第 3 章 Tag文件与Tag标记

本章导读

主要内容

- Tag 文件的结构
- Tag 标记
- Tag 文件中的常用指令

难点

- Tag 文件中的 attribute 指令
- Tag 文件中的 variable 指令

关键实践

- 解析单词

一个 Web 应用中的许多 JSP 页面可能需要使用某些相同的信息，如都需要使用相同的导航栏、标题等。如果能将许多页面都需要的共同的信息形成一种特殊文件，而且各个 JSP 页面都可以使用这种特殊的文件，那么这样的特殊文件就是可复用的代码。代码复用是软件设计的一个重要方面、是衡量软件可维护性的重要指标之一。

第 2 章学习了 include 指令标记和 include 动作标记，使用这两个标记可以实现代码的复用。但是，在某些情况下，使用 include 指令标记和 include 动作标记有一定的缺点，比如，如果 include 指令标记或动作标记要处理的文件是一个 JSP 文件，那么用户可以在浏览器的地址栏中直接输入该 JSP 文件所在 Web 服务目录访问这个 JSP 文件，这可能不是 Web 应用所希望发生的，因为该 JSP 文件也许仅仅是个导航条，仅仅供其他 JSP 文件使用 include 指令标记或动作标记来嵌入或动态加载的，而不是让用户直接访问的。另外，include 指令标记和 include 动作标记允许所要处理的文件存放在 Web 服务目录中的任意子目录中，不仅显得杂乱无章，而且使得 include 标记和所处理文件的所在目录的结构形成了耦合，不利于 Web 应用的维护。

本章我们将学习一种特殊的文本文件：Tag 文件。Tag 文件和 JSP 文件很类似，可以被 JSP 页面动态加载调用，实现代码的复用（但用户不能通过该 Tag 文件所在 Web 服务目录直接访问 Tag 文件）。

本章在 webapps 目录下新建一个 Web 服务目录 ch3，除非特别约定，例子中的 JSP 页面均保存在 ch3 目录中。

3.1 Tag 文件

视频讲解

▶ 3.1.1 Tag 文件的结构

Tag 文件是扩展名为.tag 的文本文件，其结构和 JSP 文件类似。一个 Tag 文件中可以有普通的 HTML 标记符、某些特殊的指令标记（见 3.4 节）、成员变量声明和方法的定义、Java

程序片和 Java 表达式。以下是一个简单的 Tag 文件 oddNumberSum.tag，负责计算 100 内的全部奇数的代数和。

oddNumberSum.tag

```
<%@ tag pageEncoding="utf-8" %>
<p style="font-family:宋体;font-size:36">
1～100 内的奇数之和:
<%   int sum=0,i=1;
          for(i=1;i<=100;i++){
             if(i%2==1)
                sum=sum+i;
         }
         out.println(sum);
 %>
</p>
```

▶ 3.1.2 Tag 文件的保存

❶ Tag 文件所在目录

Tag 文件可以实现代码的复用，即 Tag 文件可以被许多 JSP 页面使用。为了能让一个 Web 应用中的 JSP 页面使用某一个 Tag 文件，必须把这个 Tag 文件存放到 Tomcat 服务器指定的目录中，也就是说，如果某个 Web 服务目录下的 JSP 页面准备调用一个 Tag 文件，那么必须在该 Web 服务目录下，建立如下的目录结构：

```
Web 服务目录\WEB-INF\tags
```

例如：

```
ch3\WEB-INF\tags
```

其中的 WEB-INF(字母大写)和 tags 都是固定的目录名称，而 tags 下的子目录的名称可由用户给定。

一个 Tag 文件必须保存到 tags 目录或其下的子目录中。这里把 3.1.1 节中的 oddNumberSum.tag 保存到 ch3\WEB-INF\tags 目录中。

❷ Tag 文件的编码

保存 Tag 文件时按照 Tag 文件指定的编码保存，例如 Tag 文件使用 tag 指令(见稍后的 3.4 节)：

```
<%@ tag pageEncoding="utf-8" %>
```

指定的编码是 UTF-8，因此需要按照 UTF-8 编码保存 Tag 文件。例如，用文本编辑器“记事本”编辑 Tag 文件，在保存该 Tag 文件时，将“保存类型(T)”选择为“所有文件(*.*)”，将“编码(E)”选择为“UTF-8”。

3.2 Tag 标记

▶ 3.2.1 Tag 标记与 Tag 文件


视频讲解

某个 Web 服务目录下的 Tag 文件只能由该 Web 服务目录中的 JSP 页面调用，JSP 页面必须通过 Tag 标记来调用一个 Tag 文件。

Tag 标记的名字和 Tag 文件的名字一致，也就是说，当我们编写了一个 Tag 文件并保存到特定目录中后（见 3.1.2 节），也就给出了一个 Tag 标记，该标记的格式为：

```
<Tag 文件的名字 />
```

或

```
<Tag 文件的名字 > 其他内容(称为标体内容)</Tag 文件的名字>
```

一个 Tag 文件对应着一个 Tag 标记，把全体 Tag 标记称之为一个自定义标记库或简称为标记库。

▶ 3.2.2 Tag 标记的使用

一个 JSP 页面通过使用 Tag 标记来调用一个 Tag 文件。Web 服务目录下的一个 JSP 页面在使用 Tag 标记来调用一个 Tag 文件之前，必须首先使用 taglib 指令标记引入该 Web 服务目录下的标记库，只有这样，JSP 页面才可以使用 Tag 标记调用相应的 Tag 文件。

taglib 指令的格式如下：

```
<%@ taglib tagdir = "标记库的位置" prefix = "前缀">
```

例如：

```
<%@ taglib tagdir = "/WEB - INF/tags" prefix = "computer" %>
```

引入标记库后，JSP 页面就可以使用带前缀的 Tag 标记调用相应的 Tag 文件，其中的前缀由< taglib >指令中的 prefix 属性指定。例如 JSP 如下使用 Tag 标记调用相应的 Tag 文件：

```
<computer:oddNumberSum />
```

taglib 指令中的 prefix 给出的前缀由用户自定义，其好处是，通过前缀可以有效地区分不同标记库中具有相同名字的标记文件。

注：JSP 页面使用 Tag 标记时，冒号：的左右不要有空格。

例 3_1 中的 JSP 页面使用 Tag 标记调用 oddNumberSum.tag（该 Tag 文件见 3.1.1 节）计算 100 之内的奇数和。

例 3_1

example3_1.jsp（效果如图 3.1 所示）

```
<%@ page contentType = "text/html" %>
```

```
<%@ page pageEncoding = "utf - 8" %>
<%@ taglib tagdir = "/WEB - INF/tags" prefix = "computer" %>
<HTML><body bgcolor = cyan>
<h1>调用 Tag 文件计算 100 内奇数和: </h1>
<computer:oddNumberSum /><% -- 使用 Tag 标记 -- %>
</body></HTML>
```

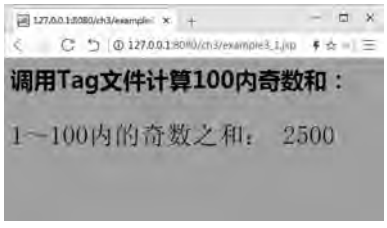

图 3.1 使用 Tag 标记

如果把 3.1.1 节中的 oddNumberSum.tag 保存在 ch3\WEB-INF\tags\example1 目录中,那么只要将上述 JSP 页面的 taglib 指令修改为:

```
<%@ taglib tagdir = "/WEB - INF/tags/example1" prefix = "computer" %>
```

即可。

▶ 3.2.3 Tag 标记的运行原理

Tomcat 服务器处理 JSP 页面中的 Tag 标记的原理如下:

- 如果该 Tag 标记对应的 Tag 文件是首次被 JSP 页面调用,那么 Tomcat 服务器会将 Tag 文件转译成一个 Java 文件,并编译这个 Java 文件生成字节码文件,然后执行这个字节码文件(这和执行 JSP 页面的原理类似)。
- 如果该 Tag 文件已经被转编译为字节码文件,Tomcat 服务器将直接执行这个字节码文件。
- 如果对 Tag 文件进行了修改,那么 Tomcat 服务器会重新将 Tag 文件转译成一个 Java 文件,并编译这个 Java 文件生成字节码文件,然后执行这个字节码文件。

3.3 Tag 文件中的常用指令


视频讲解

与 JSP 文件类似,Tag 文件中也有一些常用指令,这些指令将影响 Tag 文件的行为。Tag 文件中经常使用的指令有 tag、taglib、include、attribute、variable。

以下将分别讲述上述指令在 Tag 文件中的作用和用法。

▶ 3.3.1 tag 指令

Tag 文件中的 tag 指令类似于 JSP 文件中的 page 指令。Tag 文件通过使用 tag 指令可以指定某些属性的值,以便从总体上影响 Tag 文件的处理和表示。tag 指令的语法如下:

```
<%@ tag 属性 1 = "属性值" 属性 2 = "属性值" … 属性 n = "属性值" %>
```

在一个 Tag 文件中可以使用多个 tag 指令，因此我们经常使用多个 tag 指令为属性指定需要的值：

```
<%@ tag 属性 1 = "属性值" %>
<%@ tag 属性 2 = "属性值" %>
…
<%@ tag 属性 n = "属性值" %>
```

❶ language 属性

language 属性的值指定 Tag 文件使用的脚本语言，目前只能取值 Java，其默认值就是 Java，因此在编写 Tag 文件时，没有必要使用 tag 指令指定 language 属性的值。

❷ import 属性

import 属性的作用是为 Tag 文件引入包中的类，这样就可以在 Tag 文件的程序片部分、变量及方法定义部分、表达式部分使用包中的类。import 属性可以取多个值，import 属性默认已经有如下值："java. lang. *"　"javax. servlet. *"　"javax. servlet. jsp. *"　"javax. servlet. http. *"。

❸ pageEncoding

该属性指定 Tag 文件的字符编码，其默认值是 ISO-8859-1。目前，为了避免显示信息出现乱码现象，Tag 文件需要将该属性值设置为 UTF-8。

▶ 3.3.2　include 指令

在 Tag 文件中也有和 JSP 文件类似的 include 指令标记，其使用方法和作用与 JSP 文件中的 include 指令标记类似。

▶ 3.3.3　attribute 指令

Tag 文件充当着可复用代码的角色，如果一个 Tag 文件允许使用它的 JSP 页面向该 Tag 文件传递数据，就使得 Tag 文件的功能更为强大。在 Tag 文件中通过使用 attribute 指令让使用它的 JSP 页面向该 Tag 文件传递需要的数据。attribute 指令的格式如下：

```
<%@ attribute name = "对象名字" required = "true"|"false" type = "对象的类型" %>
```

例如 Tag 文件 myTag. tag 中有如下 attribute 指令：

```
<%@ attribute name = "result" required = "true" type = "java. lang. Double" %>
```

那么就相当于 Tag 文件中有了一个名字是 result 的对象，但 Tag 文件不需要创建该对象 result，而是等待 JSP 页面将一个 Double 型的对象的引用传递给 result。

attribute 指令中的 name 属性是必需的，该属性的值是一个对象的名字。JSP 页面在调用 Tag 文件时，可向 name 属性指定的对象传递一个引用。需要特别注意的是，type 在指定对象类型时，必须使用包名，比如，不可以将 java. lang. Double 简写为 Double。如果 attribute 指令中没有使用 type 指定对象的类型，那对象的类型默认是 java. lang. String 类型。

JSP 页面使用 Tag 标记向所调用的 Tag 文件中 name 指定的对象传递一个引用，方式如下：

```
<前缀：Tag 文件名字 对象名字 = "对象的引用" />
```

比如,JSP 页面使用 Tag 标记(假设标记的前缀为 computer)调用 myTag.tag:

```
<computer:myTag result = "new Double(3.1415926)" />
```

就向 myTag.tag 中 attribute 指令给出的对象 result 对象传递了一个 Double 对象的引用。

attribute 指令中的 required 属性也是可选的,如果省略 required 属性,那么 required 的默认值是 false。当指定 required 的值是 true 时,调用该 Tag 文件的 JSP 页面必须向该 Tag 文件中 attribute 指令中的 name 属性给出的对象传递一个引用,当指定 required 的值是 false 时,调用该 Tag 文件的 JSP 可以向该 Tag 文件中 attribute 指令中的 name 属性给出的对象传递或不传递对象的引用。

注:在 Tag 文件中不可以再定义和 attribute 指令中的 name 属性给出的对象具有相同名字的变量,否则将隐藏 attribute 指令中给出的对象,使其失效。

在下面的例 3_2 中,triangle.tag 存放在 ch3\WEB-INF\tags\example2 目录中,该 Tag 文件负责计算、显示三角形的面积。example3_2.jsp 页面保存在 ch3 中,使用 Tag 标记调用 triangle.tag 文件,并且向 triangle.tag 传递三角形三边的长度。

例 3_2

example3_2.jsp(效果如图 3.2 所示)

```
<%@ page contentType = "text/html" %>
<%@ page pageEncoding = "utf - 8" %>
<%@ taglib tagdir = "/WEB - INF/tags/example2" prefix = "getTriangleArea" %>
<HTML><body bgcolor = yellow >
<p style = "font - family:宋体;font - size:36;color:blue">
<% -- 使用 Tag 标记: -- %>
<getTriangleArea:triangle sideA = "15" sideB = "16" sideC = "20"/>
</p>
</body></HTML>
```

triangle.tag

```
<%@ tag pageEncoding = "utf - 8" %>
<%@ attribute name = "sideA" required = "true" %>
<%@ attribute name = "sideB" required = "true" %>
<%@ attribute name = "sideC" required = "true" %>
<%! public String getArea(double a,double b,double c) {
        if(a + b > c&&a + c > b&&c + b > a) {
            double p = (a + b + c)/2.0;
            double area = Math.sqrt(p * (p - a) * (p - b) * (p - c)) ;
            String result = String.format(" %.2f",area);
            return "<br>三角形面积(小数点保留 2 位):" + result;
         }
         else
            return("<br>" + a + "," + b + "," + c + "不能构成一个三角形,无法计算面积");
         }
 %>
<%  out.println("<BR>三边: " + sideA + "," + sideB + "," + sideC);
    double a = Double.parseDouble(sideA);
```

```
        double b = Double.parseDouble(sideB);
        double c = Double.parseDouble(sideC);
        out.println(getArea(a,b,c));
    %>
```

图 3.2　调用 Tag 文件计算面积

下面的例 3_3 中，JSP 页面 example3_3.jsp 只负责将一组随机数据存放到链表(java.util.LinkedList 类型对象)中，然后将链表传递给 sort.tag，sort.tag 负责按低到高顺序显示链表中的数据。sort.tag 存放在 ch3\WEB-INF\tags\example3 目录中，example3_3.jsp 保存在 ch3 目录中。

例 3_3

example3_3.jsp(效果如图 3.3 所示)

```
<%@ page contentType = "text/html" %>
<%@ page pageEncoding = "utf - 8" %>
<%@ page import = "java.util.LinkedList" %>
<%@ page import = "java.util.Random" %>
<%@ taglib tagdir = "/WEB - INF/tags/example3" prefix = "sortNumber" %>
<HTML><body bgcolor = #CCCCCC>
<%  LinkedList<Double> listNumber = new LinkedList<Double>();
    Random random = new Random();
    for(int i  = 0;i<3;i++) {
       double d = random.nextDouble();            //[0,1)之间的随机数
       listNumber.add(d);
    }
%>
<p style = "font - family:宋体;font - size:36;color:blue">
排序数据
<sortNumber:sort list = "<% = listNumber %>"/>  <% -- 使用 Tag 标记 -- %>
</body></HTML>
```

sort.tag

```
<%@ attribute name = "list" required = "true" type = "java.util.LinkedList" %>
<%@ tag import = "java.util.Collections" %>
<%@ tag import = "java.util.Iterator" %>
<%   Collections.sort(list);                        //排序链表
     Iterator<Double> ite = list.iterator();       //得到迭代器
     while(ite.hasNext()) {                        //遍历链表
        out.print("<br>" + ite.next());
     }
%>
```

图 3.3 调用 Tag 文件排序数据

▶ 3.3.4 variable 指令

Tag 文件通过使用 attribute 指令，可以使得调用该 Tag 文件的 JSP 页面动态地向其传递数据。在某些 Web 应用中，JSP 页面不仅希望向 Tag 文件传递数据，而且希望 Tag 文件能返回数据给 JSP 页面。比如，许多 JSP 页面可能都需要调用某个 Tag 文件对某些数据进行基本的处理，但不希望 Tag 文件做进一步的特殊处理以及显示数据，因为各个 JSP 页面对数据的进一步处理或显示格式的要求是不同的。因此，JSP 页面希望 Tag 文件将数据的基本处理结果存放在某些对象中，将这些对象返回给当前 JSP 页面即可。

Tag 文件通过使用 variable 指令可以将 Tag 文件中的对象返回给调用该 Tag 文件的 JSP 页面。

❶ variable 指令的格式

variable 指令的格式如下：

```
<%@ variable name-given="对象名" variable-class="对象类型" scope="有效范围"%>
```

variable 指令中属性 name-given 的值就是 Tag 文件返回给 JSP 页面的对象。该对象的名字必须符合标识符规定，即名字可以由字母、下画线、美元符号和数字组成，并且第一个字符不能是数字字符。variable 指令中属性 variable-class 的值是返回的对象的类型，对象的类型必须带有包名，比如 java.lang.Double、java.time.LocalDate 等类型。如果 variable 指令中没有使用 variable-class 给出对象的类型，那么对象的类型是 java.lang.String 类型。

variable 指令中 scope 属性的值指定对象的有效范围，scope 的值可以取 AT_BEGIN、NESTED 和 AT_END。当 scope 的值是 AT_BEGIN 时，JSP 页面一旦开始使用 Tag 标记，就得到了 variable 指令返回给 JSP 页面的对象，JSP 页面就可以在 Tag 标记的标记体中或 Tag 标记结束后的各个部分中使用 variable 指令返回给 JSP 页面的对象。当 scope 的值是 NESTED 时，JSP 页面只可以在 Tag 标记的标记体中使用 variable 指令返回给 JSP 页面的对象。当 scope 的值是 AT_END 时，JSP 页面只可以在 Tag 标记结束后。才可以使用 variable 指令返回给 JSP 页面的对象。

下面的 variable 指令给出的对象的名字是 time，类型为 java.time.LocalDate，有效范围是 AT_END：

```
<%@ variable name-given="time"
    variable-class="java.time.LocalDate" scope="AT_END"%>
```

❷ 对象的返回

Tag 文件为了给 JSP 页面返回一个对象，就必须将返回的对象的名字以及该对象的引用存储到 Tomcat 服务器提供的内置对象 jspContext 中。Tag 文件只有将对象的名字及其引用存储到 jspContext 中，JSP 页面才可以使用该对象。比如，Tag 文件的 variable 指令：

```
<%@ variable name-given="time"
    variable-class="java.time.LocalDate" scope="AT_END" %>
```

为 JSP 页面返回名字是 time 的 LocalDate 对象。那么 Tag 文件中必须让 jspContext 调用

```
setAttribute("对象名",对象的引用);
```

方法存储名字是 time 的对象以及该对象的引用,例如:

```
jspContext.setAttribute("time",LocalDate .now());
```

将名字是 time 的 LocalDate 对象存储到 jspContext 中。

下面的例 3_4 中,JSP 页面 example3_4.jsp 将 String 对象交给 Tag 文件 handleData.tag,handleData.tag 解析出 String 对象的字符序列中的全部数字,并计算出数字总和,将数字总和放在 Double 对象 price 中,然后返回给 JSP 页面 example3_4.jsp。example3_4.jsp 输出 price 对象中的数字总和。handleData.tag.tag 存放在 ch3\WEB-INF\tags\example4 目录中,example3_4.jsp 保存在 ch3 目录中。

例 3_4

example3_4.jsp(效果如图 3.4 所示)

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="utf-8" %>
<%@ taglib tagdir="/WEB-INF/tags/example4" prefix="getPrice" %>
<HTML><body bgcolor=#FFCCFF>
<% String str="麻辣豆腐:20.6 元,红烧肉:68.9 元,烤鸭:199 元";
 %>
<getPrice:handleData mess="<%= str %>"/>  <%-- 使用 Tag 标记 --%>
<p style="font-family:宋体;font-size:36">
菜单:<br>"<%= str %>"<br>价格总和:
<%= price %>          <%-- 使用 Tag 标记返回的 Double 对象 price --%>
</p>
<% str="毛巾:2.6 元,香皂:6.9 元,牙刷:12.3 元";
 %>
<getPrice:handleData mess="<%= str %>"/>
<p style="font-family:黑体;font-size:36;color:blue">
购物小票:<br>"<%= str %>"<br>价格总和:
<%= price %>
</p>
</body></HTML>
```

handleData.tag

```
<%@ attribute name="mess" required="true" type="java.lang.String" %>
<%@ tag import="java.util.regex.Pattern" %>
<%@ tag import="java.util.regex.Matcher" %>
<%@ variable name-given="price" variable-class="java.lang.Double"
  scope="AT_BEGIN" %>
<%!
public Double getPriceSum(String input){                    //定义方法
    Pattern pattern;                                        //模式对象
```

```
        Matcher matcher;                                          //匹配对象
        String regex = "-?[0-9][0-9]*[.]?[0-9]*";                 //匹配数字的正则表达式
        pattern = Pattern.compile(regex);                         //初始化模式对象
        matcher = pattern.matcher(input);                         //初始化匹配对象,用于检索 input
        double sum = 0;
        while(matcher.find()) {
           String str = matcher.group();
           sum += Double.parseDouble(str);
        }
        return new Double(sum);
    }
    %>
    <%  //将返回的 Double 对象放在 jspContext 中,用名字 price 返回给 JSP 页面
        jspContext.setAttribute("price",getPriceSum(mess));
    %>
```

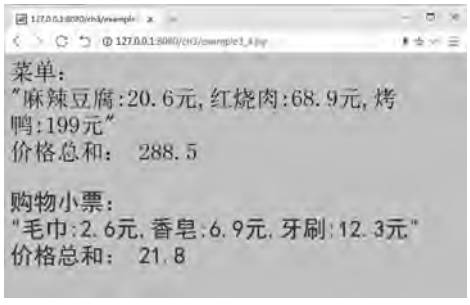

图 3.4 向 JSP 页面返回对象

注：在 JSP 页面中不可以再定义与 Tag 文件返回的对象具有相同名字的变量，否则 Tag 文件无法将 variable 指令给出的对象返回给 JSP 页面(并将出现编译错误)。如果 Tag 文件同时使用 variable 指令和 attribute 指令，那么 variable 指令中 name-given 和 attribute 指令中 name 给出的对象不能相同(否则将出现编译错误)。

▶ 3.3.5 taglib 指令

JSP 页面或 Tag 文件都可以使用 taglib 指令引入标记库(如前面各个例子所示)。taglib 指令格式如下：

```
<%@ taglib tagdir = "自定义标记库的位置" prefix = "前缀">
```

一个 Tag 文件也可以使用几个 taglib 指令标记引入若干个标记库，例如：

```
<%@ taglib tagdir = "/WEB-INF/tags" prefix = "beijing" %>
<%@ taglib tagdir = "/WEB-INF/tags/tagsTwo" prefix = "dalian" %>
```

3.4 上机实验


视频讲解

提供了详细的实验步骤要求，按步骤完成，提升学习效果，积累经验，不断提高 Web 设计能力。

▶ 3.4.1 实验 1 解析单词

❶ 实验目的

掌握 JSP 页面中使用 Tag 文件，向 Tag 文件传递对象，Tag 文件负责处理对象中的数据。

❷ 实验要求

(1) JSP 页面 giveText.jsp 负责将 String 对象，比如 String str ="how are you"，传递给所调用的 Tag 文件 backWords.tag。

(2) backWords.tag 文件负责解析出 String 对象中的单词，并将这些单词返回给 JSP 页面 giveText.jsp。

(3) JSP 页面 giveText.jsp 负责显示 backWords.tag 返回给它的单词。

(4) 在 Tomcat 服务器的 webapps 目录下(比如，D:\apache-tomcat-9.0.26\webapps)新建一个名字是 ch3_practice_one 的 Web 服务目录。把 giveText.jsp 文件保存到 ch3_practice_one 目录中。在 ch3_practice_one 目录下再建立目录结构：\WEB-INF\tags\ practice1，将 backWords.tag 保存在 practice1 目录中。

(5) 用浏览器访问 JSP 页面 giveText.jsp。

❸ 参考代码

参考代码运行效果如图 3.5 所示。

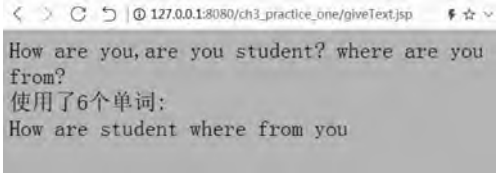

图 3.5 解析单词

giveText.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="utf-8" %>
<%@ page import="java.util.Iterator" %>
<%@ taglib tagdir="/WEB-INF/tags/practice1" prefix="getWords" %>
<HTML><body bgcolor=#CCCCFF>
<% String str="How are you,are you student? where are you from? ";
 %>
<getWords:backWords okString ="<% = str %>" /><% -- 使用 Tag 标记 -- %>
<p style="font-family:宋体;font-size:26">
<% = str %><br>
<%
      Iterator<String> ite=words.iterator();    //使用 Tag 标记返回的对象 words
      out.print("使用了"+ words.size()+"个单词:<br>");
      while(ite.hasNext()) {                    //遍历集合
         out.print(" "+ite.next());
      }
 %>
</p></body></HTML>
```

backWords.tag

```
<%@ tag import = "java.util.HashSet" %>
<%@ tag import = "java.util.regex.Pattern" %>
<%@ tag import = "java.util.regex.Matcher" %>
<%@ attribute name = "okString" required = "true" type = "java.lang.String" %>
<%@ variable name-given = "words" variable-class = "java.util.HashSet "
    scope = "AT_BEGIN" %>
<%
    HashSet<String>    set = new HashSet<String>();    //集合不允许有相同的元素
    Pattern pattern;                                   //模式对象
    Matcher matcher;                                   //匹配对象
    String regex = "[a-zA-Z]+" ;                       //匹配英文单词
    pattern = Pattern.compile(regex);                  //初始化模式对象
    matcher = pattern.matcher(okString);               //初始化匹配对象,用于检索 okString
    while(matcher.find()) {
       String str = matcher.group();
       set.add(str);
    }
   //将返回的 set 对象放在 jspContext 中,用名字 words 返回给 JSP 页面
   jspContext.setAttribute("words",set);
%>
```

▶ 3.4.2 实验2 显示日历

❶ 实验目的

和实验1的目的相同,进一步强化掌握JSP页面使用Tag文件,即向Tag文件传递对象,Tag文件负责处理数据。

❷ 实验要求

(1) Tag文件calendar.tag负责显示日历。

(2) 编写JSP页面useCalendar.jsp,要求useCalendar.jsp使用Tag标记使用calendar.tag,并将日期的年份和月份传递给calendar.tag。

(3) 在Tomcat服务器的webapps目录下(比如,D:\apache-tomcat-9.0.26\webapps)新建一个名字是ch3_practice_two的Web服务目录。把useCalendar.jsp文件保存到ch3_practice_two目录中。在ch3_practice_two目录下再建立目录结构:\WEB-INF\tags\ practice2,将calendar.tag保存在practice2目录中。

(4) 用浏览器访问JSP页面useCalendar.jsp。

❸ 参考代码

参考代码运行效果如图3.6所示。

图3.6 显示日历

useCalendar.jsp

```
<%@ page contentType="text/html" %>
<%@ page pageEncoding="utf-8" %>
<%@ taglib tagdir="/WEB-INF/tags/practice2" prefix="getCalendar" %>
<HTML><body bgcolor=#CCCCFF>
<getCalendar:calendar year ="2025" month="2" /><% -- 使用 Tag 标记 -- %>
</body></HTML>
```

calendar.tag

```
<%@ tag import="java.time.LocalDate" %>
<%@ tag import="java.time.DayOfWeek" %>
<%@ attribute name="year" required="true" type="java.lang.String" %>
<%@ attribute name="month" required="true" type="java.lang.String" %>
<%
    int y=Integer.parseInt(year);
    int m=Integer.parseInt(month);
    LocalDate date=LocalDate.of(y,m,1);
    int days=date.lengthOfMonth();                          //得到该月有多少天
    int space=0;                                            //存放空白字符的个数
    DayOfWeek dayOfWeek=date.getDayOfWeek();                //得到1号是星期几
    switch(dayOfWeek) {
          case SUNDAY:      space=0;
                            break;
          case MONDAY:      space=1;
                            break;
          case TUESDAY:     space=2;
                            break;
          case WEDNESDAY:   space=3;
                            break;
          case THURSDAY:    space=4;
                            break;
          case FRIDAY:      space=5;
                            break;
          case SATURDAY:    space=6;
                            break;
   }
   String[] calendar=new String[space+days];                //用于存放日期和1号前面的空白
   for(int i=0;i<space;i++)
       calendar[i]="--";
   for(int i=space,n=1;i<calendar.length;i++){
       calendar[i]=String.valueOf(n) ;
       n++;
   }
%>
<h3><%=year %>年<%=month %>月的日历:</h3>
<table border=0>
  <tr><th>星期日</th><th>星期一</th><th>星期二</th><th>星期三</th>
      <th>星期四</th><th>星期五</th><th>星期六</th>
  </tr>
<%
```

```
        int n = 0;
        while(n < calendar.length){
            out.print("< tr >");
            int increment = Math.min(7,calendar.length - n);
            for(int i = n;i < n + increment;i++) {
              out.print("< td >" + calendar[i] + "</td >");
            }
            out.print("</tr >");
            n = n + increment;
        }
    %>
    </table >
```

习 题 3

1. 用户可以使用浏览器直接访问一个 Tag 文件吗?

2. Tag 文件应当存放在怎样的目录中?

3. Tag 文件中的 tag 指令可以设置哪些属性的值?

4. Tag 文件中的 attribute 指令有怎样的作用?

5. Tag 文件中的 variable 指令有怎样的作用?

6. 编写两个 Tag 文件 Rect.tag 和 Circle.tag。Rect.tag 负责计算并显示矩形的面积,Circle.tag 负责计算并显示圆的面积。编写一个 JSP 页面 lianxi6.jsp,该 JSP 页面使用 Tag 标记调用 Rect.tag 和 Circle.tag。调用 Rect.tag 时,向其传递矩形的两个边的长度;调用 Circle.tag 时,向其传递圆的半径。

7. 编写一个 Tag 文件:GetArea.tag 负责求出三角形的面积,并使用 variable 指令返回三角形的面积给调用该 Tag 文件的 JSP 页面。JSP 页面负责显示 Tag 文件返回的三角形的面积。JSP 在调用 Tag 文件时,使用 attribute 指令将三角形三边的长度传递给 Tag 文件。one.jsp 和 two.jsp 都使用 Tag 标记调用 GetArea.tag。one.jsp 返回的三角形的面积保留最多 3 位小数,two.jsp 返回的三角形的面积保留最多 6 位小数。