el&jstl&综合案例

学习目标

1. 能够说出el表达式的作用(掌握)
2. 能够使用el表达式获取javabean的属性(掌握)
3. 能够使用jstl标签库的if标签(掌握)
4. 能够使用jstl标签库的foreach标签(重点掌握)
5. 能够使用jstl标签库的choose标签(了解)
6. 能够说出开发模式的作用(了解)
7. 能够使用三层架构模式完成显示用户案例(掌握)

第1章 EL表达式

1.1 EL表达式的基本概述

想要知道什么是EL表达式，它为了解决什么问题而诞生，我们先通过一个场景来了解一下：

现在有一个需求：在jsp使用java代码再request中设置四个数据（10 20 30 40 ）的向页面输出（10+20+（30-40））计算结果，以我们现在的技术去实现会是这样实现：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%-- 在jsp设置(request)四个数据（10 20 30 40 ）的向页面输出（10+20+（30-40））计算结果 --%>
<%
    request.setAttribute("num1", 10);
    request.setAttribute("num2", 20);
    request.setAttribute("num3", 30);
    request.setAttribute("num4", 40);
%>
java代码输出：<%= (Integer)request.getAttribute("num1") +
        (Integer)request.getAttribute("num2") +
        ( (Integer)request.getAttribute("num3") - (Integer)request.getAttribute("num4"))%>
</body>
</html>

```

从上面的代码中我们发现，使用之前的脚本片段来完成实在太过复杂和繁琐，一个简单的算术计算不应该如此艰难的完成。

因此我们需要一个新的技术，来简化java代码的一些操作，这个就是我们需要学习的EL表达式技术。

EL全称：Expression Language

作用：代替jsp中脚本表达式的功能，简化对java代码的操作。

1.2 EL表达式的格式和作用

1. **EL表达式的格式**：${表达式内容}
2. **EL表达式的作用**： 从域对象中查找指定的数据。

1.3 EL表达式的基本使用

1.3.1 EL获得容器（域对象）的数据

再上面的案例中，我们使用原来的方式获取数据十分麻烦，接下来我们使用EL表达式的方式完成上面的需求：

jsp演示代码：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%-- 在jsp设置(request)四个数据（10 20 30 40 ）的向页面输出（10+20+（30-40））计算结果 --%>
<%
    request.setAttribute("num1", 10);
    request.setAttribute("num2", 20);
    request.setAttribute("num3", 30);
    request.setAttribute("num4", 40);
%>
使用EL表达式输出：${num1 + num2 + (num3 - num4)}

</body>
</html>

```

显而易见的使用EL表达式我们获取数字，并执行运算都方便了许多。在上面这个案例中我们是从request中获取容器，那么我们可以获取其他容器中的数据吗？我们来测试一下：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <%--
        在三个容器中存入同名的数据（request、session、servletcontext）
        然后再使用EL表达式获取。
    --%>
    <%
        request.setAttribute("addr","上海");
        request.getSession().setAttribute("addr","北京");
```

```
        request.getSession().getServletContext().setAttribute("addr","广州");

    %>
    ${addr}
</body>
</html>

```

测试的结果出来后，我们会发现，获取出来的是——上海。设置三个容器的数据都是同名数据，这造成了我们获取数据的时候，单单使用名称已经无法区分所以数据，那么怎么解决这个问题呢？如何才能从指定的容器中获取数据呢？

我们需要在获取数据的时候指定相关的容器：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%--
    在三个容器中存入数据（request、session、servletcontext）
    然后再使用EL表达式获取。
--%>
<%
    request.setAttribute("addr","上海");
    request.getSession().setAttribute("addr","北京");
    request.getSession().getServletContext().setAttribute("addr","广州");

%>
指定从request容器获取数据：${requestScope.addr}<br>
指定从session容器获取数据：${sessionScope.addr}<br>
指定从servletcontext容器获取数据：${applicationScope.addr}<br>
</body>
</html>

```

那么我们之前没有指定容器是如何获取数据的呢？其实${addr}在获取容器的时候，默认按request、session、servletcontext顺序从获取数据，只要获取到就不再往下找了，所以上一个案例一直获取到的上海。

1.3.2 EL获取和解析复杂数据

上面的案例我们在获取数据的时候，都是简单的字符串，接下来我们来获取一些复杂数据，复杂数据特指：数组，集合（List Map）和JavaBean。

1.3.2.1 获取数组

servlet代码：

```
package cn.itcast.web;

import cn.itcast.domain.Person;

```

```java
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(name = "ELServlet",urlPatterns = "/el")
public class ELServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

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

        //准备需要解析的数据
        //数组
        int[] arr = {11,22,33,44,55};
        request.setAttribute("arr",arr);

        request.getRequestDispatcher("/4.jsp").forward(request,response);
    }
}
```

jsp代码：

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>

<%--获取数组数据--%>
${arr[4]}<br>

</body>
</html>

```

1.3.2.2 获取数组注意事项

获取数组中某一数据，使用中括号添加角标即可

1.3.2.3 获取集合（list map）

servlet准备数据：

```java
package cn.itcast.web;

import cn.itcast.domain.Person;

import javax.servlet.ServletException;
```

```java
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.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@WebServlet(name = "ELServlet",urlPatterns = "/el")
public class ELServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

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

        //准备需要解析的数据

        //list集合
        List list = new ArrayList();
        list.add("北京传智播客");
        list.add("上海传智播客");
        list.add("广州传智播客");
        request.setAttribute("list",list);
        //map集合
        Map map = new HashMap();
        map.put("language1", "php");
        map.put("language2", "go");
        map.put("language3", "c#");
        map.put("aa.bb.cc", "python");
        request.setAttribute("map",map);

        request.getRequestDispatcher("/4.jsp").forward(request,response);
    }
}

```

jsp代码：

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>

<%--获取list集合数据--%>
获取list集合：${list}<br>
获取list集合某一个数据：${list[0]}<br>
获取map集合：${map}<br>
获取map集合某一数据：${map.language1}<br>
获取map集合特殊key数据：${map["aa.bb.cc"]}<br>

```

```
</body>
</html>

```

1.3.2.4 获取集合注意事项

1. 设置map集合数据的key，尽量不要出现“.“
2. 凡是在EL表达式中使用“.“可以获取的数据，使用"[]"也可以获取

1.3.2.5 获取JavaBean数据

```
package cn.itcast.web;

import cn.itcast.domain.Person;

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.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@WebServlet(name = "ELServlet",urlPatterns = "/el")
public class ELServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

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

        //准备需要解析的数据

        //JavaBean
        Person p = new Person();
        p.setName("李四");
        p.setAge(18);
        request.setAttribute("p",p);
        request.getRequestDispatcher("/4.jsp").forward(request,response);
    }
}

```

jsp:

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>

获取JavaBean数据：${p}<br>
获取JavaBean指定属性的数据：${p.name} ${p.age}<br>
使用中括号，获取JavaBean指定属性的数据：${p["name"]} ${p["age"]}<br>
</body>
</html>

```

1.3.2.6 获取JavaBean数据注意事项

1. JavaBean数据获取类似获取map集合的方式，可以使用“.“获取数据的地方，都可以使用"[]"获取数据。
2. 如果el表达式获取不到数据，页面没有显示内容，不是显示”null“
3. el在获取JavaBean的数据时候，底层调用的是getXXX方法。

1.3.3 EL执行运算

EL不仅可以用来获取数据，之前的案例我们还看到了可以执行运算，因此，接下来我们要学习EL执行运算相关的知识点，它包括了算术运算、逻辑运算、比较运算、empty运算符、三元运算

1.3.3.1 支持算术运算符：+ - * / %

jsp演示：

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%
    request.setAttribute("x",3);
    request.setAttribute("y",4);
    request.setAttribute("z","5");
%>
${x + y}<br>
${x - y}<br>
${x * y}<br>
${x / y}<br>
${x % y}<br>
<%--注意事项:
    1 在EL中，只要是数字就能执行运算，EL在执行计算的时候，默认，将所有数据转换成Long类型
    2 在EL中，如果在一个算式中有数据不存在，那么这个数据不参与运算，不报错继续执行。
--%>
${x+y+z}<br>
${x+y+z+a}<br>
</body>
</html>

```

1.3.3.2 算术运算注意事项

1. 在EL中，只要是数字就能执行运算，EL在执行计算的时候，默认，将所有数据转换成Long类型
2. 在EL中，如果在一个算式中有数据不存在，那么这个数据不参与运算，不报错继续执行。

1.3.3.3 逻辑运算符

下图展示了EL可以支持的逻辑运算，注意：英文和符号效果一致，推荐使用符号

逻辑运算符	说　明	范　　例	结　果
&& 或 and	交集	${ A && B } 或 ${ A and B }	true / false
|| 或 or	并集	${ A || B } 或 ${ A or B }	true / false
! 或 not	非	${ !A } 或 ${ not A }	true / false

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%
    request.setAttribute("flag", true);
    request.setAttribute("info", false);

%>
${flag && info }
${flag || info }
${!info }
</body>
</html>

```

1.3.3.4 逻辑运算符注意事项

注意：逻辑运算中的异或"^"EL不支持。

1.3.3.5 比较运算

下图展示了EL支持的比较运算符，注意：英文和符号效果一致，推荐使用符号

关系运算符	说　　明	范　　例	结　果
= = 或 eq	等于	${ 5 = = 5 } 或 ${ 5 eq 5 }	true
!= 或 ne	不等于	${ 5 != 5 } 或 ${ 5 ne 5 }	false
< 或 lt	小于	${ 3 < 5 }或 ${ 3 lt 5 }	true
> 或 gt	大于	${ 3 > 5 }或 ${ 3 gt 5 }	false
<= 或 le	小于等于	${ 3 <= 5 }或 ${ 3 le 5 }	true
>= 或 ge	大于等于	${ 3 >= 5 }或 ${ 3 ge 5 }	false

jsp代码演示：

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%
    request.setAttribute("x",3);
    request.setAttribute("y",4);
%>
${x < y }
${x <= y }
${x > y }
${x >= y }
${x == y }
${x != y }
</body>
</html>

```

1.3.3.6 比较运算注意事项

注意：使用比较运算符要保证数据是存在的并且是可比较的。例如：不要出现设置数据为request.setAttribute("y",null);然后再进行比较。

1.3.3.7 empty运算符和三元运算符

empty运算符是用来判断当获取的数据是否为空或者当前获取的集合是否没有任何数据，三元运算符和java的三元运算符功能一致。

jsp：

```jsp
<%--再当前jsp导入ArrayList--%>
<%@ page import="java.util.ArrayList" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%
    request.setAttribute("str",null);
    request.setAttribute("list",new ArrayList<>());
%>
${empty str}<br>
${empty list}<br>
${str == null? "数据为空":str}<br>
</body>
</html>
```

1.3.3.8 empty运算符注意事项

注意：以上的empty运算符案例中，empty运算符可以和比较运算符组合使用。

例如：${not empty str}表示str不为空，返回true。

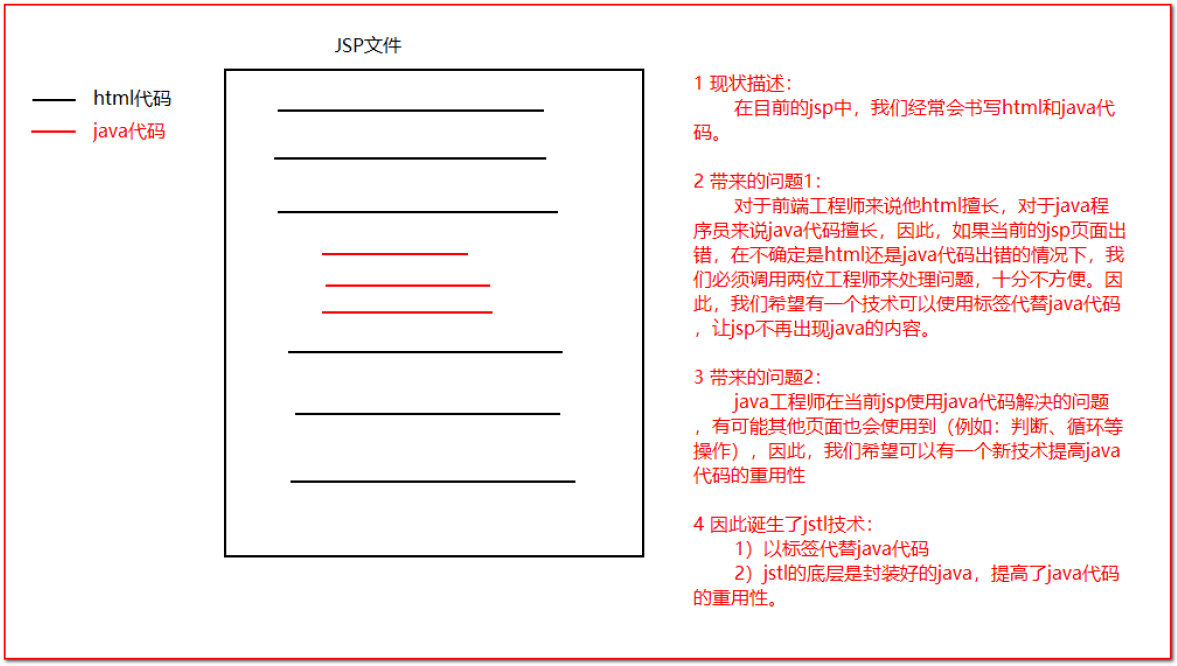
1.4 使用EL表达式获取存放在Cookie中的数据

第2章 JSTL的核心标签库使用

2.1 jstl标签的基本概述

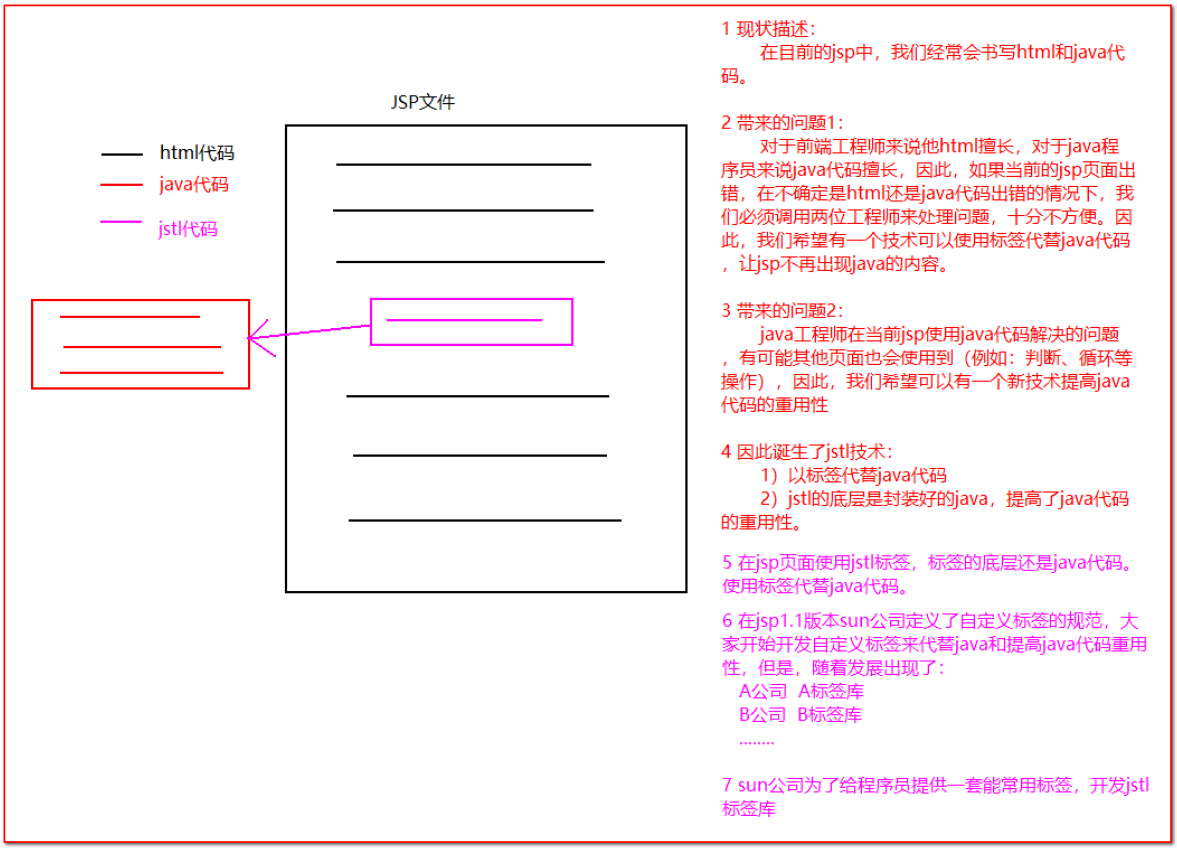
2.1.1 jstl的由来

接下来我们要学习一个新的知识点，那么首先，我们要知道他的由来：



2.1.2 jstl的概述

从JSP 1.1规范开始JSP就支持使用自定义标签，使用自定义标签大大降低了JSP页面的复杂度，同时增强了代码的重用性，因此自定义标签在WEB应用中被广泛使用。许多WEB应用厂商都开发出了自己的一套标签库提供给用户使用，这导致出现了许多功能相同的标签，令网页制作者无所适从，不知道选择哪一家的好。为了解决这个问题，Apache Jakarta小组归纳汇总了那些网页设计人员经常遇到的问题，开发了一套用于解决这些常用问题的自定义标签库，这套标签库被SUN公司定义为标准标签库（The JavaServer Pages Standard Tag Library），简称JSTL。使用JSTL可以解决用户选用不同WEB厂商的自定义标签时的困惑，JSP规范同时也允许WEB容器厂商按JSTL标签库的标准提供自己的实现，以获取最佳性能。



JSTL标签库提供5大功能（了解）：

1. core：jstl的核心标签库。（目前还在使用）
2. fmt：格式化（国际化）的标签（使用较少，对页面显示数据，格式化，现在都交给前端去做）
3. functions：jstl中提供对字符串操作的函数库(不再使用，建议在数据显示在页面之前，在后台程序中，先格式化好字符串，然后直接显示，不再页面做处理，如果有前端，交给前端处理（javascript 解析json格式数据））
4. sql：jstl提供的在jsp页面上书写sql，操作数据库，目前已经不再（不允许）使用（jsp开发模式二（MVC），这个开发模式不允许，在页面操作数据库）
5. xml：jstl操作xml文件的。目前已经不再使用（页面传递数据，页面解析数据，使用json格式字符串代替xml。）

□因此我们需要了解的只由jstl的核心标签库。

2.1.2 jstl核心标签库列表

标签名称	作用
<c:out>	通常用于输出一段文本内容到客户端浏览器
<c:set>	用于设置各种Web域中的属性
<c:remove>	用于删除各种Web域中的属性
<c:catch>	用于捕获嵌套在标签体中的内容抛出的异常
<c:if>	用户java代码if(){}语句功能(掌握)
<c:choose>	用于指定多个条件选择的组合边界，它必须与c:when和c:otherwise标签一起使用
<c:forEach>	用户代替java代码for循环语句(掌握)
<c:forTokens>	用户迭代操作String字符
<c:param>	给请求路径添加参数
<c:url>	重写url，在请求路径添加sessionid
<c:import>	用于在JSP页面中导入一个URL地址指向的资源内容
<c:redirect>	用于将当前的访问请求转发或重定向到其他资源

2.2 jstl标签的安装

我们知道了jstl可以帮助我们解决jsp页面出现java和提高java在页面的重用性问题，那么接下来，我们需要的是学习使用jstl。

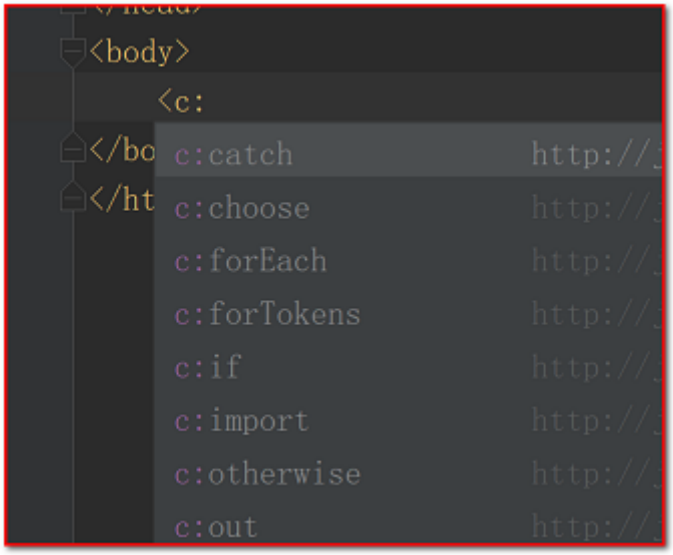
2.2.1 导入jar包

```
javax.servlet.jsp.jstl.jar
jstl-impl.jar
```

2.2.2 使用taglib指令在jsp页面导入要使用的jstl标签库

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

在jsp书写"<c:"，看到如下提示，说明安装成功：



2.3 常用的jstl标签

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

2.3.1 if标签

2.3.1.1 if标签作用

起到java代码的判断的作用

2.3.1.2 if标签属性介绍

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

test：判断是否执行标签内的内容（true——执行标签中的内容，false，不执行）。

var：用来保存test属性的结果（使用var属性给他取个名字），这个结果可以保存到指定的容器中。

scope：指定保存数据的容器。

注：是否支持EL表达式——是否可以书写EL表达式在属性中。

2.3.1.3 if标签注意事项

if标签，相当于java中的if(){}语句，而不是if(){}else{}语句

按照属性的数据类型传入数据，否则报错

2.3.1.4 if标签演示

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

    <c:if test="${true}" scope="session" var="flag">
        测试if标签，设置test为true
    </c:if>
    <br>测试if标签,获取test属性的值，从知道的session容器中获取:
${sessionScope.flag}
</body>
</html>

```

2.3.2 choose标签

2.3.2.1 choose标签作用

/c:choose标签用于指定多个条件选择的组合边界，它必须与<c:when>和<c:otherwise>标签一起使用。三个标签组合发挥java代码if(){}else if(){} else{}语句的作用。

2.3.2.2 choose标签子标签介绍

<c:when>，相当于else if(){}。<c:when>标签含有test属性，作用与if相同

<c:otherwise>，相当于else{}。

2.3.2.2 choose标签注意事项

三个标签必须组合使用，一组标签中不能出现两个<c:otherwise>。

2.3.2.3 choose标签演示

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%
    request.setAttribute("num", 4);
    request.setAttribute("flag", 1);

%>
<!-- c:choose 表示那些when 和 otherwise是一组的 -->
<c:choose>
    <c:when test="${num == 1 }">星期一</c:when>
    <c:when test="${num == 2 }">星期二</c:when>
    <c:when test="${num == 3 }">星期三</c:when>
    <c:when test="${num == 4 }">星期四</c:when>
    <c:when test="${num == 5 }">星期五</c:when>
    <c:when test="${num == 6 }">星期六</c:when>
    <c:when test="${num == 7 }">星期日</c:when>
```

```
    <c:otherwise>参数不合法</c:otherwise>
</c:choose>

<c:choose>
    <c:when test="${flag == 1 }">白天</c:when>
    <c:when test="${flag == 2 }">黑夜</c:when>
    <c:otherwise>参数不合法</c:otherwise>
</c:choose>
</body>
</html>

```

2.3.3 foreach标签(重要)

2.3.3.1 foreach标签作用

起到java代码的for循环作用

2.3.3.2 foreach标签属性介绍

表 8.6 <c:forEach>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

var：在不循环对象的时候，保存的是控制循环的变量；在循环对象的时候，保存的是被循环对象中的元素

items：指定要循环的对象

varStatus：保存了当前循环过程中的信息（循环的开始、结束、步长、次数等）

begin：设置循环的开始

end：设置循环的结束

step：设置步长——间隔几次循环，执行一次循环体中的内容

2.3.3.3 foreach标签演示

演示foreach循环标签的时候我们分开两种情况：不循环对象和循环对象。

我们先来看不循环对象的时候：

2.3.3.3.1 foreach不循环对象

begin、end、step 三属性的演示：设置循环开始、结束和步长。

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%-- 演示foreach标签 --%>
<%
    for (int i = 1; i <= 5; i++) {

    }
%>
<%--
begin="1" int i = 1;
end="5"   i<=5;
step="6" 步长，控制循环，间隔几次循环，执行一次循环体中的内容

step="1" 1 2 3 4 5  M M M M M
step="2" 12 34 5    M M M
step="3" 123 45     M M
step="4" 1234 5     M M
step="5" 12345      M

--%>
<c:forEach begin="1" end="5" step="1" >
    M
</c:forEach>
</body>
</html>
```

var属性演示：在不循环对象的时候，相当于将for循环中的循环变量i，每次都记录下来

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%-- 演示foreach标签 --%>
<%
    for (int i = 1; i <= 5; i++) {

    }
%>
<%--
var 属性在不循环对象的时候，相当于将for循环中的循环变量i，每次都记录下来。
--%>
<c:forEach begin="11" end="15" step="1" var="info">
    ${info}
```

```
</c:forEach>
</body>
</html>
```

varStatus属性演示：保存了当前循环过程中的信息

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%-- 演示foreach标签 --%>
<%--
varStatus :保存了当前循环过程中的信息，信息包括以下内容：
  1     public java.lang.Integer getBegin()
         返回为标签设置的begin属性的值，如果没有设置begin属性则返回null
  2     public int getCount()
         返回当前已循环迭代的次数
  3     public java.lang.Object getCurrent()
         返回当前迭代到的元素对象
  4     public java.lang.Integer getEnd()
          返回为标签设置的end属性的值，如果没有设置end属性则返回null
  5     public int getIndex()
         返回当前迭代的索引号
  6     public java.lang.Integer getStep()
         返回为标签设置的step属性的值，如果没有设置step属性则返回null
  7     public boolean isFirst()
         返回当前是否是第一次迭代操作
  8     public boolean isLast()
         返回当前是否是最后一次迭代操作
--%>
<c:forEach begin="11" end="15" step="1" var="info" varStatus="sta">
    <td>${sta.index}</td>
    <td>${sta.count}</td>
    <td>${sta.first}</td>
    <td>${sta.last}</td><br>
</c:forEach>
</body>
</html>

```

接下来我们在来看foreach如何循环对象

2.3.3.3.2 foreach循环对象（数组、list、map）

```
<%--import="java.util.*"导入java.util下的内容，给当前jsp使用--%>
<%@ page contentType="text/html;charset=UTF-8" import="java.util.*"
language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%--演示循环数组--%>
```

```jsp
<%
    int[] arr = {666,888,999,1024};
    request.setAttribute("arr", arr);

%>
<%-- var在循环对象的时候，临时保存被循环到元素 --%>
<c:forEach items="${arr }" var="num">
${num }
</c:forEach>
<hr>
<%
    List list = new ArrayList();
    list.add("卡奴");
    list.add("兰恩");
    list.add("云娜");

    request.setAttribute("list", list);

%>
<c:forEach items="${list }" var="wind">
    ${wind }
</c:forEach>
<hr>
<%
    Map map = new HashMap();
    map.put("ms1", "简历");
    map.put("ms2", "身份证");
    map.put("ms3", "学历证明");
    map.put("ms4", "体检报告");

    request.setAttribute("map", map);

%>
<c:forEach items="${map }" var="entry">

    ${entry.key }
    ${entry.value }
</c:forEach>
</body>
</html>

```

EL表达式和JSTL的总结

1. 使用EL表达式获取域对象里面的数据:${key}
2. 使用EL表达式执行运算，empty运算符可以判断对象是否为null，字符串是否为空字符串，集合的长度是否为0
3. 使用EL表达式获取cookie的值,${cookie.cookie的name.value}
4. jstl的使用步骤:拷贝jar，使用taglib指令引入jstl到需要的页面
5. jstl的if标签，用于替代if语句，它需要test属性用于写判断表达式(需要结合EL表达式一起使用)
6. jstl的choose、when、otherwise标签，这就相当于else if
7. jstl最重要的标签forEach标签

1. begin:开始遍历的下标
2. end:结束遍历的下标
3. var:遍历出来的值，往域对象存放时候的key
4. varStatus:遍历出来的元素的状态往域对象存放时候的key
    1. index:下标
    2. count:计数
    3. first
    4. last
    5. current，当前的数据
5. items:要进行遍历的数据集
6. step:步长

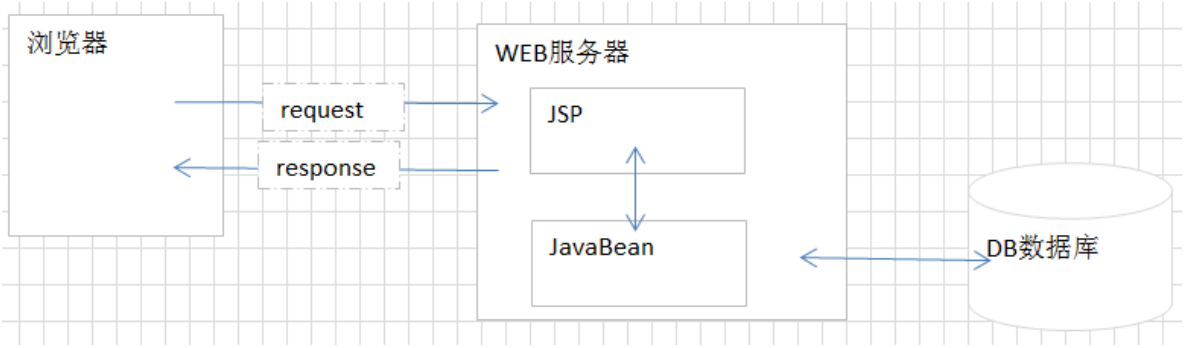
第3章 三层架构和MVC模式

3.1 JSP开发模式

当SUN公司推出JSP后，同时也提供相应的开发模式，JavaWeb经历了JSP Model1 第一代，JSPModel2 第二代，JSP Model 3 三个时期。

3.1.1.JSP Model1 第一代

JSP Model1是JavaWeb早期的模型，它适合小型Web项目，开发成本低！Model1第一代时期，服务器端只有JSP页面，所有的操作都在JSP页面中，连访问数据库的API也在JSP页面中完成。也就是说，所有的东西都在一起，对后期的维护和扩展极为不利。



JSP的作用:负责与客户端的交互

JavaBean的概念:用Java代码实现的可重用组件。

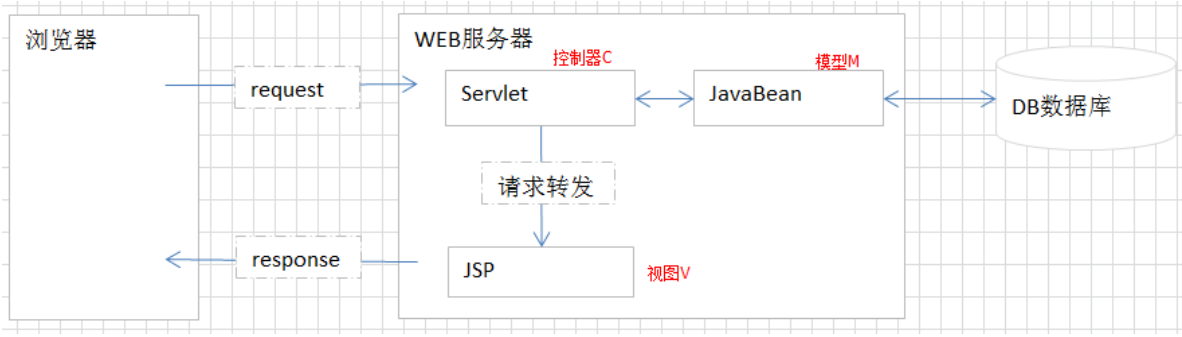
JavaBean和实体类是有区别的

3.1.2 JSP Model2 第二代

JSP Model1第二代有所改进，把业务逻辑的内容放到了JavaBean中，而JSP页面负责显示以及请求调度的工作。虽然第二代比第一代好了些，但还让JSP做了过多的工作，JSP中把视图工作和请求调度（控制器）的工作耦合在一起了。

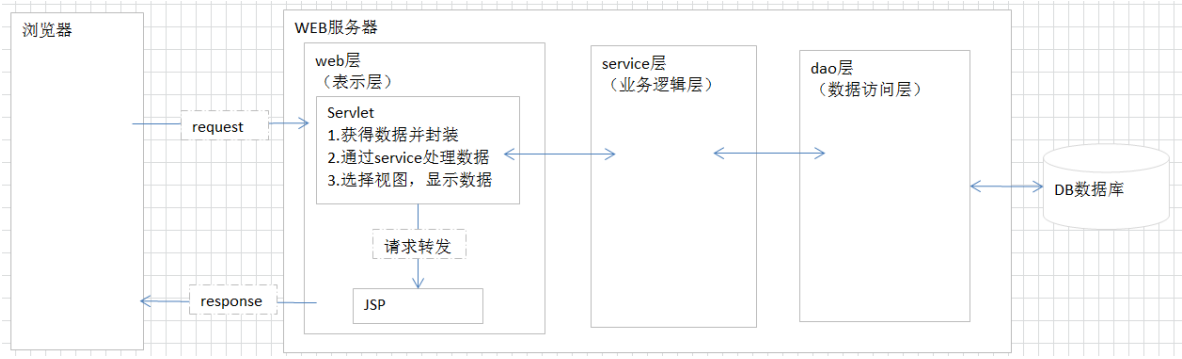
3.1.3 JSP Model3 第三代 (MVC)

JSP Model 3 Model3使用到的技术有：Servlet、JSP、JavaBean。Model2 是MVC设计模式在Java语言的具体体现。 □ JSP：视图层，用来与用户打交道。显示数据给用户； □ Servlet：控制层，负责接收用户的请求,找到合适的模型对象来处理业务逻辑，转发到合适的视图； □ JavaBean：模型层，完成具体的业务工作，例如：转账、登录等。



3.1.3 三层架构

JSP模式是理论基础，但实际开发中，我们常将服务器端程序，根据逻辑进行分层。一般比较常见的是分三层，我们称为：经典三层体系架构。三层分别是：表示层(web层)、业务逻辑层(service)、数据访问层(dao层)。 □ 表示层：又称为 web层，与浏览器进行数据交互的。Servlet □ 业务逻辑层：又称为service层，专门用于处理业务数据的。 □ 数据访问层/持久层：又称为dao层，与数据库进行数据交换的。将数据库的一条记录与JavaBean进行对应。



JavaBean:使用Java语言编写的可重用组件

实体类:封装数据，有对应的set和get方法的这么一些类

耦合性:

1. 常量的硬编码，通过配置文件解决
2. 编译期的耦合。(JDBC) spring框架

第4章 使用三层架构和MCV模式完成用户显示列表案例

4.1 案例需求

使用三层架构和MVC模式开发代码，完成用户显示列表功能。

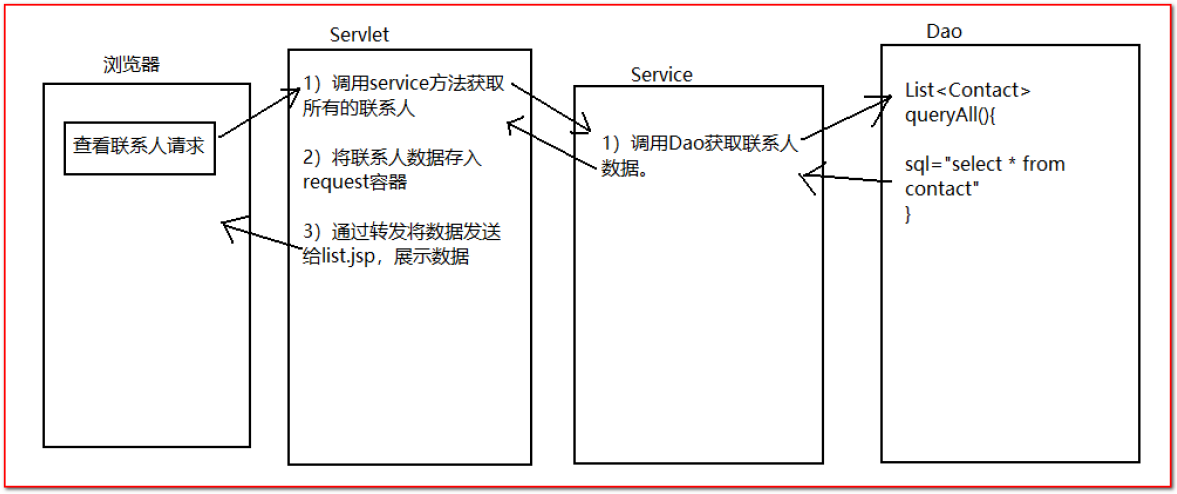
4.2 案例效果

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	修改 删除
2	李四	男	12	上海	243424242	243424242@qq.com	修改 删除
3	王五	女	13	广州	474574574	474574574@qq.com	修改 删除
4	赵六	男	14	北京	987069697	987069697@qq.com	修改 删除
5	钱七	女	15	广州	412132145	412132145@qq.com	修改 删除

添加联系人

4.3 案例分析



4.4 实现步骤

4.4.1 导入页面

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE html>
<!-- 网页使用的语言 -->
<html lang="zh-CN">
<head>
    <!-- 指定字符集 -->
    <meta charset="utf-8">
    <!-- 使用Edge最新的浏览器的渲染方式 -->
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!-- viewport视口：网页可以根据设置的宽度自动进行适配，在浏览器的内部虚拟一个容器，容器的宽度与设备的宽度相同。
    width: 默认宽度与设备的宽度相同
    initial-scale: 初始的缩放比，为1:1 -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -->
    <title>Bootstrap模板</title>

    <!-- 1. 导入CSS的全局样式 -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <!-- 2. jQuery导入，建议使用1.9以上的版本 -->
    <script src="js/jquery-2.1.0.min.js"></script>
    <!-- 3. 导入bootstrap的js文件 -->
    <script src="js/bootstrap.min.js"></script>
    <style type="text/css">
```

```
        td, th {
            text-align: center;
        }
    </style>
</head>
<body>
<div class="container">
    <h3 style="text-align: center">显示所有联系人</h3>
    <table border="1" class="table table-bordered table-hover">
        <tr class="success">
            <th>编号</th>
            <th>姓名</th>
            <th>性别</th>
            <th>年龄</th>
            <th>籍贯</th>
            <th>QQ</th>
            <th>邮箱</th>
            <th>操作</th>
        </tr>
        <tr>
            <td>1</td>
            <td>张三</td>
            <td>男</td>
            <td>20</td>
            <td>广东</td>
            <td>44444222</td>
            <td>zs@qq.com</td>
            <td><a class="btn btn-default btn-sm" href="修改联系人.html">修改</a> <a class="btn btn-default btn-sm" href="修改联系人.html">删除</a></td>
        </tr>
        <tr>
            <td>2</td>
            <td>张三</td>
            <td>男</td>
            <td>20</td>
            <td>广东</td>
            <td>44444222</td>
            <td>zs@qq.com</td>
            <td><a class="btn btn-default btn-sm" href="修改联系人.html">修改</a> <a class="btn btn-default btn-sm" href="修改联系人.html">删除</a></td>
        </tr>
        <tr>
            <td>3</td>
            <td>张三</td>
            <td>男</td>
            <td>20</td>
            <td>广东</td>
            <td>44444222</td>
            <td>zs@qq.com</td>
            <td><a class="btn btn-default btn-sm" href="修改联系人.html">修改</a> <a class="btn btn-default btn-sm" href="修改联系人.html">删除</a></td>
        </tr>
        <tr>
            <td>4</td>
            <td>张三</td>
```

```
            <td>男</td>
            <td>20</td>
            <td>广东</td>
            <td>44444222</td>
            <td>zs@qq.com</td>
            <td><a class="btn btn-default btn-sm" href="修改联系人.html">修改</a> <a class="btn btn-default btn-sm" href="修改联系人.html">删除</a></td>
        </tr>
        <tr>
            <td>5</td>
            <td>张三</td>
            <td>男</td>
            <td>20</td>
            <td>广东</td>
            <td>44444222</td>
            <td>zs@qq.com</td>
            <td><a class="btn btn-default btn-sm" href="修改联系人.html">修改</a> <a class="btn btn-default btn-sm" href="修改联系人.html">删除</a></td>
        </tr>
        <tr>
            <td colspan="8" align="center"><a class="btn btn-primary" href="添加联系人.html">添加联系人</a></td>
        </tr>
    </table>
</div>
</body>
</html>

```

4.4.2 导入页面相关的资源文件

复制今天资料文件夹/案例原型下的三个文件夹到web根路径：

```
css
fonts
js
```

4.4.3 导入jar包、配置文件、实体类和工具类

1. 导入jar包

```
commons-beanutils-1.8.3.jar
commons-logging-1.1.1.jar
druid-1.0.9.jar
javax.servlet.jsp.jstl.jar
jstl-impl.jar
mysql-connector-java-5.1.18-bin.jar
spring-beans-4.2.4.RELEASE.jar
spring-core-4.2.4.RELEASE.jar
spring-jdbc-4.2.4.RELEASE.jar
spring-tx-4.2.4.RELEASE.jar
```

2. 配置文件:druid-config.properties

```
driverClass:com.mysql.jdbc.Driver
jdbcUrl:jdbc:mysql:///day05
username:root
password:root
```

3. 实体类

```
package cn.itcast.domain;

public class Contact {

    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 + '\'' +
                '}';
    }
}
```

4. 工具类

```
package cn.itcast.utils;

import com.alibaba.druid.pool.DruidDataSource;

import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ResourceBundle;

public class JDBCUtils {

    private static DruidDataSource dc =  new DruidDataSource();

    static {
        ResourceBundle bundle = ResourceBundle.getBundle("druid-
config");
        String driverClass = bundle.getString("driverClass");
```

```
        String jdbcUrl = bundle.getString("jdbcUrl");
        String username = bundle.getString("username");
        String password = bundle.getString("password");

        dc.setDriverClassName(driverClass);
        dc.setUrl(jdbcUrl);
        dc.setUsername(username);
        dc.setPassword(password);
    }

    public static Connection getConnection() throws SQLException {
        return dc.getConnection();
    }

    public static DataSource getDataSource(){
        return dc;
    }
}
```

4.4.4 编写servlet代码

```
package cn.itcast.web;

import cn.itcast.domain.Contact;
import cn.itcast.service.ContactService;
import cn.itcast.service.impl.ContactServiceImpl;

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.util.List;

@WebServlet(name = "QueryAllServlet",urlPatterns = "/queryAll")
public class QueryAllServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //获取联系人数据
        ContactService contactService = new ContactServiceImpl();
        List<Contact> list = contactService.queryAll();
        //将联系人数据转发到页面展示
        request.setAttribute("list" ,list);
        request.getRequestDispatcher("/list.jsp").forward(request,response);
    }
}

```

4.4.5 编写service代码

接口：

```
package cn.itcast.service;

import cn.itcast.domain.Contact;

import java.util.List;

public interface ContactService {
    /**
     * 查询所有联系人的方法
     *
     * */
    List<Contact> queryAll();
}

```

实现类：

```
package cn.itcast.service.impl;

import cn.itcast.dao.ContactDao;
import cn.itcast.dao.impl.ContactDaoImpl;
import cn.itcast.domain.Contact;
import cn.itcast.service.ContactService;

import java.util.List;

public class ContactServiceImpl implements ContactService {

    private ContactDao contactDao = new ContactDaoImpl();

    @Override
    public List<Contact> queryAll() {
        return contactDao.queryAll();
    }
}

```

4.4.6 编写dao代码

接口：

```
package cn.itcast.dao;

import cn.itcast.domain.Contact;

import java.util.List;

public interface ContactDao {
```

```
    /**
     *  查询所有联系人的方法
     *
     * */
    List<Contact> queryAll();
}

```

实现类:

```
package cn.itcast.dao.impl;

import cn.itcast.dao.ContactDao;
import cn.itcast.domain.Contact;
import cn.itcast.utils.JDBCUtils;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;

public class ContactDaoImpl implements ContactDao {

    private JdbcTemplate template = new
JdbcTemplate(JDBCUtils.getDataSource());
    @Override
    public List<Contact> queryAll() {
        String sql = "select * from contact";
        List<Contact> query = template.query(sql, new RowMapper<Contact>() {
            @Override
            public Contact mapRow(ResultSet rs, int i) throws SQLException {

                Contact contact = new Contact();

                contact.setId(rs.getInt("id"));
                contact.setName(rs.getString("name"));
                contact.setSex(rs.getString("sex"));
                contact.setAge(rs.getInt("age"));
                contact.setAddress(rs.getString("address"));
                contact.setQq(rs.getString("qq"));
                contact.setEmail(rs.getString("email"));
                System.out.println(contact);

                return contact;
            }
        });
        return query;
    }
}

```

单例模式

多例模式:每次都会新创建对象

单例模式:在整个项目中只有一个对象

作用:可以减少系统的开销

实现单例模式的方法:

1. 懒汉式
2. 饿汉式
3. 双重检测锁机制
4. 静态内部类的方式