第5章 JSP与JavaBean

JavaBean是一款可重复使用的软件组件，实际上，JavaBean是用Java语言编写的一个特殊的Java类，该类的一个实例称为一个JavaBean，简称Bean。JavaBean一般用于实现网页中的业务逻辑或数据库操作。JavaBean的作用如图5-1所示。

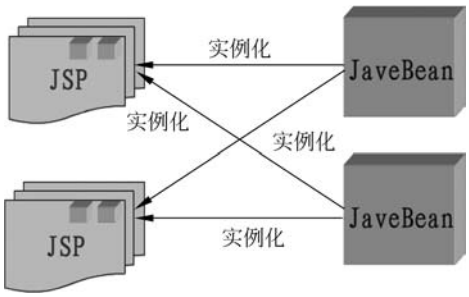

图5-1　JavaBean的作用

相较于其他Java类，JavaBean具有以下特点。

（1）使用JavaBean可以实现代码的重复运用。

（2）JavaBean易编写、易维护、易使用。

（3）由于JavaBean是基于Java语言的，所以JavaBean不依赖平台，可以在任何安装了Java运行环境的平台上使用，具有可跨平台的特点。

首先，需要在JSP页面加载一个Bean，也就是实例化一个JavaBean对象，然后JSP页面将数据的处理过程指派给一个或几个Bean来完成，也就是JSP页面调用Bean完成数据的处理，并将有关处理结果存放在Bean中，由JSP页面负责显示Bean中的数据。JSP与JavaBean的相关联系如图5-2所示。

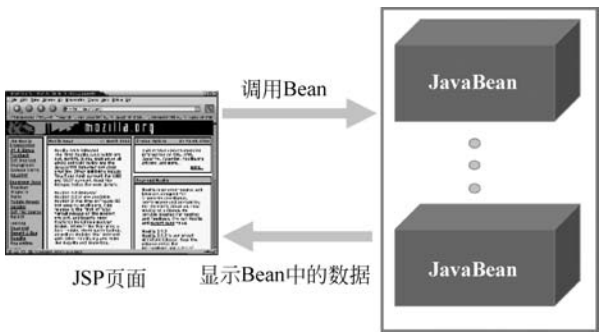

图5-2　JSP与JavaBean的相关联系

在线视频

5.1 编写 JavaBean 和使用 JavaBean

5.1.1 编写 JavaBean

编写 JavaBean 就是编写一个 Java 类，这个类创建的一个对象称为一个 JavaBean，简称 Bean。为了让应用程序构建工具知道 Bean 的属性和用法，JavaBean 的命名需要遵循以下规则。

(1) 类中必须提供获取和修改方法用来获取或修改成员变量 XXX 的属性。

getXxx()用来获取属性 xxx。

setXxx()用来修改属性 xxx，属性名首字母必须大写。

(2) 对于 boolean 类型的成员变量，即布尔逻辑类型的属性，允许使用“is”代替上面的“get”和“set”。

(3) 类中声明的方法的访问属性都必须是 public 的。

(4) 类中声明的构造方法必须是 public、无参数的。

Example_5_1_rectangle.java

```
public class Example_5_1_rectangle {
    private double width;
    private double height;

    public Example_5_1_rectangle(){
        this.height = 10;
        this.width = 5;
    }
    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        this.width = width;
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }

    public double recArea() {
        return this.height * this.width;
    }
```

```
    public double recPerimeter() {
        return this.width * 2 + this.height * 2;
    }
}
```

上述案例中，分别设置了长方形的长和宽且均为双精度型变量，同时设置了其 set 和 get 方法，用于获取和修改成员变量的值，在获得成员变量值之后若调用 recArea()或 recPerimeter()方法可返回长方形的面积和周长。

5.1.2 Bean 字节码的保存

为了让 JSP 页面使用 JavaBean，Tomcat 服务器必须使用相应的字节码创建一个 Bean，为了让服务器找到字节码，字节码文件必须保存在特定的目录中，在开发工具中的目录结构如图 5-3 所示。Bean 字节码保存分为以下三个步骤。

图 5-3 Bean 字节码保存目录

(1) 在当前 Web 服务目录下建立如下目录：Web 服务目录\WEB-INF\classes。

(2) 根据类的包名，在目录 classes 下建立相应的子目录：Web 服务目录\WEB-INF\classes\com\programs。

(3) 将 Bean 字节码保存在相应的子目录。

5.1.3 使用 JavaBean

Bean 是通过 JSP 动作标记——useBean 加载使用的，格式如下。

```
<jsp:useBean id = "Bean 的名字" class = "创建 Beans 的字节码" scope = "Bean 有效范围"/>
<jsp:useBean id = "Bean 的名字" class = "创建 Beans 的类" scope = "Bean 有效范围">
</jsp:useBean>
```

例如：

```
<jsp:useBean id = "rectangle" class = "com.programs.rectangle_Example29" scope = "page"/>
```

5.1.4 Bean 的加载原理

JSP 引擎加载 Bean 时首先查找内置 pageContext 对象中是否存在这样的 Bean，如存在则分配给用户，如果不存在，则根据 class 指定的字节码文件创建一个 Bean，并添加到 PageContext 对象中，然后分配给用户。Bean 的加载原理如图 5-4 所示。

注意：

如果修改了字节码文件，也就是修改了.java 源文件，必须重启 JSP 引擎才能使用。

5.1.5 Bean 的生命周期

使用 JSP 动作标记 useBean 来加载使用 Bean。useBean 中的 scope 给出了 Bean 的生命周期，即 scope 取值决定了 JSP 引擎分配给用户的 Bean 的存活时间。

图 5-4　Bean 的加载原理

scope 可以有四种取值，分别是 page、request、session、application，具体的作用和说明如下。

（1）当 Bean 的有效范围是 page 期间，JSP 引擎分配给每个 JSP 页面的 Bean 是互不相同的，也就是说，尽管每个 JSP 页面 Bean 的功能相同，但它们占有不同的内存空间。page 期间的生命周期如图 5-5 所示。

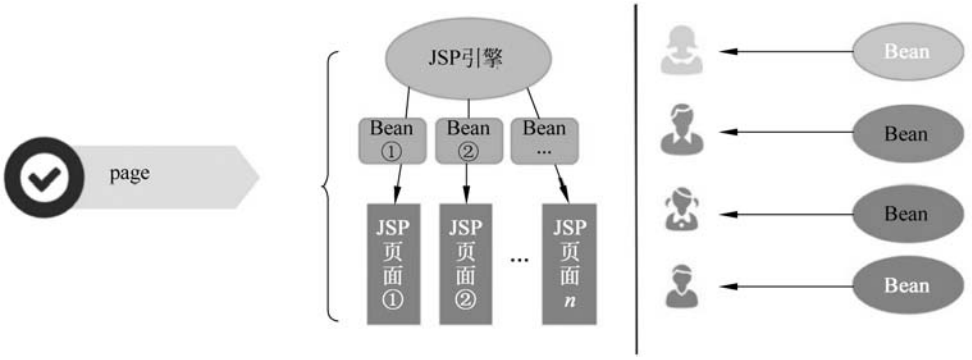

图 5-5　page 期间的生命周期

不同用户的 scope 取值是 page 的 Bean 也是互不相同的，所以一个用户对自己 Bean 属性的改变不会影响到另一个用户。

（2）当 Bean 的有效范围是 request 期间，JSP 引擎分配给每个 JSP 页面 Bean 以及不同用户的 Bean 也是互不相同的。request 期间的生命周期如图 5-6 所示。

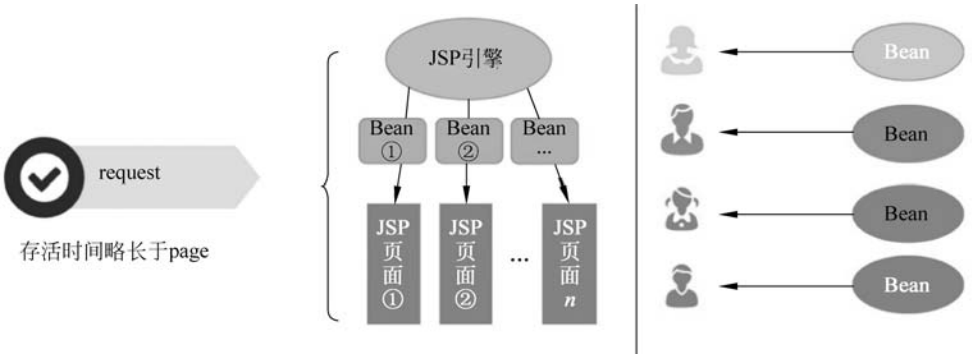

图 5-6　request 期间的生命周期

与 page 不同的是，JSP 引擎对请求做出响应之后，即页面执行完毕后才取消分配给 JSP 页面的这个 Bean，所以 request 存活时间略长于 page。

(3) Bean 的有效范围是用户的会话期间，不同页面的 Bean 是同一个 Bean，所以当用户在某个页面改变了这个 Bean 的属性，其他页面的同一个 Bean 的属性也会发生同样的变化。session 期间的生命周期如图 5-7 所示。

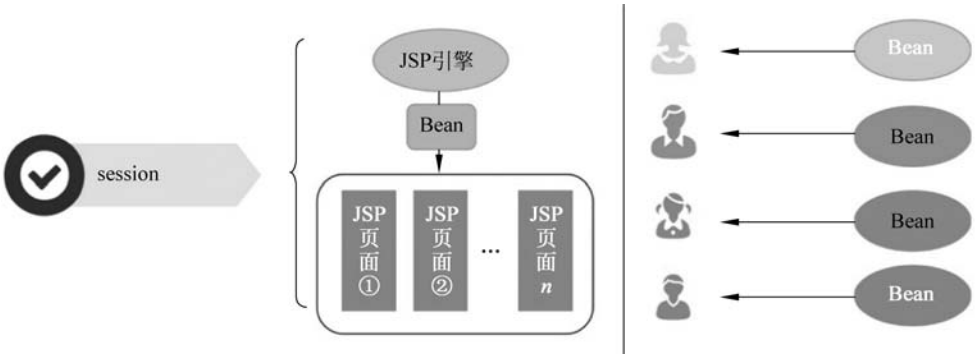

图 5-7 session 期间的生命周期

但不同用户的 scope 取值为 session 的 Bean 是互不相同的。

(4) Bean 的有效范围是 application 期间，JSP 引擎为 Web 服务目录下所有的 JSP 页面分配一个共享的 Bean，不同用户的 Bean 是同一个，直到服务器关闭。application 期间的生命周期如图 5-8 所示。

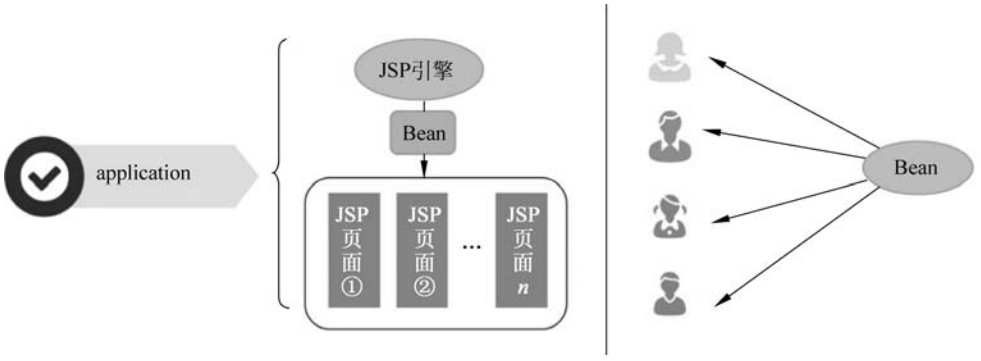

图 5-8 application 期间的生命周期

也就是说，当多个用户同时访问一个 JSP 页面时，任何一个用户对自己 Bean 的属性的改变，都会影响到其他用户。直到服务器关闭，才取消有效范围是 application 的 Bean。

在下面的案例中，创建的 Bean 类是上述 JavaBean 的实例 rectangle.java，使用 scope 为 page 的生命周期。

Example5_2_javabean_scope_page.jsp

```
<%@ page contentType = "text/html;charset = UTF - 8" language = "java" %>
<html>
```

```
<head>
<title>scope - page</title>
</head>
<body>
<jsp:useBean id="rectangle" class="com.programs.Example5_1_rectangle" scope="page" />
<p>长方形的长为:<% =rectangle.getHeight() %></p>
<p>长方形的宽为:<% =rectangle.getWidth() %></p>
<p>修改长方形的长和宽</p>
<%
    rectangle.setHeight(77);
    rectangle.setWidth(11);
 %>
<p>修改后的长方形的长为:<% = rectangle.getHeight() %></p>
<p>修改后的长方形的宽为:<% = rectangle.getWidth() %></p>
<h3>故可以求得目前长方形的周长为:<% = rectangle.recPerimeter() %></h3>
<h3>故可以求得目前长方形的面积为:<% = rectangle.recArea() %></h3>
</body>
</html>
```

页面显示效果如图 5-9 所示。

图 5-9　page 生命周期案例页面显示

下面的案例中使用 scope 取值为 session 的生命周期，其中 id 为 rectangle 的 Bean，使用的 JavaBean 为上述案例中的 Example5_1_rectangle.java。

首先在 Example5_3_a_javabean_scope_session.jsp 页面中获得当前类中初始化的长方形的长和宽，并计算其面积和周长，然后进入链接为 Example5_3_b_javabean_scope_session.jsp 的页面中对长方形的长和宽进行修改并计算修改后的长方形的长和宽，当再次进入之前的页面或在两个页面进行跳转时会一直保持修改之后的值。案例代码和页面显示效果如下，其中页面效果中包含初始化的页面显示，即尚未改变其值和改动长方形的长和宽之后的页面显示，因为 session 生命周期的影响其值一直保持到浏览器关闭。

Example5_3_a_javabean_scope_session.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
<title>scope-session_a</title>
</head>
<body>
<jsp:useBean id="rectangle" class="com.programs.Example5_1_rectangle" scope="session" />
<p>长方形的长为:<%=rectangle.getHeight()%></p>
<p>长方形的宽为:<%=rectangle.getWidth()%></p>

<h3>故可以求得目前长方形的周长为:<%= rectangle.recPerimeter()%></h3>
<h3>故可以求得目前长方形的面积为:<%= rectangle.recArea()%></h3>
<a href="Example5_3_b_javabean_scope_session.jsp">去往
Example5_3_b_javabean_scope_session.jsp 页面</a>
</body>
</html>
```

Example5_3_b_javabean_scope_session.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
<title>scope-session_b</title>
</head>
<body>
<jsp:useBean id="rectangle" class="com.programs.Example5_1_rectangle" scope="session" />
<p>长方形的长为:<%=rectangle.getHeight()%></p>
<p>长方形的宽为:<%=rectangle.getWidth()%></p>
<%
    rectangle.setHeight(35);
    rectangle.setWidth(10);
%>
<p>修改后的长方形的长为:<%= rectangle.getHeight()%></p>
<p>修改后的长方形的宽为:<%= rectangle.getWidth()%></p>
<h3>故可以求得目前长方形的周长为:<%= rectangle.recPerimeter()%></h3>
<h3>故可以求得目前长方形的面积为:<%= rectangle.recArea()%></h3>
<a href="Example5_3_a_javabean_scope_session.jsp">去往
Example5_3_a_javabean_scope_session.jsp 页面</a>
</body>
</html>
```

页面显示效果如图 5-10～图 5-12 所示。

在下面的案例中使用 scope 取值为 application，当第一个用户访问页面时会显示初始化的长方形的长和宽，然后修改长方形的长和宽。因为是 application 的生命周期，所以当其他用户也访问此页面时，长方形的长和宽会显示为最新的值且一直保持不变直到服务器关闭。

图 5-10　session 生命周期案例初始化页面显示

图 5-11　session 生命周期案例改动之后页面显示 1

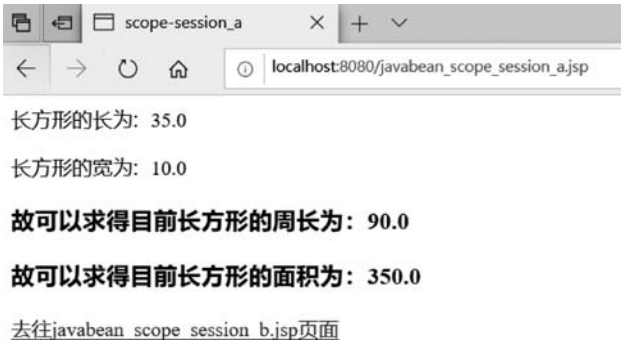

图 5-12　session 生命周期案例改动之后页面显示 2

Example5_4_javabean_scope_application.jsp

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

```
</head>
<body>
<jsp:useBean id="rectangle" class="com.programs.Example5_4_rectangle" scope=
"application" />
<P>长方形的长为:<% =rectangle.getHeight()%></p>
<P>长方形的宽为:<% =rectangle.getWidth()%></p>
<p>修改长方形的长和宽</p>
<%
    rectangle.setHeight(99);
    rectangle.setWidth(22);
 %>
<p>修改后的长方形的长为:<% = rectangle.getHeight()%></p>
<p>修改后的长方形的宽为:<% = rectangle.getWidth()%></p>
<h3>故可以求得目前长方形的周长为:<% = rectangle.recPerimeter()%></h3>
<h3>故可以求得目前长方形的面积为:<% = rectangle.recArea()%></h3>
</body>
</html>
```

页面显示效果如图 5-13 和图 5-14 所示,设定的长方形的长和宽在页面刷新之后依旧未改变,但是当此时关闭服务器将会使其值发生改变。

图 5-13 application 生命周期案例初始化页面显示

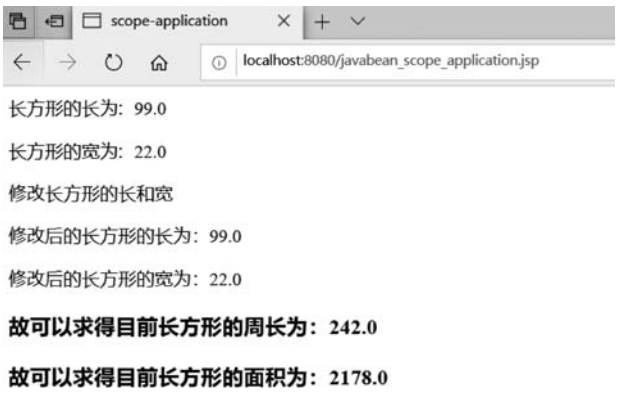

图 5-14 其他用户访问 application 生命周期案例页面显示

如图 5-15 所示表示在 application 生命周期期间，不同用户访问 JSP 页面时都会使用的相同的 Bean，即同一个类文件。

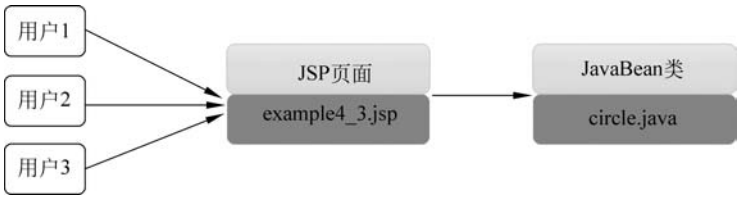

图 5-15　模拟 application 生命周期


在线视频

5.2　获取和修改 Bean 的属性

如果想修改或者使用 Bean 的属性，可以使用 set，get 方法，除此之外，还可以使用动作标记 getProperty、setProperty 获取或修改 Bean 的属性。

需要注意的是，当 JSP 页面使用 getProperty、setProperty 标记获取或修改属性 xxx 时，必须保证 Bean 有相应的 getXxx()和 setXxx()方法；而且在程序片段中直接用 Bean 调用方法不需要方法命名遵守 getXxx()和 setXxx()规则。

5.2.1　getProperty 动作标记

使用 getProperty 动作标记可以获得 Bean 的属性值，并将这个值用串的形式发送给用户的浏览器。使用 getProperty 动作标记之前，必须使用 useBean 动作标记获得相应的 Bean。

getProperty 动作标记的格式如下。

```
<jsp:getProperty name = "Bean 的名字" property = "Bean 的属性" />
```

或

```
<jsp:getProperty name = "Bean 的名字" property = "Bean 的属性"/>
</jsp:getProperty>
```

通过上述动作标记可以得出其定义的基本格式，而该动作标记可以等价于 Java 表达式：<%=bean.getXxx() %>。

5.2.2　setProperty 动作标记

使用 setProperty 动作标记可以设置 Bean 的属性值，同样，在使用 getProperty 动作标记之前，必须使用 useBean 动作标记获得相应的 Bean。

setProperty 动作标记可以通过两种方式设置 Bean 的属性值。

(1) 将 Bean 属性的值设置为一个表达式的值或字符串，其基本格式如下。

```
<jsp:setProperty name = "Bean 的名字" property = "Bean 的属性" value = "<% = expression %>"/>
```

或

```
<jsp:setProperty name = " Bean 的名字" property = "Bean 的属性" value = 字符串/>
```

(2) 通过 HTTP 表单的参数的值来设置 Bean 的相应属性的值。

① 用 HTTP 表单的所有参数的值设置 Bean 相对应的属性的值,参数名必须一致。

```
<jsp:setProperty name = "Bean 的名字" property = " * " />
```

② 用 HTTP 表单的某个参数的值设置 Bean 的某个属性的值,不要求参数名一致。

```
<jsp:setProperty name = "Bean 的名字" property = "Bean 属性名" param = "表单中的参数名" />
```

```
<jsp:setProperty name = "Bean 的变量名" property = "Bean 的属性名" value = "Bean 的属性值" />
<!-- 或者将 setProperty 放在 useBean 的标签体中 -->
<jsp:useBean id = "user" class = "model.User">
<jsp:setProperty name = "user" property = "name" value = "Bob"/>
</jsp:useBean>
```

下面的案例中编写一个购物车的 JavaBean,在 JSP 页面中通过上述用 HTTP 表单的所有参数的值设置 Bean 相对应的属性的值的方法,其中,scope 的值为 request,其代码和页面的显示效果如下。

Example5_5_a_shopping.java

```
public class Example5_5_a_shopping {
    private String goods;
    private int number;

    public String getGoods() {
        return goods;
    }

    public void setGoods(String goods) {
        this.goods = goods;
    }

    public int getNumber() {
        return number;
    }

    public void setNumber(int number) {
        this.number = number;
    }
}
```

Example5_5_b_shopping_setorget.jsp

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

```
<title>购物车</title>
</head>
<body>
<jsp:useBean id="shopping" class="com.programs.Example5_5_a_shopping" scope="request"/>
<jsp:setProperty name="shopping" property="goods" value="方便面"/>
<p>
    购买商品名称:<jsp:getProperty name="shopping" property="goods"/>
</p>
<jsp:setProperty name="shopping" property="number" value="<%=3%>"/>
<p>
    购买商品数量:<jsp:getProperty name="shopping" property="number"/>包
</p>
</body>
</html>
```

页面显示效果如图 5-16 所示。

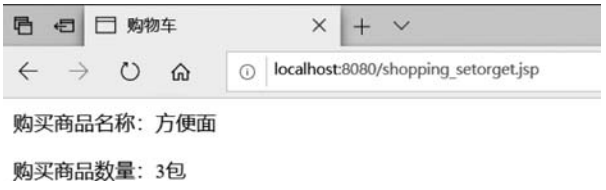

图 5-16　用 HTTP 表单所有参数设置 Bean 的属性值案例页面显示

下面的案例中使用 Example5_6_a_ticket 的类来创建 JavaBean 文件，用上述 HTTP 表单的某个参数的值设置 Bean 的某个属性的值，设置 JSP 页面的 Bean 属性名为 ticket，scope 为 request。采用模拟小型购票系统的方式，在 ticket_setorget_Example34_2.jsp 页面中输入起始站、达到站和购票数量，在 ticket_setorget_receive_Example34_3.jsp 的 JSP 页面中接收用户输入的相关属性值并进行输出。

Example5_6_a_ticket.java

```
public class ticket_Example34_1 {
    private String departureStation;
    private String arrivalStation;
    Private int number;

    public String getDepartureStation() {
        return departureStation;
    }

    public void setDepartureStation(String departureStation) {
        this.departureStation = departureStation;
    }

    public String getArrivalStation() {
```

```
        return arrivalStation;
    }

    public void setArrivalStation(String arrivalStation) {
        this.arrivalStation = arrivalStation;
    }

    public int getNumber() {
        return number;
    }

    public void setNumber(int number) {

    this.number = number;
    }
}
```

Example5_6_b_ticket_setorget.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
<title>购票系统</title>
</head>
<body>
<form action="Example5_6_c_ticket_setorget_receive.jsp">
<p>始发站:</p><input type="text" name="departureStation">
<p>到达站:</p><input type="text" name="arrivalStation">
<p>购票数量:</p><input type="text" name="number">

<input type="submit" value="提交至接收页面">
</form>
<hr>
</body>
</html>
```

Example5_6_c_ticket_setorget_receive.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
<title>购票系统</title>
</head>
<body>
<jsp:useBean id="ticket" class="com.programs.Example5_6_a_ticket" scope="request"/>

<jsp:setProperty name="ticket" property="departureStation"
param ="departureStation" />
<jsp:setProperty name="ticket" property="arrivalStation"
```

```
param = "arrivalStation"/>
< jsp:setProperty name = "ticket" property = "number" param = "number"/>

< p >始发站:</p >< b >< jsp:getProperty name = "ticket"
property = "departureStation"/></b >
< p >到达站:</p >< b >< jsp:getProperty name = "ticket"
property = "arrivalStation"/></b >
< p >购票数量:</p >< b >< jsp:getProperty name = "ticket"
property = "number"/></b >
</body >
</html >
```

页面显示效果如图 5-17 和图 5-18 所示。

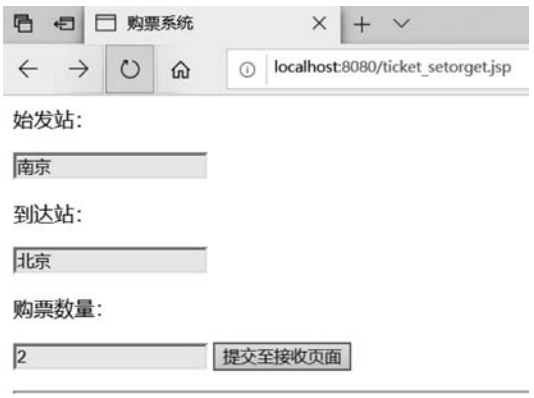

图 5-17　用 HTTP 表单某个参数设置 Bean 的属性值案例提交页面显示

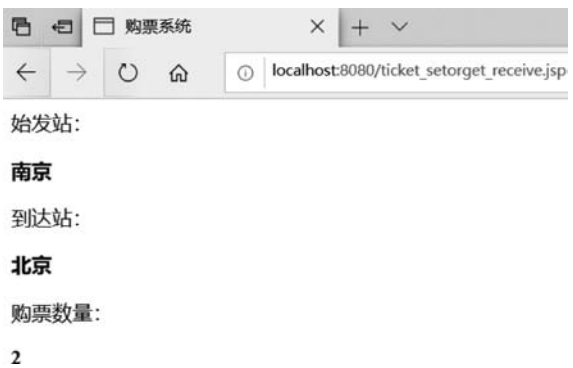

图 5-18　用 HTTP 表单某个参数设置 Bean 的属性值案例结果页面显示


在线视频

5.3　Beans 的辅助类

在写一个 Bean 的时候，除了需要用 import 语句引入 JDK 平台提供的类，可能还需要其他自己编写的一些类。

只要将这些类和创建 Bean 的类写在一个 Java 源文件中即可，但必须将源文件编译后产生的全部字节码文件复制到相应的目录中。

下面的案例中，使用 Java 的内置工具类 Date 类处理时间的格式问题。在 Bean 的类文件 Example5_7_b_timesConvert.java 中，需要通过 Example5_7_b_timesConvert.java 的类文件来辅助完成该功能，该类文件是用来处理时间格式的。

在 Bean 的类文件中调用辅助类的 handle_times()方法，根据 JSP 页面用户提交的表单内容选择相应的时间格式进行输出，通过 convert_times 的方法返回处理之后的时间格式，若在初始的页面中因为提交的内容为空，故返回的结果为空，在下拉框中选择对应的时间格式内容提交之后，页面最终显示处理完成之后的结果，代码和页面显示如下。

Example5_7_a_timesBeans.java

```
public class Example5_7_a_timesBeans {
    private String types;
    public String getTypes() {
        return types;
    }

    public void setTypes(String types) {
        this.types = types;
    }

    public String convert_times() {
        if (types != null) {
            return new Example5_7_b_timesConvert().handle_times(Integer.parseInt(this.
types));
        }
        return null;
    }
}
```

Example5_7_b_timesConvert.java

```
public class Example5_7_b_timesConvert{
    public String handle_times(int types) {
        SimpleDateFormat sdf;
        if (types == 1) {
            sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        } else if (types == 2) {
            sdf = new SimpleDateFormat("yyyy年MM月dd日 HH:mm:ss");
        } else if (types == 3) {
            sdf = new SimpleDateFormat("yyyy/MM/dd HH:mm");
        } else {
            sdf = null;
        }
        Date now = new Date();
        if (sdf != null) {
            return sdf.format(now);
        } else {
```

```
                return null;
            }
        }
    }
```

Example5_7_c_beans_auxiliary.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
<title>Beans 辅助类</title>
</head>
<body>
<jsp:useBean id="times" class="com.programs.Example5_7_a_timesBeans" scope="request"/>
<p>修改当前时间的格式:</p>
<form action="" method="post">
<select name="types" name="types">
<option value="1">yyyy-MM-dd HH:mm:SS</option>
<option value="2">yyyy 年 MM 月 dd 日 HH:mm</option>
<option value="3">yyyy/MM/dd HH:mm:ss</option>
</select>
<input type="submit" value="提交">
</form>
<jsp:setProperty name="times" property="types" param="types"/>
处理完成之后的结果如下:
<%
    String result = times.convert_times();
    out.print(result);
%>
</body>
</html>
```

页面显示效果如图 5-19～图 5-21 所示。

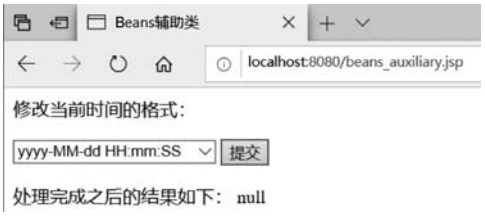

图 5-19　Beans 辅助类案例初始页面显示

图 5-20　Beans 辅助类案例初始选择页面显示

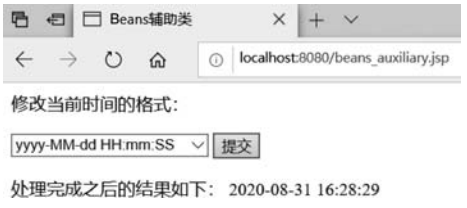

图 5-21　Beans 辅助类案例结果页面显示

5.4 JSP 和 Bean 结合的简单例子

JSP 页面中调用 Bean 将数据处理代码从页面中分离出来，实现代码复用，以便有效地维护一个 Web 应用。

下面的案例中使用 JavaBean 为 Student 对用户输入的成绩进行汇总。通过 JSP 页面提交表单，用户在 JSP 页面提交表单后在 JavaBean 中完成该功能，具体的代码和页面显示如下。

Example5_8_a_Student.java

```
public class Example5_8_a_Student {
    private String name;
    private double math,english,computer,sum = 0;

    public String getName() {
        return name;}
    public void setName(String name) {
        this.name = name;}
    public double getMath() {
        return math;}
    public void setMath(double math) {
        this.math = math;}
    public double getEnglish() {
        return english;}
    public void setEnglish(double english) {
        this.english = english;}
    public double getComputer() {
        return computer;}
    public void setComputer(double computer) {
        this.computer = computer;}
    public double getSum() {
        sum = math + english + computer;
        return sum;}
    public void setSum(double sum) {
        this.sum = sum;}
}
```

Example5_8_b_Student_grade.jsp

```
<%@ page contentType = "text/html;charset = UTF - 8" language = "java" %>
<html>
<head>
<title>求学生成绩</title>
</head>
<body>
<jsp:useBean id = "student" class = "com.programs.Example5_8_a_Student" scope = "request"/>
<form action = "" method = "post">
```

```
<b>输入学生的姓名</b><input type="text" name="name">
<p>数学成绩:</p><input type="text" name="math">
<p>英语成绩:</p><input type="text" name="english">
<p>计算机成绩:</p><input type="text" name="computer">
<input type="submit" value="提交">
</form>
<jsp:setProperty name="student" property="*"/>
<p>总成绩为:</p><jsp:getProperty name="student" property="sum"/>
</body>
</html>
```

页面显示效果如图 5-22 和图 5-23 所示。

图 5-22　JSP＋Bean 结合实例案例初始化页面显示

图 5-23　JSP＋Bean 结合实例案例结果页面显示

5.5 上机案例

本章的内容已经结束，读者在阅读本节内容时是否遇到了难以解决的问题呢？关于JSP与Bean的结合案例读者是否掌握了呢？当这里JSP动作标记已经介绍完毕了，读者在日常学习过程中可能会遇到一些其他的问题，不妨将这些问题记录下来，以便之后复习本章内容时可以做针对性学习。

我们试想公司的信息系统是如何管理的呢？结合JavaBean，试着猜想是否会有一个员工的类，这个类中含有员工的哪些重要信息？本节的上机案例就可以实现一个员工信息的提交和查看，其中包含员工所需要的重要信息，参考代码如下。

Example5_9_a_exam.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>上机案例</title>
</head>
<body>
<h3>添加员工信息</h3>
<form action="Example5_9_b_exam.jsp" method="post">
<b>员工姓名:</b><input type="text" name="username"><br>
<b>员工年龄:</b><input type="text" name="age"><br>
<b>员工部门:</b><input type="text" name="department"><br>
<b>员工工号:</b><input type="text" name="departmentID"><br>
<input type="submit" value="提交">
</form>
</body>
</html>
```

Example5_9_b_exam.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>上机案例</title>
</head>
<body>
<jsp:useBean id="staff" class="com.programs.Example5_9_c_Staff" scope="request"/>
<jsp:setProperty name="staff" property="*"></jsp:setProperty>
<p>员工姓名:<jsp:getProperty name="staff"
```

```
property = "username"></jsp:getProperty></p>
<p>员工年龄:<jsp:getProperty name = "staff"
property = "age"></jsp:getProperty></p>
<p>员工部门:<jsp:getProperty name = "staff"
property = "department"></jsp:getProperty></p>
<p>员工工号:<jsp:getProperty name = "staff"
property = "departmentID"></jsp:getProperty></p>

</body>
</html>
```

Example5_9_c_Staff.java

```
<%@ page language = "java" contentType = "text/html; charset = UTF - 8"
         pageEncoding = "UTF - 8" %>
<!DOCTYPE html>
<html>
<head>
<meta charset = "UTF - 8">
<title>上机案例</title>
</head>
<body>
<jsp:useBean id = "staff" class = "com.programs.Example5_9_c_Staff" scope = "request"/>
<jsp:setProperty name = "staff" property = " * "></jsp:setProperty>
<p>员工姓名:<jsp:getProperty name = "staff"
property = "username"></jsp:getProperty></p>
<p>员工年龄:<jsp:getProperty name = "staff"
property = "age"></jsp:getProperty></p>
<p>员工部门:<jsp:getProperty name = "staff"
property = "department"></jsp:getProperty></p>
<p>员工工号:<jsp:getProperty name = "staff"
property = "departmentID"></jsp:getProperty></p>

</body>
</html>
```

小结

(1) JavaBean 是一个可重复使用的软件组件,是遵循一定标准、用 Java 语言编写的一个类,该类的一个实例称作一个 JavaBean。

(2) 一个 JSP 页面可以将数据的处理过程指派给一个或几个 Bean 来完成,只需在 JSP 页面中调用这个 Bean 即可。在 JSP 页面中调用 Bean 可以将数据的处理代码从页面中分离出来,实现代码复用,更有效地维护一个 Web 应用。

(3) Bean 的生命周期分为 page、request、session 和 application。

本章小结可参考图 5-24 所示。

图 5-24 第 5 章小结

习题

1. JavaBean 可以通过相关 JSP 动作指令进行调用，下面哪个不是 JavaBean 可以使用的 JSP 动作指令？（ ）

A. 完成一定运算和操作，包含一些特定的或通用的方法

B. 负责数据的存取

C. 接收客户端的请求，将处理结果返回客户端

D. 在多台机器上跨几个地址空间运行

2. 在常见的 JavaBean 中，其属性必须声明为 private，方法必须声明为（ ）访问类型。

A. private B. static C. protect D. public

3. 关于 JavaBean，下列的叙述哪一项不正确？（ ）

A. JavaBean 的类必须是具体的、公开的，并且具有无参数的构造器

B. JavaBean 的类属性是私有的，要通过公共方法进行访问

C. JavaBean 和 Servlet 一样，使用之前必须在项目的 Web.xml 中注册

D. JavaBean 属性和表单控件名称能很好地耦合，得到表单提交的参数

4. JavaBean 是一个_____类，其中必须包含一个_____方法。

5. 在 JavaBean 中通过使用_____方法设置 Bean 的私有属性值，通过使用_____方法获取 Bean 的私有属性值。

6. JavaBean 的作用域中使用范围最大的是_____。

7. 在 JSP 中使用 JavaBean 的标签是_____，其中 id 的用途是_____。

8. 简述 JavaBean 的加载原理。

9. 简述 JavaBean 的生命周期及其作用。

10. 编写一个 JavaBean 和 JSP 结合的案例，实现学生个人信息的提交和获取。