    <hr/>${i }
</c:forEach>

```

- 复杂的使用:

```

~~
//遍历: list set map和数组
List list = new ArrayList();
list.add("aaa");
list.add("bbb");
list.add("ccc");
list.add("ddd");
list.add("eee");
list.add("fff");
pageContext.setAttribute("list", list);
%>
<table border="1">
  <tr>
    <th>数据</th>
    <th>索引</th>
    <th>计数</th>
    <th>第一个</th>
    <th>最后一个</th>
  </tr>

```

- c:forEach中的varStatus属性。

指向一个字符串，该字符串引用一个对象。 map.put("vs",一个对象);

这个对象记录着当前遍历的元素的一些信息:

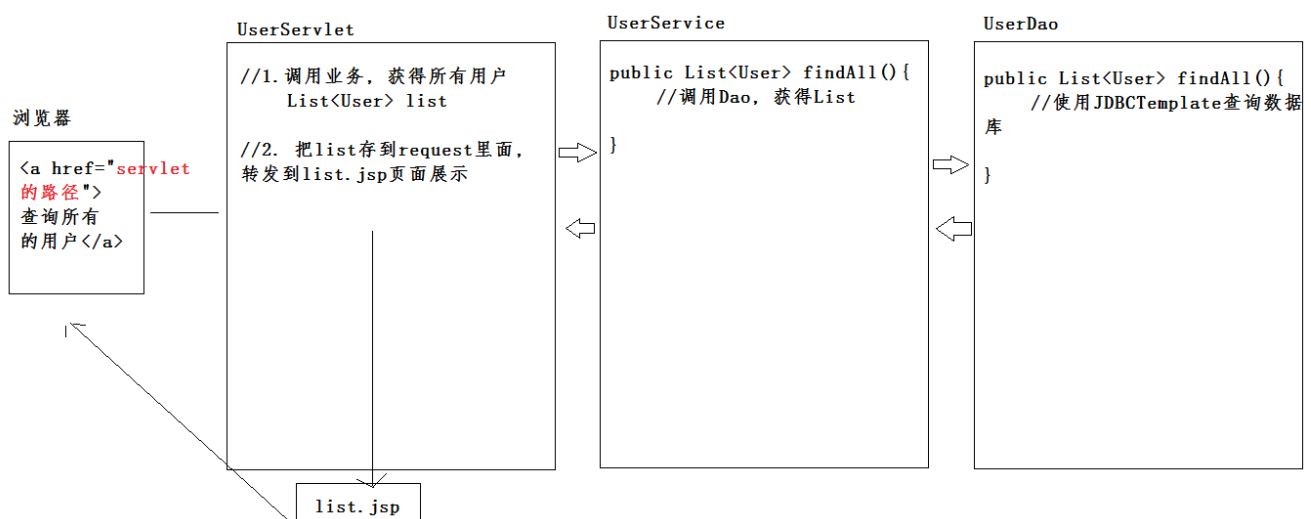
index:返回索引。从0开始

count:返回计数。从1开始

last:是否是最后一个元素

first:是否是第一个元素

### 三, 思路分析



### 四, 代码实现

#### 1.准备工作

- 数据库的创建

```
CREATE TABLE user (  
  id int primary key auto_increment,  
  name varchar(50),  
  sex varchar(50),  
  age int,  
  address varchar(50),  
  qq varchar(50),  
  email varchar(50)  
);  
  
INSERT INTO `user` (`id`, `name`, `sex`, `age`, `address`, `qq`, `email`) VALUES  
(null, '张三', '男', 11, '广东', '766335435', '766335435@qq.com'),  
(null, '李四', '男', 12, '广东', '243424242', '243424242@qq.com'),  
(null, '王五', '女', 13, '广东', '474574574', '474574574@qq.com'),  
(null, '赵六', '女', 18, '广东', '77777777', '77777777@qq.com'),  
(null, '钱七', '女', 15, '湖南', '412132145', '412132145@qq.com'),  
(null, '王八', '男', 25, '广西', '412132775', '412132995@qq.com');
```

- JavaBean的创建

```
public class User {

    private int id;
    private String name;
    private String sex;
    private int age;
    private String address;
    private String qq;
    private String email;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSex() {
        return sex;
    }

    public void setSex(String sex) {
        this.sex = sex;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getQq() {
        return qq;
    }

}
```

```

public void setQq(String qq) {
    this.qq = qq;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

@Override
public String toString() {
    return "Contact{" +
        "id=" + id +
        ", name='" + name + '\'' +
        ", sex='" + sex + '\'' +
        ", age=" + age +
        ", address='" + address + '\'' +
        ", qq='" + qq + '\'' +
        ", email='" + email + '\'' +
        '}';
}
}

```

## 2.代码

- 页面

```
<a href="${pageContext.request.contextPath}/userServlet">查询所有的用户</a>
```

- UserServlet

```

@WebServlet(name = "UserServlet",value = "/userServlet")
public class UserServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        try {
            //1.调用业务
            UserService userService = new UserService();
            List<User> list = userService.findAll();
            //2. 把list存到域对象里面
            request.setAttribute("list",list);
            request.getRequestDispatcher("/list.jsp").forward(request,response);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }
}

```

- UserService

```

public class UserService {
    public List<User> findAll() throws Exception {
        UserDao userDao = new UserDao();
        return userDao.findAll();
    }
}

```

- UserDao

```

public class UserDao {

    public List<User> findAll() throws Exception {
        JdbcTemplate jdbcTemplate = new JdbcTemplate(C3P0Utils.getDataSource());
        return jdbcTemplate.query("select * from user",new BeanPropertyRowMapper<>(User.class));
    }
}

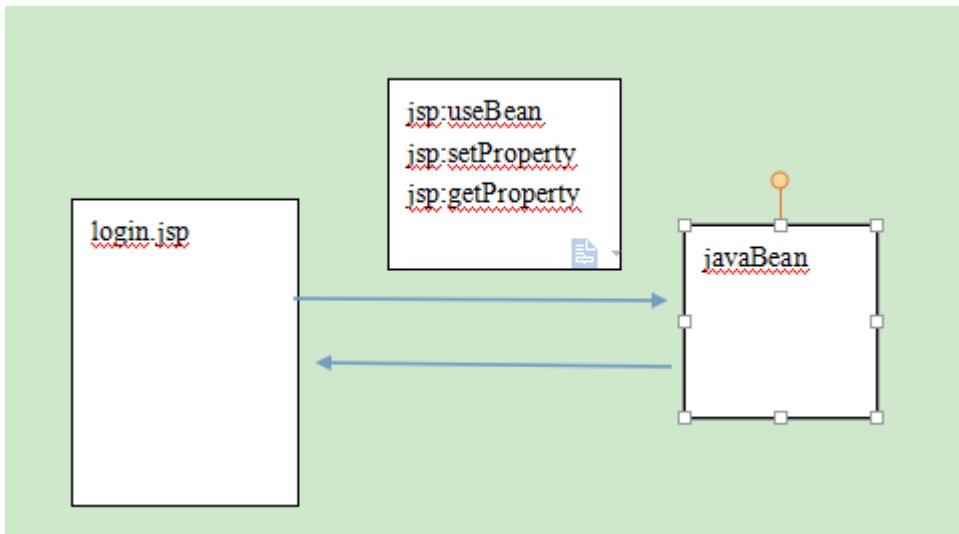
```

## 五,开发模式

### 1.JSP的开发模式一

JSP + JavaBean

javaBean:实体类。特点：私有化的属性、公共的getter setter方法、无参的构造。



## 2.JSP的开发模式二

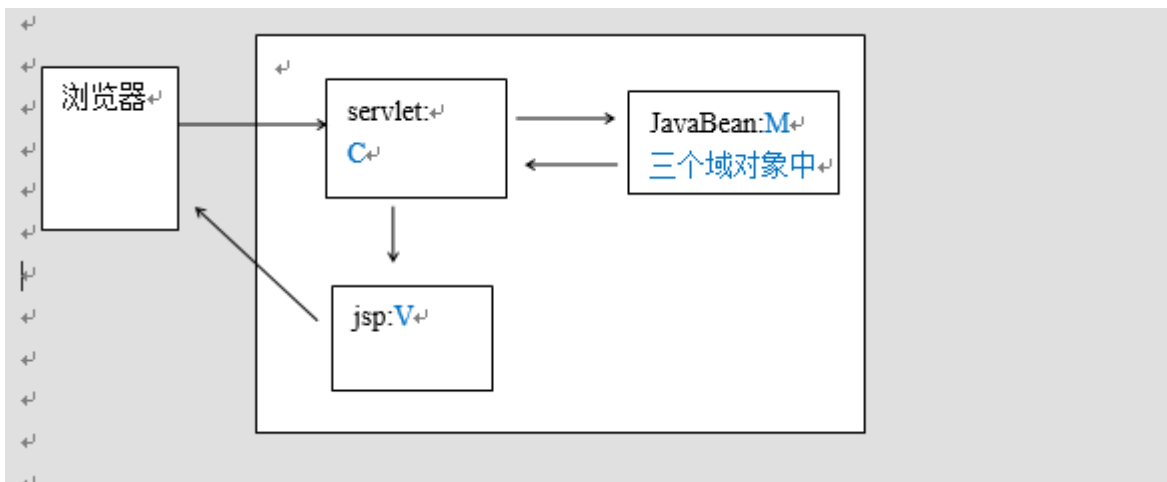
JSP + Servlet + JavaBean 称为MVC的开发模式.

MVC:开发模式、

M: model 模型 (javaBean: 封装数据)

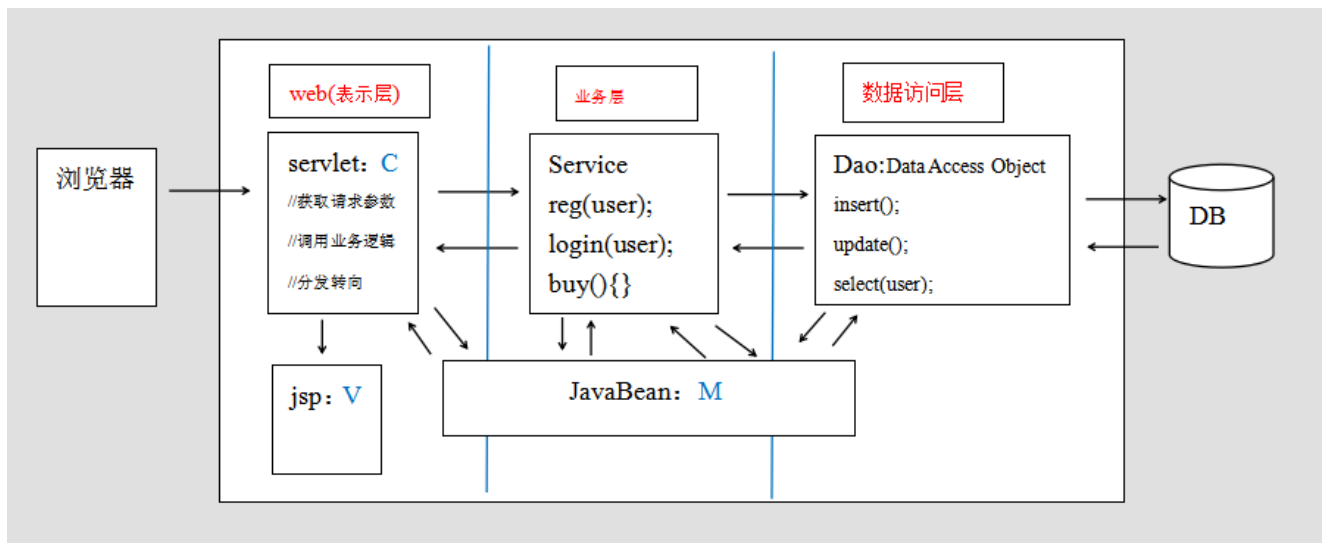
V: View 视图 (JSP: 展示数据)

C: controller 控制器 (Servlet: 处理逻辑代码, 做为控制器)



## 3.三层架构

高内聚, 低耦合



三层架构包名:

简单:

com.itheima.web

com.itheima.service

com.itheima.dao

com.itheima.domain/bean

com.itheima.utils

....

复杂:

com.itheima.项目名.模块名.web

com.itheima.项目名.模块名.service

com.itheima.项目名.模块名.dao