第3章 使用 Struts 2 框架提高开发效率

本章学习目标

- 了解 Struts 2 标签的作用及在程序中的使用步骤。
- 掌握表单标签与常见的表单元素 Struts 2 标签的使用方法。
- 掌握常用的 Struts 2 数据标签、控制标签的使用方法。
- 能通过 Struts 2 的 OGNL 访问数据对象。
- 能综合应用 Struts 2 框架标签、OGNL 语言提高软件开发效率。

前面案例已经介绍了利用 Struts 2 框架实现 MVC 应用程序，其中 V（视图）是传统的 JSP 页面，M（模型）是传统的 JavaBean，而 C（控制器）是 Struts 2 的 Action。

Struts 2 框架还提供了许多方便 MVC 模式编程的技术。例如，通过 OGNL 语言很容易使数据对象在 M、V、C 各层之间进行交互；通过 Struts 2 标签可以在视图层（V）上访问与展现数据；通过国际化很容易开发能适合多种国际语言的应用程序等。

下面重点介绍利用 Struts 2 标签、OGNL 对象图导航语言提高软件的开发效率。

[redacted]标签的应用

Struts 2 是一个表现层的 MVC 框架，它的重点也是控制器和视图。控制器由 Action 类提供支持，而视图主要由 Struts 2 标签来提供支持。

所谓 Struts 2 标签就是 Struts 2 框架提供的一些页面功能组件，类似其他网页标签的使用，可以在网页上快速地实现某个功能，这样可以大量减少 Java 代码的编写。Struts 2 标签由 Struts 2 驱动包提供，即配置好 Struts 2 开发环境后就可以在 JSP 等页面上使用 Struts 2 标签了。

Struts 2 标签库提供了大量的标签，它们既能大大简化数据的输出显示，也能大大简化页面效果的编码，从而大幅度提高软件开发效率。

在 Struts 2 标签库之前已有 JSP 自定义标签、JSTL（JSP 标准标签库）、struts 1 标签库等，但 Struts 2 标签库提供的标签能简化应用的表现逻辑，具有明显的优点且编码简单。第 2 章已经在案例中使用了 Struts 2 标签显示数据，从中可知只要在 JSP 页面中加<%@taglib prefix="s" uri="/struts-tags"%>语句就可以直接使用 Struts 2 标签了。

Struts 2 标签非常丰富，几乎包括视图层所需要的各种功能的实现技术。Struts 2 标签主要有表单标签、表单元素标签、数据标签、控制标签以及其他非表单标签等。

Struts 2 标签的优点主要表现在以下几点。

- Struts 2 标签库的标签不依赖任何表现层技术。也就是说，其大部分标签可以在各种表现层技术中使用（包括 JSP）。
- Struts 2 把各种标签都定义在一个 s 标签库中，这些标签虽然非常庞大，但都定义在 URI 为“/struts-tags”的命名空间下，所以使用非常方便。

下面分别介绍上述 Struts 2 的标签及其应用。

3.1.1　表单标签 form

表单标签包括<s:form>标签和表单元素标签（表单元素标签见下节介绍）。

我们知道，网页中<form>表单可将用户界面数据的数据传递到服务器进行处理。其语法形式是<form>…</form>，且需通过 request 对象进行数据的传递（参见第 1 章案例 1-1 中的代码）。

从案例 1-1 的代码我们可以看出，通过<form>表单处理用户输入数据需要编写大量的处理代码，这样给程序员编码带来了许多麻烦。但是，在 Struts 2 框架下处理用户输入的数据时，可以通过 Struts 2 标签大大简化代码的编写。

如果要处理 Struts 2 封装在 Action 属性中的数据，只需要用 Struts 2 的表单标签<s:form></s:form>，就可以实现界面中的数据直接输入到服务器端的 Action 属性中（其实在第 2 章案例 2-5 中已经见过它们的使用）。

在 Struts 2 框架支持下，在视图层（V）的 JSP 网页程序中使用 Struts 2 标签，只需要在 JSP 网页中加上<%@taglib prefix="s" uri="/struts-tags"%>定义标签的语句，就可以使用 Strust 2 表单标签<s:form>，并可以将其和其他各种表单元素标签配合使用，完成各种视图层数据的展示。

Struts 2 表单元素标签包括<s:combobox>、<s:checkboxlist>和<s:radio>等。一般这些表单标签与 HTML 表单元素之间有一一对应关系，读者可以查阅相关文献进行了解。

3.1.2　表单元素标签

Struts 2 的表单元素标签在<s:form>中进行数据的处理与展示。这些标签有<s:combobox>、<s:checkboxlist>和<s:radio>等，它们分别以不同的形式对数据进行展示与处理。下面通过案例介绍表单标签<s:form>及表单元素标签<s:combobox>、<s:checkboxlist>和<s:radio>的应用。

【案例 3-1】分别用 Struts 2 表单标签<s:form>及表单元素标签<s:combobox>、<s:checkboxlist>和<s:radio>在 JSP 页面中显示数组中定义的数据。

本案例分为 3 个部分，即在 Struts 2 表单标签<s:form>中分别采用表单元素标签<s:combobox>、<s:checkboxlist>和<s:radio>，在 JSP 页面中以下拉组合框、复选框、单选框的形式显示定义在数组中的课程数据，以便用户进行选择操作。

1. 表单元素标签 combobox

Struts 2 表单元素标签<s:combobox>以组合框的形式对数据进行处理，即它将数组中的数据以组合框的形式在 JSP 页面中提供给用户选择，用户选择的数据项则通过另一个文本框显示出来，并存放在一个变量“subject”中。

例如，学生选择课程信息时，能供选的课程有 3 个：“Java 程序设计基础”、“JSP 程序设计”、“SSH 框架开发技术”，则这 3 个数据存放在一个数组中，通过下列语句实现下拉组合框。

```
<s:form>
<s:combobox label="请选择课程" theme="css_xhtml" labelposition="top"
    list="{'Java 程序设计基础','JSP 程序设计','SSH 框架开发技术'}"
    size="20" maxlength="20" name="subject"/>
</s:form>
```

上述语句是一个 Struts 2 表单标签<s:form> </s:form>，其中<s:combobox ... />语句是其表单元素标签，它们的运行效果就是图 3-1 所示的下拉组合框。在这个组合框中，有一个文本框和下拉框，下拉框中显示了供选择的数据，而选择的结果显示在文本框中。用户既可以进行编辑输入，也可以进行选择输入。所有这些功能均由<s:combobox>标签实现。

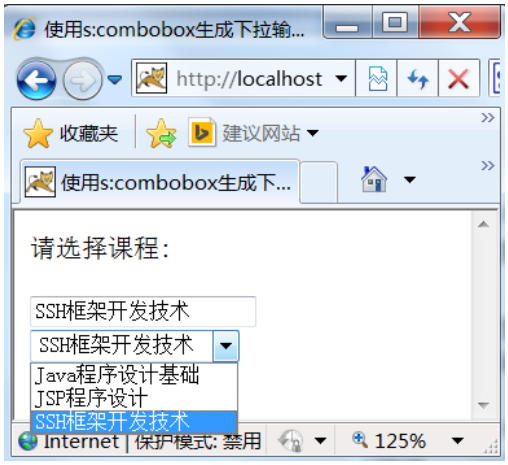

图 3-1　标签<s:combobox>的运行结果

在上述语句中，用 theme="css_xhtml"设置 Struts 2 的内置主题。Struts 2 的所有界面（UI）标签都是基于主题和模板的，而模板只是一个 UI 标签的外在表现形式。例如，我们使用<s:combobox…./>标签时，Struts 2 就会利用 combobox 模板来生成一个有模板特色的下拉组合框。如果为所有的 UI 标签都提供对应的模板，那么此系列的模板就会形成一个主题。

Struts 2 提供了 4 个主题，分别是 simple、xhtml、css_xhtml 和 ajax。

程序员还可以用其他的参数生成不同风格的下拉组合框。通过相应的参数设置，可以调整显示的格式，以便美化界面。

注意：该表单标签的 name 属性指定的变量将存放用户选择的值，它可以对应 Action 类中的属性。当该表单对应的 Action 对应的属性有值时，表单元素就会显示该属性值，且该值也是该表单元素的 value 的值，这样用 Struts 2 标签表单元素标签进行编程时，不但容易实现数据在 MVC 各层之间的传递，并且容易定制数据的显示形式。后面的表单元素同理，因此不再重

复解释了。

首先，要在 JSP 页面中使用 Struts 2 标签，只要在 Java EE 项目添加 Struts 2 的支持便可（前面的案例已经介绍了项目获取 Struts 2 的支持与配置，这里不再赘述）。其次，在 JSP 页面上还需要添加如下标签前缀定义语句。

```
<%@taglib prefix="s" uri="/struts-tags"%>
```

上面的语句将 Struts 2 的标签前缀定义为“s”，就像 Java 中生成的一个对象名；uri="/struts-tags"表示标签库的路径。相当于 import 一个具体的类。一旦 JSP 页面中添加了该语句，后面所介绍的所有 Struts 2 标签都可以使用了，因此不再进行强调。

2. 表单元素标签 checkboxlist

Struts 2 表单元素标签<s:checkboxlist>标签以复选框的形式供用户输入数据。如下列代码将“Java 程序设计基础”、“JSP 程序设计”、“SSH 框架开发技术”3 门课程供学生选择，学生任意选择其中的课程（1 门或多门，也可以不选）时，可以使用 checkboxlist 表单元素标签。

如果用复选框供用户进行选择，其代码如下，该代码运行的结果如图 3-2 所示。

```
<s:form>
<s:checkboxlist label="请选择课程" theme="css_xhtml" labelposition="top"
      list="{'Java 程序设计基础' , 'JSP 程序设计' , 'SSH 框架开发技术'}"
       name="subjects"/>
</s:form>
```

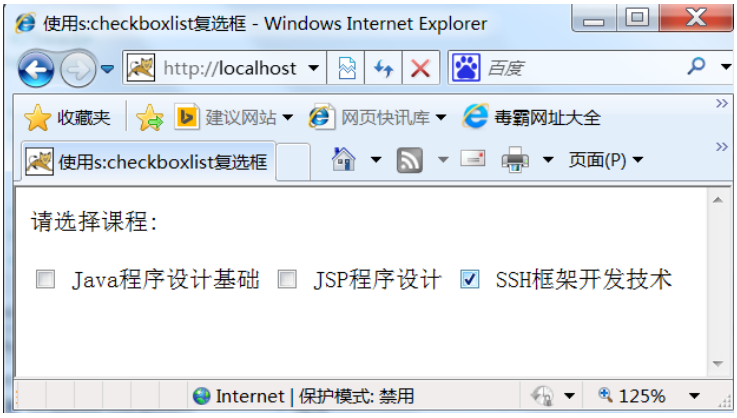

图 3-2　<s:checkboxlist>的运行结果

上述代码用 checkboxlist 表单元素标签获取的数据将存放在数组数据变量“subjects”中。<s:checkboxlist>标签其他部分的含义同<s:combobox>，此处不重复说明。

另外，也可以将一个类中定义的数据用 checkboxlist 标签显示出来供用户选择。例如，以下 Java 类代码创建了一个 Book 类型对象的数组（Book 类的定义略），它存放了 3 本教材及其作者。

```
public class BookService{
    public Book[] getBooks()    {
        return new Book[]
        {
```

```
                new Book("软件项目开发与管理","牛德雄"),
                new Book("JSP 程序开发技术","刘晓林"),
                new Book("基于 SSH 框架的软件开发","熊君丽")
            };
        }
    }
```

下面希望通过 checkboxlist 标签显示这 3 个作者，以便供读者选择，效果如图 3-3 所示。在使用 checkboxlist 标签时，要访问该 Java 类获得需要显示的数据，然后再通过<s:bean>数据标签定义数据对象，再通过如 list="#bs.books"访问数据对象数组，并通过 listKey="name"，listValue="author"的定义指定 Map 类型的<键、值>对，通过 listValue="author"指定应显示的数据，代码实现如下。

```
<s:form>
<s:bean name="beans.BookService" id="bs"/>
<s:checkboxlist name="b" label="请选择教材的作者" labelposition="top"
    list="#bs.books"
    listKey="name"
    listValue="author"/>
</s:form>
```

上述代码运行的结果如图 3-3 所示。

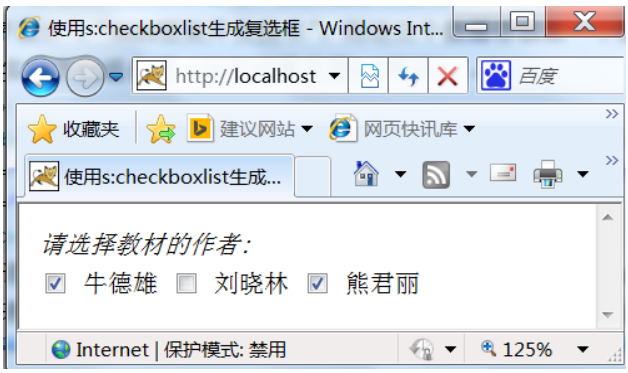

图 3-3　使用<s:checkboxlist>访问数据对象运行的结果

本案例中用到了数据标签<s:bean>，它同 new 语句一样将一个类实例化成一个对象，对象名就是 id="bs"中指定的“bs”，然后可以通过#bs 来访问该对象。关于<s:bean>数据标签以及通过“#”访问数据对象，本书后面将会介绍。

3. 表单元素标签 radio

Struts 2 表单元素标签<s:radio>标签以单选框的形式提供用户的数据输入。如下列代码将提供“Java 程序设计基础”、“JSP 程序设计”、“SSH 框架开发技术”3 门课程供学生选择，学生只能任意选择其中的一门课程，这时候就可以使用 radio 表单元素标签。

如果用单选框供用户进行选择，其代码如下，该代码运行的结果如图 3-4 所示。

```
<s:form>
<s:radio label="请选择课程" theme="css_xhtml" labelposition="top"
    list="{'Java 程序设计基础' , 'JSP 程序设计' , 'SSH 框架开发技术'}"
```

```
        name="subject"/>
</s:form>
```

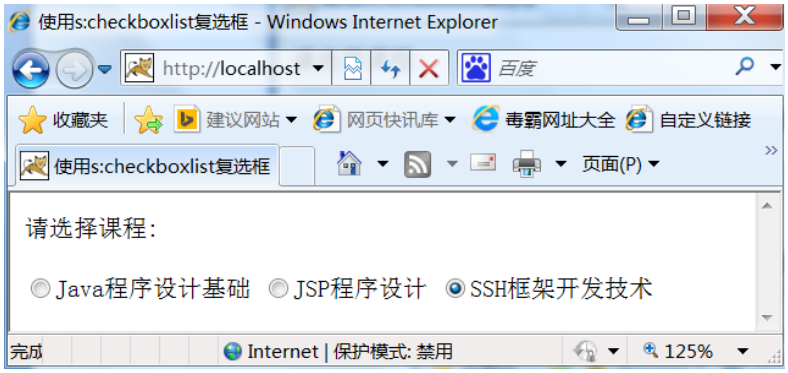

图 3-4　<s:radio>标签运行结果

上述代码用 radio 表单元素标签获取的数据将存放在数据变量“subject”中。<s:radio>标签其他部分的含义同<s:checkboxlist>，此处不重复说明。

另外，与 checkboxlist 标签一样，也可以将一个类中定义的数据用 radio 标签显示出来供用户选择。与 checkboxlist 例子中的 Book 类及 BookService 类中存放的 3 本教材及其作者数据类似，可按如下方式使用 radio 标签对类中的数据进行访问。

```
<s:form>
<s:bean name="beans.BookService" id="bs"/>
<s:radio name="b" label="请选择教材的作者" labelposition="top"
        list="#bs.books"
        listKey="name"
        listValue="author"/>
</s:form>
```

上述代码运行的结果如图 3-5 所示。

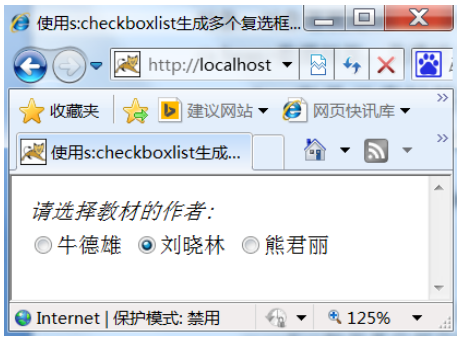

图 3-5　使用<s:radio>访问数据对象运行的结果

上述代码中，radio 标签语句中其他部分与 checkboxlist 标签含义相同，这里不再重复介绍。

类似的表单标签还有<s:select>列表选择标签等，这里不再一一介绍，有需要的读者请参考其他教材。下面罗列出这些表单元素标签。

- datetimepicker：生成一个日期、时间下拉框进行日期数据的选择。

- doubleselect：生成一个级联列表框（两个下拉列表框）进行数据选择输入。但 doubleselect 一定要设置<s:form>的 action 属性。
- select：生成一个下拉列表框，如果为该元素指定 list 属性，系统会用 list 属性指定的集合来生成下拉列表框的选项。
- optgroup：生成一个下拉列表框的选项组进行数据的选择输入。

3.1.3 数据标签

Struts 2 框架丰富的标签是其优势之一，除了前面介绍的表单标签与表单元素标签外，还有数据标签、控制标签以及其他非表单标签等。

本节介绍 Struts 2 框架几个主要的数据标签及其使用。数据标签主要访问 Struts 2 框架 ValueStack（值栈）中的数据，用于提供各种数据访问相关的功能，它可以显示一个 Action 中的属性。数据标签主要包含下面几个。

- bean：用于创建一个 JavaBean 实例，如果指定 id 属性，则可以将创建的 JavaBean 实例投入到 Stack Context 中。
- property：property 标签用于获取数据对象中的值可以输出，包括 ValueStack StackContext Stack Action 中的数据值。
- push：用于将某个值放入 ValueStack 的栈顶。
- set：用于设置一个新的变量，并可以将变量放入指定的范围内。
- param：用于设置一个参数，通常用作 bean 标签和 url 标签的子标签。

另外，还有如下几个常用的 Struts 2 数据标签。

- action：用于在 JSP 页面直接调用一个 Action，如果指定 executeResult 参数，还可以将 Action 的处理结果包含到本页面中来。
- date：用于格式化输出日期数据。其可选属性 nice 可以指示是否要输出到当前时间之间的天数。
- debug：用于在页面生成一个调试链接，当单击该链接时，可以看到 ValueStack 和 StackContext 中的内容。
- url：用于生成一个 URL 地址。

【案例 3-2】 通过 Struts 2 数据标签 bean、property、param、push、set 分别处理与显示数据对象（JavaBean）、Action 属性、ValueStack、ValueContext 中的数据。

1. 操作数据对象中的数据

使用 bean 标签可以代替 Java 代码对数据对象（JavaBean）进行操作，包括进行实例化及数据的存取。例如，如下 JavaBean 封装了一个人（Person）的基本信息。

```
package beans;
public class People
{
    private String name;
    private int age;
    //以下为 setter/getter 方法（略）
    ……
}
```

在 JSP 页面中可以通过 bean 标签直接对其实例化生成数据对象（不需要用 Java 的 new 语句），并进行操作，具体代码如下。

```
<body>
<s:bean name="beans.People" id="p">
        <s:param name="name" value="'张国强'"/>
        <s:param name="age" value="22"/>
</s:bean>
<s:property value="#p.name"/><br>
<s:property value="#p.age"/>
</body>
```

上述代码的运行结果如图 3-6 所示。

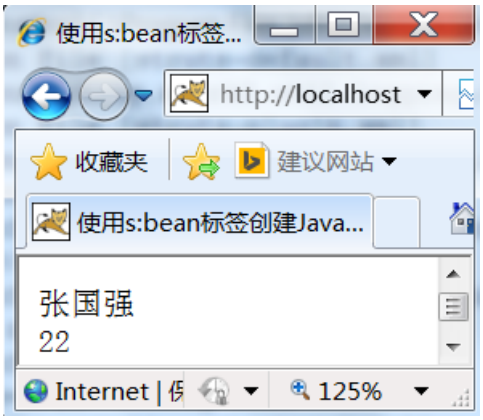

图 3-6　bean 标签代码运行结果

在上述代码中，使用<s:bean>数据标签通过 Person 类实例化一个数据对象 p，然后通过<s:param>标签对 p 中的两个属性赋值。最后通过<s:property>标签显示 p 对象中的这两个属性值。

在上面的代码中，<s:bean>标签通过 id="p"指定生成的对象名为“p”，然后通过<s:property>属性标签访问并显示 p 中的属性值。这里，“#p”中的“#”是 Struts 2 框架的 OGNL（对象图导航语言，后面介绍），它是在一般数据对象前加的前缀，其目的是区别一般的数据对象属性访问与 Action 属性访问。即在一般数据对象前需加“#”前缀，而对 Action 属性的访问则不需要，而是直接用属性名进行访问。

对于上述数据对象，可以通过 push 和 set 标签进行操作。例如，上述代码通过 bean 标签生成了一个数据对象，可以通过 push 标签将该对象压入值栈 ValueStack 的栈顶，然后就可以不带“#”访问该值栈中的数据，参见如下代码。

```
<body>
<s:bean name="beans.People" id="p">
        <s:param name="name" value="'张国强'"/>
        <s:param name="age" value="22"/>
</s:bean>
<s:push value="#p">
        <s:property value="name"/><br>
        <s:property value="age"/><br>
```

```
    </s:push>
    </body>
```

上述代码的运行效果与图 3-6 类似，但是其区别是<s:property value="name"/>访问栈顶对象的属性代码中没有“#”指引的对象，而是直接访问属性“name”。

2. 操作 StackContext 中的数据

ValueStack 由 OGNL 框架实现，可以把它简单地看作一个栈（stack）。而 Stack Context（Stack 的上下文，保存方式是 MAP 类型），它包含一系列对象，包括 request、session、attr、application、map 等。ValueStack 中保存的值可以直接取用，而 StackContext 中的需要在前面加“#”。例如，有如下代码：

```
    <body>
    1.创建一个数据对象 p 并给其属性赋值<br>
    <s:bean name="beans.People" id="p">
        <s:param name="name" value="'张国强'"/>
        <s:param name="age" value="23"/>
    </s:bean>
    2.用 set 标签在 StackContext 中在默认（request）范围内创建对象 q1，将 p 值存放到 q1 中，并进行显示。
<br>
    <s:set value="#p" name="q1"/>
    <s:property value="#q1.name"/><br>
    <s:property value="#q1.age"/><br>
    3.用 set 标签在 Stack Context 中将对象 p 值存放到 application 范围内 q2 中，并进行显示。<br>
    <s:set value="#p" name="q2" scope="application"/>
    <s:property value="#attr.q2.name"/><br>
    <s:property value="#attr.q2.age"/><br>
    4.用 set 标签在 Stack Context 中将对象 p 值放入 session 范围内 q3 中，并用 EL 进行显示。<br>
    <s:set value="#p" name="q3" scope="session"/>
    ${sessionScope.q3.name}<br>
    ${sessionScope.q3.age}<br>
    </body>
```

上述代码运行结果如图 3-7 所示。

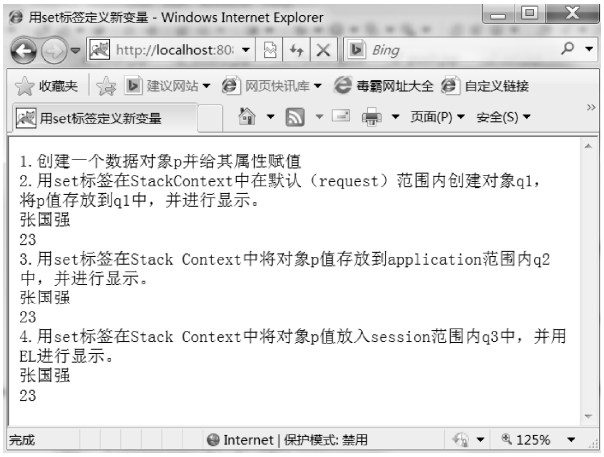

图 3-7　set 标签代码运行结果

上面的代码首先根据 bean 创建一个数据对象 p 并对其属性进行赋值，然后分别用 set 标签创建 q1、q2、q3 三个对象，其范围分别为 StackContext 中的 request、application、session，然后通过<s:property>标签或 EL 表达式显示这些数据对象中的数据，如图 3-7 所示。注意，显示 q1、q2、q3 对象中的数据需要在前面加“#”。

Struts 2 数据标签还有 action、date、debug、url 等。action 标签用于在 JSP 页面直接调用一个 Action；date 标签用于格式化输出一个日期数据；debug 标签用于在页面生成一个调试链接，用于查看 ValueStack 和 StackContext 中的内容；url 标签用于生成一个 URL 地址。这些标签的使用比较简单，具体可参看其他文献。

3.1.4　控制标签

在交互式页面中，常常有如采用选择、循环等实现交互功能，这些功能如果采用 Java 代码进行编写会显得烦琐与臃肿。有一些控制标签就是为了完成这些任务而开发的。同样，Struts 2 也有类似的控制标签，如 if、elseif、iterator 等。由于迭代操作往往是对某个集合的迭代，所以还有一些对集合进行操作的控制标签，如 append 标签用于将多个集合拼接成一个集合；generator 标签用于将一个字符串拼接成一个集合，然后可对这个集合进行操作。

Struts 2 控制标签主要有以下几个。

- if：用于控制选择输出的标签。
- elseif：与 if 标签结合使用，用于控制选择输出的标签。
- else：与 if 标签结合使用，用于控制选择输出的标签。
- iterator：是一个迭代器，用于迭代输出一个集合的元素。
- append：用于将多个集合拼接成一个集合。
- generator：用于将一个字符串解析成一个集合。
- merge：用于将多个集合拼接成一个集合（但与 append 的使用方式不同）。
- sort：用于对集合元素进行排序。
- subset：截取集合的某些元素形成一个新的子集。

【案例 3-3】 通过控制标签按要求采用列表的方式在界面上显示某个集合的元素。

1. 用循环标签 iterator 列表显示某集合元素

如果需要在页面中实现循环功能，则可以使用 iterator 标签。如有一个集合需要在页面上列表显示其中的元素，这时就需要循环。如果用 Java 代码实现则比较烦琐，如果用 iterator 迭代标签，则代码显得简洁且开发效率高。

以下代码就是通过 iterator 标签迭代显示某集合中的元素，其运行结果如图 3-8 所示。

```
<body>
<table border="1" width="200">
<s:iterator value="{'软件项目开发与管理','JSP 程序设计','SSH 框架开发技术'}" id="name">
    <tr>
        <td><s:property value="name"/></td>
    </tr>
</s:iterator>
</table>
</body>
```

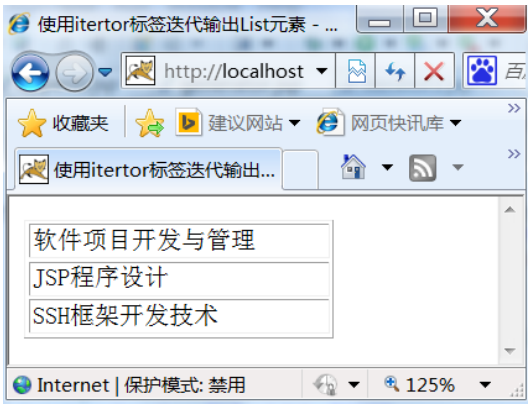

图 3-8　iterator 标签运行结果

图 3-8 使用 Struts 2 迭代控制标签 iterator 循环显示了某个集合中的元素，实现的代码非常简洁。

2. 将集合元素按奇偶行的背景色不同进行列表显示

在以下代码中，有两个 Map 类型的集合，现在要将它们拼接成一个集合并进行列表显示。显示时如果是奇数行则背景色为深色，如果是偶数行则背景色为浅色，如图 3-9 所示。

这里用 append 标签将两个集合拼接起来，拼接后的集合存放在 classList 变量中。然后通过 iterator 标签对该集合的元素进行遍历与显示。在遍历时将每个元素状态定义为“st”，如果该元素为奇数（test="#st.odd"），则该行的背景色为深色，这时需要用 if 标签进行判断，再通过 property 标签分别显示该 Map 的“键(key)”与“值(value)”的数据。

```
<body>
<s:append id="classList">
        <s:param value="#{'软件项目开发与管理':'牛德雄','JSP 程序设计':'刘晓林'}" />
        <s:param value="#{'SSH 框架开发技术':'熊君丽'}" />
</s:append>
<table border="1" width="240">
<s:iterator value="#classList" status="st">
        <tr <s:if test="#st.odd">style="background-color:#bbbbbb"</s:if>>
                <td><s:property value="key"/></td>
                <td><s:property value="value"/></td>
        </tr>
</s:iterator>
</table>
</body>
```

上述代码中使用了如下控制标签。

- append 标签拼接集合元素。
- if 标签进行条件判断。
- iterator 标签进行循环控制。

上述代码应用了部分控制标签进行动态页面编码，简化了编码工作量，其运行效果如图 3-9 所示。

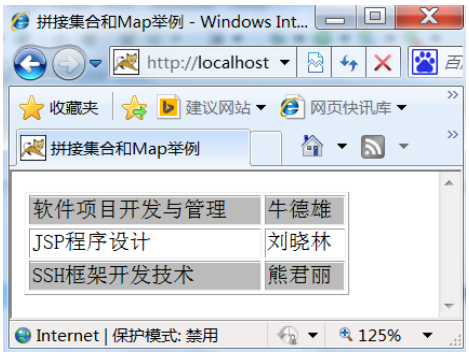

图 3-9　代码运行结果

上面代码中使用了条件标签、循环（迭代）标签等标签实现了列表显示集合元素，代码简洁、开发工作量小。如果用 Java 代码实现这些功能，则代码重复，开发工作量大，并且难以分工与维护。Struts 2 的控制标签在交互界面编码方面为开发者提供了许多便利。

3. 通过字符串生成集合并列表显示

也可以使用 generator 将一个字符串生成一个集合。如在下面的代码中，将字符串“'软件项目开发与管理,JSP 程序设计,SSH 框架开发技术'”分隔符设置为“,”，生成一个集合并列表显示其中的元素。

```
<body>
<table border="1" width="240">
<s:generator val="'软件项目开发与管理,JSP 程序设计,SSH 框架开发技术'" separator=",">
<s:iterator status="st">
        <tr <s:if test="#st.odd">style="background-color:#bbbbbb"</s:if>>
            <td><s:property/></td>
        </tr>
</s:iterator>
</s:generator>
</table>
</body>
```

上述代码运行结果如图 3-10 所示。

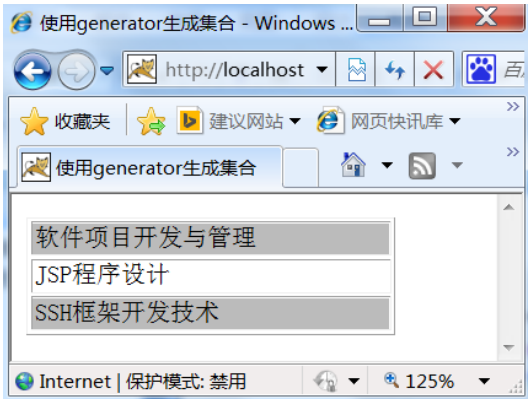

图 3-10　代码运行结果

图 3-10 显示了一个字符串中的数据（用“,”隔开）。该程序首先通过 generator 标签由该字符串生成一个集合，再通过 iterator 标签进行迭代显示。

Struts 2 控制标签主要完成表现层的输出流程控制，如条件控制、循环控制等操作，同时也可以完成对集合的合并、排序等操作。通过案例 3-3 介绍了 Struts 2 的 if、iterator、append、generator 等控制标签的使用。限于篇幅，其他控制标签的介绍略，读者可以参考其他文献。

3.1.5 其他非表单标签

还有一些非表单标签主要用于在页面中生成一些非表单的可视化的元素，例如 Tab 页面、树形结构等，主要包括以下几个。

- tabbedPanel：生成 HTML 页面的 Tab 页。
- tree：生成一个树形结构。
- treenode：生成树形结构的节点。
- a：生成一个超级链接。
- actionmessage：如果 Action 实例的 getActionMessage()方法不为 null，则该标签负责输出该方法的返回信息。

下列的代码是非表单标签 tree 的使用，它以“树”形结构显示作者及其作品，以供读者选择。

```
<body>
<h3>使用 s:tree 和 s:treenode 标签生成静态树</h3>
<s:tree label="选择课程教材" id="kcjc" theme="ajax"
      showRootGrid="true" showGrid="true" treeSelectedTopic="treeSelected">
    <s:treenode theme="ajax" label="刘晓林" id="lz">
        <s:treenode theme="ajax" label="JSP 程序开发" id="jsp"/>
        <s:treenode theme="ajax" label="J2EE 软件开发" id="j2ee"/>
        <s:treenode theme="ajax" label="J2ME 软件开发" id="j2me"/>
    </s:treenode>
    <s:treenode theme="ajax" label="牛德雄" id="myh">
        <s:treenode theme="ajax" label="软件项目开发与管理" id="rjgc"/>
    </s:treenode>
    <s:treenode theme="ajax" label="熊君丽" id="zqh">
        <s:treenode theme="ajax" label="SSH 框架软件开发技术" id="wljs"/>
    </s:treenode>
</s:tree>
</body>
```

上述代码使用了 tree 和 treenode 标签，运行结果如图 3-11 所示。

Struts 2 非表单标签还有 tabbedPanel（生成 HTML 页面的 Tab 页）、a（生成一个超级链接）、actionmessage 等，由于篇幅所限，这些非表单标签就不一一介绍了，请读者参阅其他相关文献。

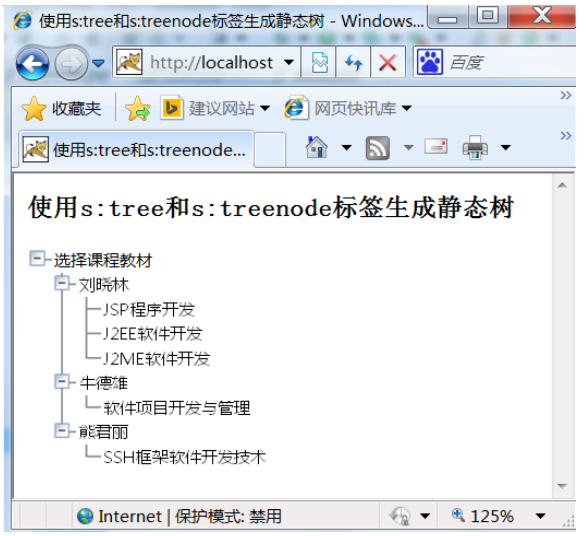

图 3-11　tree 标签的运行结果

[redacted]问数据对象

在 Struts 2 框架中，StackContext 里存放了 Struts 2 所有上下文的信息，这些信息提供给程序员，以进行操作实现数据的交互功能。这些信息往往以数据对象的形式存放。

OGNL（Object Graphic Navigation Language，对象图导航语言）是一种功能强大的表达式语言，通过它可以用简单的表达式访问数据对象中的属性值。这些数据对象可以是一般普通的数据对象，也可以是 Action 中的属性。

OGNL 是 Struts 2 框架默认的表达式语言（例如，在前面的 Struts 2 框架的 2.0.6 版本中，它是由 ognl-2.6.11.jar 包提供的支持）。通过 OGNL 增强了 Struts 2 的数据访问能力，大大简化了代码的编写，从而能大大提高程序开发效率。

3.2.1　OGNL 概述

在用 Struts 2 框架进行 MVC 模式的软件开发时，常常需要处理数据对象，例如可以操作一个普通的实体类实例化的数据对象，也可以操作 Action 属性的值。如果用 Java 代码实现对这些数据的操作会很烦琐，OGNL 提供了一套简便的表达式访问这些数据对象，从而能大大提高软件的开发效率。

前面的 Struts 2 案例已经讲过，如果要访问数据对象，可以在该数据对象前加前缀“#”，这就是 OGNL 的应用。另外，Struts 2 还在原有的 OGNL 基础上增加了对 ValueStack（值栈）对象的支持。

在传统的 OGNL 表达式求值中，系统会假设只有一个“根”对象，但 Struts 2 的 Stack Context 需要多个“根”对象，其中 ValueStack 只是多个“根”对象的其中之一。在 Struts 2 中，ValueStack 值栈是建立在 OGNL 基础之上的。系统的 ValueStack 是 OGNL 表达式的第一个根对象，如果最近有 Action 存在，则 Action 的上下文（ActionContext）是第二个根对象。OGNL 对这些对

象中的数据操作时，不需要在对象前加“#”前缀。

例如，在 Action 中有一个属性类型是对象 Student(name,age)，数据对象变量名是“student”，那么访问属性 sname 的 OGNL 表达式为：student.sname，即不需要在其前面加“#”前缀。

其实，当系统创建了 Struts 2 的 Action 实例后，该 Action 实例已经被保存到 ValueStack 中，所以用户可直接引用，无须加“#”，这就是因为有 OGNL 提供了直接访问 ValueStack 中数据对象的支持。访问其他 Context 中的对象时（如 parameters 对象、request 对象、session 对象、application 对象和 attr 对象），由于不是根对象，在访问时要加前缀“#”。

这些对象包括以下一些。

- parameters 对象，用于访问 HTTP 的请求参数。如果访问保存在其中的参数 username，则用 #parameters.username 或 #parameters['username']，相当于调用 Servlet 的 request.getParameter("username")。
- request 对象，用于访问 HttpServletRequest 的属性。如果访问保存在其中的参数 username，则用#request.username 或#request['username']，相当于调用 HttpServletRequest 的属性 request.getAttribute("username")。
- session 对象，用于访问 HttpSession。如果访问保存在其中的参数 username，则用 #session.username 或#session['username']，相当于调用 session.getAttribute("username")。
- application 对象，用于访问 ServletContext。如果访问保存在其中的参数 username，则用 #application.username 或 #application['username']，相当于调用 Servlet 的 getAttribute("username")。
- attr 对象，用于按 page→request→session→application 的顺序访问其属性，例如 #attr.username。

注意：Action 实例中的属性都需要创建 setter/getter 方法，以利于 OGNL 访问。如果该属性是对象类型，也同样需要添加 setter/getter 方法。

3.2.2 用 OGNL 操作数据对象

其实，在本书第 2 章的案例 2-1 代码中已经出现了 ActionContext（Action 上下文对象）的使用，即通过该对象存取一些与 Action 相关的信息。其实，这些信息的处理包括 Request、Session、Application、Locale、ValueStack 等，其中 ValueStack 可以解析 OGNL 表达式，从而动态获取一些值，同时可以给表达式提供数据对象。

其实，ActionContext 是被存放在当前线程中的，在执行拦截器、action 和 result 的过程中，由于它们都是在一个线程中按照顺序执行的，所以可以在任意时候获取并操作 ActionContext 中的数据。

OGNL 既能方便地访问存放在 ValueStack 根对象中 Action 属性的数据，也能方便地访问一般普通数据对象中的数据。下面通过案例演示介绍 OGNL 表达式如何操作这些数据对象中的数据。

【案例 3-4】 用 OGNL 表达式分别访问一般数据对象和 Action 属性中的数据。

本案例从不同的角度提供对数据对象的访问，包括数据标签、EL 表达式、OGNL 等。

首先，创建一个封装了如下数据的实体类 beans/Student.java，其部分代码如下：

```
public class Student {
        private int sid=1;
```

```
    private String sname="张国强";
    private String ssex="男";
    private String sgrade="13 软件 1 班";
    private int sage=21;
    private float sscore=89;
    //setter/getter 方法，省略
}
```

该类封装了表 3-1 所示的一个学生数据（下面简称“1 号学生”）。

表 3-1　1 号学生数据

学　号	姓　名	性　别	班　级	年　龄	分　数
1	张国强	男	13 软件 1 班	21	89

当该类创建了一个对象时，则该对象就默认含有 1 号学生的信息。

其次，定义一个 Action 控制器 control/ShowStudentAction.java，该 Action 的属性也是学生的信息，并且也赋了初值。该 Action 的部分代码如下。

```
public class ShowStudentAction implements Action{
    private int id=2;
      //1. 封装一个学生的信息
    private String name="李国清";
      private String sex="女";
      private String grade="13 软件 2 班";
      private int age=20;
      private float score=86;
      //setter/getter 方法，省略
      //2. 定义一个对象类型的属性
          Student student =new Student();

          public Student getStudent() {
                return student;
          }
          public void setStudent(Student student) {
                this.student = student;
          }
      //3. 编写 Action 的 execute()方法，其作用只有两个，一是在 ActionContext 中定义一个变量并
      赋值，二是直接跳转到 struts.xml 中定义的“success”指定的页面
      public String execute() throws Exception
          {
          ActionContext.getContext().getSession().put("user","黄国强");
          return "success";
         }
}
```

该 Action 控制器包括 3 个部分，首先通过属性驱动封装了表 3-2 所示学生的数据（下面简称“2 号学生”）。

表 3-2　2 号学生数据

学　号	姓　名	性　别	班　级	年　龄	分　数
2	李国清	女	13 软件 2 班	20	86

再次，定义一个 Student 类类型的属性及其 setter/getter 方法（注意，该类被实例化后初始化的数据为 1 号学生）。

最后，重写 Action 的 execute()方法，其内容有以下两个。

（1）在 ActionContext 中定义一个变量并赋值（后面介绍其作用）。

（2）直接跳转到 struts.xml 中定义的“success”指定的 JSP 页面(studentinfo.jsp)。

```
<?xml version="1.0" encoding="GBK"?>
<!DOCTYPE struts PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
        "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
    <package name="control" extends="struts-default">
        <action name="ShowStudent" class="control.ShowStudentAction">
            <result name="error">/index.jsp</result>
            <result name="success">/studentinfo.jsp</result>
        </action>
    </package>
</struts>
```

下面通过对这些数据对象的访问展示 OGNL 的功能。这些操作编码均在 JSP 文件（studentinfo.jsp）中。

1. 用数据标签访问数据对象

通过数据标签 bean、property 实例化并显示数据对象中的 1 号学生的信息，其代码如下。

```
显示数据对象中的学生信息：<br>
    <s:bean name ="beans.Student" id ="p">
    </s:bean>
        学号:<s:property value="#p.sid"/><br>
        姓名:<s:property value="#p.sname"/><br>
        性别:<s:property value="#p.ssex"/><br>
        年龄:<s:property value="#p.sage"/><br>
        班级:<s:property value='#p.sgrade'/><br>
        成绩:<s:property value="#p.sscore"/><br>
  <br>
```

上述代码显示的结果如图 3-12 所示，这里先通过 Student 类实例化一个对象“p”，然后通过“#p”访问其中的属性（1 号学生信息）。

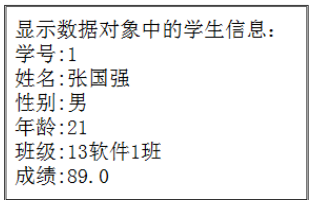

图 3-12　显示 Student 类中 1 号学生信息

该操作说明通过 OGNL 访问的不是 Struts 2 的 Stack Context 的根对象（如 ValueStack 对象与 ActionContext 对象），而是其他 Context 中的对象，这时需要在对象名前加“#” 加以区别（如访问姓名时用"#p.sname"）。

如果数据对象在不同的范围访问，则可以通过指定数据对象的范围进行。如下指定数据对象访问范围代码，其运行的结果也与图 3-12 所示类似。

```
通过指定数据对象所在的范围(attr)查找：    <br>
        学号:<s:property value="#attr.p.sid"/><br>
        姓名:<s:property value="#attr.p.sname"/><br>
        性别:<s:property value="#attr.p.ssex"/><br>
        年龄:<s:property value="#attr.p.sage"/><br>
        班级:<s:property value='#attr.p.sgrade'/><br>
        成绩:<s:property value="#attr.p.sscore"/><br>
    <br>
```

该代码的运行结果如图 3-13 所示。

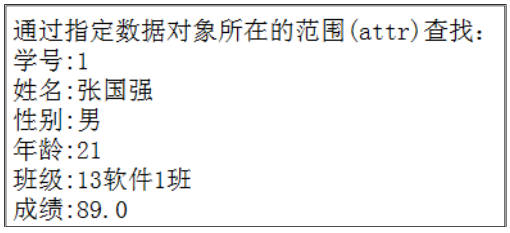

图 3-13　显示 Student 类中 1 号学生信息

数据对象的访问范围有 request 对象、session 对象、 application 对象和 attr 对象等。如果指定的是 attr 对象，则按 page→request→session→application 的顺序访问其属性。

2. 访问 Action 属性中的数据

直接通过 property 标签访问 Action 中的属性数据，其代码如下。

```
显示 Action 属性中的学生信息：<br>
    学号:<s:property value="id"/><br>
    姓名:<s:property value="name"/><br>
    性别:<s:property value="sex"/><br>
    年龄:<s:property value="age"/><br>
    班级:<s:property value='grade'/><br>
    成绩:<s:property value="score"/><br>
```

该代码运行的结果如图 3-14 所示。

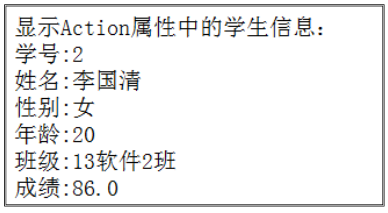

图 3-14　显示 Action 中 2 号学生信息