# 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文件的存储目录
- Tag标记
- Tag文件中的常用指令
- Tag标记的嵌套

**难点**

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

**关键实践**

- 使用标记体
- 使用attribute指令和variable指令

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。

前面学习了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。例如，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服务目录来访问这个JSP文件，这可能不是设计者所希望发生的，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，但是用户不能通过该Tag文件所在Web服务目录直接访问这个Tag文件。

使用Tag文件具有以下重要的两点好处：

(1) 在设计 Web 应用时,可以通过编写 Tab 文件实现代码复用。

(2) 可将 JSP 页面中的关于数据处理的代码放在一个 Tag 文件中,让 JSP 页面只负责显示数据,即通过使用 Tag 文件将数据的处理和显示相分离,有利于 Web 应用的维护。

Tomcat 服务器的 webapps 目录的子目录都可以作为一个 Web 服务目录,在 webapps 目录下新建一个 Web 服务目录 ch3,除非特别约定,本章例子中的 JSP 页面均保存在 ch3 中。

## 3.1 Tag 文件的结构

Tag 文件是扩展名为.tag 的文本文件,其结构几乎和 JSP 文件相同。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 3.3 节)、成员变量和方法的声明、Java 程序片和 Java 表达式。

为了便于在名字上明显地区分 Tag 文件和 JSP 文件,本章中的 Tag 文件的名字的首写字母为大写,JSP 文件的名字的首写字母为小写。以下是两个简单的 Tab 文件:AddSum.tag 和 EvenSum.tag。AddSum.tag 负责计算 1～100 内的全部奇数之和;EvenSum.tag 负责计算 1～100 内的全部偶数之和。

**AddSum.tag**

```
<P>这是一个 Tab 文件,负责计算 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);
  %>
```

**EvenSum.tag**

```
<P>这是一个 Tab 文件,负责计算 1～100 内的偶数之和:
  <% int sum = 0,i = 1;
       for(i = 1; i<= 100; i++ ){
          if(i % 2 == 0)
            sum = sum + i;
       }
       out.println(sum);
  %>
```

## 3.2 Tag 文件的存储目录

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 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。把 3.1 节中的 AddSum.tag 保存到

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

目录中，将 EvenSum.tag 保存到

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

目录中。

**注意**：Tag 文件必须使用 ANSI 编码保存。

# 3.3 Tag 标记

## 3.3.1 Tag 标记与 Tag 文件

某个 Web 服务目录下的 Tag 文件只能由该 Web 服务目录(包括该 Web 服务目录的子目录)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 标记是伴随着 Tag 文件一同诞生的，即编写了一个 Tag 文件并保存到特定目录中后，也就自定义出了一个标记，该标记的格式为：

```
<Tag 文件名字 />
```

或

```
<Tag 文件名字 >
    标记体
</ Tag 文件名字>
```

一个 Tag 文件对应着一个标记，该标记被习惯地称为 Tag 标记，将存放在同一目录中的若干个 Tag 文件所对应的 Tag 标记的全体称为一个自定义标记库或简称为标记库。

## 3.3.2 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 = "自定义标记库的位置" prefix = "前缀">
```

一个 JSP 页面可以使用几个 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/geng" prefix = "dalian" %>
```

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:

```
<beijing: AddSum/>
<dalian: EvenSum/>
```

**注意**: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。

JSP 引擎处理 JSP 页面中的 Tag 标记的原理如下:

(1) 如果该 Tag 标记对应的 Tag 文件是首次被 JSP 页面调用,那么 JSP 引擎会将 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来实现 Tag 文件的动态处理,将有关的结果发送到用户端(这和执行 JSP 页面的原理类似)。

(2) 如果该 Tag 文件已经被编译为字节码文件,那么 JSP 引擎将直接执行这个字节码文件来实现 Tag 文件的动态处理。

(3) 如果对 Tag 文件进行了修改,那么 JSP 引擎会重新将 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来实现 Tag 文件的动态处理。

现在用一个例子来说明怎样在 JSP 页面中调用 Tag 文件。例 3-1 中的 JSP 页面保存在 Web 服务目录 ch3 中,该 JSP 页面所调用的 Tag 文件是 3.1 节中提到的 AddSum.tag 和 EvenSum.tag。example3_1.jsp 的效果如图 3-1 所示。

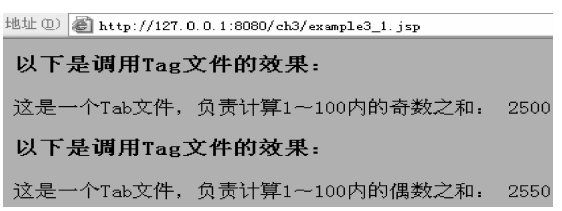



图 3-1 在 JSP 页面中使用 Tag 标记

**例 3-1**

**example3_1.jsp**

```
<%@ page contentType = "text/html; Charset = GB2312" %>
<%@ taglib tagdir = "/WEB - INF/tags" prefix = "beijing" %>
<%@ taglib tagdir = "/WEB - INF/tags/geng" prefix = "dalian" %>
<html><body bgcolor = cyan>
  <h3>以下是调用 Tag 文件的效果:</h3>
      <beijing: AddSum />
  <h3>以下是调用 Tag 文件的效果:</h3>
      <dalian: EvenSum />
</body></html>
```

**注意**:Tag 文件在“<%!”和“%>”标记符号之间声明的变量,和 JSP 在“<%!”和“%>”之间声明的变量类似,其有效范围是整个 Tag 文件。但是有一点非常不同,每当 Tag 文件

对应的字节码被执行完毕后,这些变量即可释放所占有的内存空间。

### 3.3.3 Tag 标记的标记体



一个 Tag 文件会对应一个 Tag 标记,并让 JSP 页面使用这个 Tag 标记动态执行该 Tag 文件。Tag 标记的格式为:

```
<Tag 文件名字 />
```

或

```
<Tag 文件名字 >
      标记体
</ Tag 文件名字>
```

默认情况下,JSP 可以使用没有标记体的 Tag 标记,也可以使用带有标记体的 Tag 标记调用一个 Tag 文件。那么,Tag 标记的标记体起着怎样的作用呢?

当 JSP 页面调用一个 Tag 文件时可能希望动态地向该 Tag 文件传递信息,那么就可以使用带有标记体的 Tag 标记来执行一个 Tag 文件,Tag 标记中的“标记体”就会传递给相应的 Tag 文件,这个 Tag 文件通过使用

```
<jsp: doBody />
```

标记处理 JSP 页面传递过来的“标记体”。

那么,标记体可以是怎样的形式呢? 默认的情况下,标记体可以是一些文本数据,有关细节会在后续的 3.4 节中详细讨论。

在例 3-2 中,example3_2.jsp 页面调用 Show.tag 文件,而且通过标记体向 Show.tag 文件传递文本数据。Show.tag 文件使用<jsp: doBody/>处理 example3_2.jsp 页面传递过来的文本数据,将该文本数据循环显示 3 次并逐次增大文本字体的字号。Show.tag 保存在 ch3\WEB-INF\tags 目录中,example3_2.jsp 的效果如图 3-2 所示。

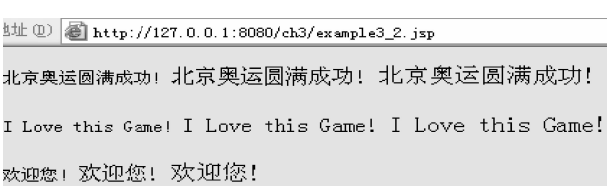



图 3-2 Tag 标记的标记体

**例 3-2**

**example3_2.jsp**

```
<%@ page contentType = "text/html; Charset = GB2312" %>
<%@ taglib prefix = "look" tagdir = "/WEB - INF/tags" %>
<HTML>
  <look: Show>
      北京奥运圆满成功!                    <%-- 标记体 --%>
  </look: Show>
  <look: Show>
```

```
        I Love this Game!                    <%-- 标记体 --%>
    </look:Show>
    <look:Show>
        欢迎您!                              <%-- 标记体 --%>
    </look:Show>
</HTML>
```

**Show.tag**

```
<body bgcolor=yellow><P>
<%   int size=1;
       for(int i=1; i<=3; i++){
           size=size+1;
 %>       <font size=<%=size%>>
              <jsp:doBody />
            </font>
<% }
 %>
</P></body>
```

## 3.4 Tag 文件中的常用指令

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、variable、include、attribute 和 taglib。

以下将分别讲解上述指令在 Tag 文件中的作用和用法。

### 3.4.1 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"%>
```

tag 指令可以操作的属性有 body-content、language、import 和 pageEncoding。以下将分别讲解怎样设置这些属性的值。

**1. body-content 属性**

标记的格式为:

```
<Tag文件名字 />
```

或

```
<Tag 文件名字 >
      标记体
</ Tag 文件名字>
```



JSP 文件通过使用 Tag 标记调用相应的 Tag 文件,那么 JSP 文件到底应该使用 Tag 标记的哪种格式来调用 Tag 文件呢? 答案是:一个 Tag 文件通过 tag 指令指定 body-content 属性的值可以决定 Tag 标记的使用格式。也就是说,body-content 属性的值可以确定 JSP 页面使用 Tag 标记时是否可以有标记体,如果允许有标记体,该属性会给出标记体内容的类型。

body-content 属性的值有 empty、tagdependent、scriptless,默认值是 scriptless。

如果 body-content 属性的值是 empty,那么 JSP 页面必须使用没有标记体的 Tag 标记:

```
<Tag 文件名字 />
```

来调用相应的 Tag 文件。

如果 body-content 属性的值是 tagdependent 或 scriptless,那么 JSP 页面可以使用无标记体或有标记体的 Tag 标记:

```
<Tag 文件名字 >
      标记体
</ Tag 文件名字>
```

来调用相应的 Tag 文件。如果属性值是 scriptless,那么标记体中不能有 Java 程序片;如果属性值是 tagdependent,那么 Tag 文件将标记体的内容按纯文本处理。

Tag 标记中的标记体由相应的 Tag 文件负责处理,因此,当 JSP 页面使用有标记体的 Tag 标记调用一个 Tag 文件时,可以通过“标记体”向该 Tag 文件动态地传递文本数据或必要的 JSP 指令。

Tag 文件通过使用标记:

```
<jsp: doBody />
```

来获得 JSP 页面传递过来的“标记体”,并进行处理,也就是说,在一个 Tag 文件中,<jsp: doBody />标记被替换成对“标记体”进行处理后所得到的结果,有关<jsp: doBody/>的用法可参见例 3-2。

**2. 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。

**3. import 属性**

import 属性的作用是为 Tag 文件引入 Java 核心包中的类,这样就可以在 Tag 文件的程序片部分、变量及方法声明部分、表达式部分使用 Java 核心包中的类。import 属性可以取多个值,import 属性已经有如下值:java.lang.*、javax.servlet.*、javax.servlet.jsp.* 和 javax.servlet.http.*。

**4. pageEncoding**

该属性的值指定 Tag 文件的字符编码,其默认值是 ISO-8859-1。

### 3.4.2 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。

### 3.4.3 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，可以让使用它的 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" %>
```

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。type 指定对象的类型。例如：type=" java.util.Date"。需要特别注意的是，对象的类型必须带有包名。例如，不可以将 java.util.Date 简写为 Date。如果 attribute 指令中没有使用 type 指定对象的类型，那对象的类型是 java.lang.String 类型。

JSP 页面使用 Tag 标记向所调用的 Tag 文件中 name 属性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" />
```

或

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

例如，一个 Tag 文件：MyTag.tag 中有如下的 attribute 指令：

```
<%@ attribute name="length" required="true" %>
```

那么 JSP 页面就可以如下使用 Tag 标记(假设标记的前缀为 computer)调用 MyTag.tag：

```
<computer：MyTag length="1000" />
```

或

```
<computer：MyTag length="1000" >
      我向 Tag 文件中传递的值是 1000
<computer：MyTag />
```

再举一例，一个 Tag 文件 YourTag.tag 中已有如下的 attribute 指令：

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

那么 JSP 页面可以使用 Tag 标记(假设标记的前缀为 computer)调用 YourTag.tag，将一个 java.lang.Double 类型对象的引用传递给 YourTag.tag 文件中的 result 对象：

```
<computer：YourTag result="<% = new Double(666.999) %>" />
```

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 属性指定的对象传递一个引用,即当 required 的值是 true 时,如果使用"<前缀: Tag 文件名字 />"调用 Tag 文件就会出现错误。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 属性指定的对象传递或不传递对象的引用。

**注意**:在 Tag 文件中不可以再定义和 attribute 指令中的 name 属性指定的对象具有相同名字的变量,否则将隐藏 attribute 指令中的对象,使其失效。

在例 3-3 中,Triangle. tag 存放在 ch3\WEB-INF\tags 目录中,该 Tag 文件负责计算、显示三角形的面积。example3_3. jsp 使用 Tag 标记调用 Triangle. tag 文件,并且向 Triangle. tag 文件传递三角形三边的长度。example3_3. jsp 的效果如图 3-3 所示。

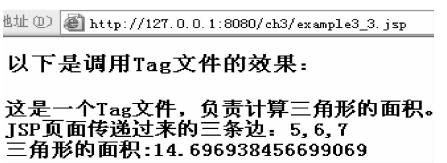



图 3-3 调用 Tag 文件计算面积

**例 3-3**

**example3_3. jsp**

```
<%@ page contentType = "text/html; Charset = GB2312" %>
<%@ taglib tagdir = "/WEB - INF/tags" prefix = "computer" %>
<HTML><BODY>
   <H3>以下是调用 Tag 文件的效果:</H3>
   <computer: Triangle sideA = "5" sideB = "6" sideC = "7"/>
</BODY></HTML>
```

**Triangle. tag**

```
<h4>这是一个 Tag 文件,负责计算三角形的面积。
<%@ 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));
               return "<BR>三角形的面积: " + area;
            }
            else
               return("<BR>" + a + "," + b + "," + c + "不能构成一个三角形,无法计算面积。");
        }
  %>
  <%   out.println("<BR>JSP 页面传递过来的三条边: " + sideA + "," + sideB + "," + sideC");
       double a = Double.parseDouble(sideA);
       double b = Double.parseDouble(sideB);
       double c = Double.parseDouble(sideC);
       out.println(getArea(a,b,c));
  %>
```

在例 3-4 中,JSP 页面只负责将学生的姓名和成绩分别存放到链表(java.util.LinkedLst 类型对象)中,然后将链表传递给 Sort.tag,Sort.tag 负责按从低到高的顺序显示学生的成绩。example3_4.jsp 的效果如图 3-4 所示。

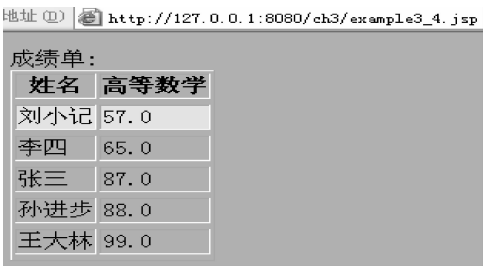



图 3-4　调用 Tag 文件排序成绩

**例 3-4**

**example3_4.jsp**

```
<%@ page contentType="text/html; Charset=GB2312" %>
<%@ page import="java.util.*" %>
<%@ taglib tagdir="/WEB-INF/tags" prefix="show" %>
<HTML><BODY bgcolor=cyan>
   <%   LinkedList listName=new LinkedList();
         LinkedList listScore=new LinkedList();
         listName.add("张三");
         listScore.add(new Double(87));
         listName.add("李四");
         listScore.add(new Double(65));
         listName.add("刘小记");
         listScore.add(new Double(57));
         listName.add("王大林");
         listScore.add(new Double(99));
         listName.add("孙进步");
         listScore.add(new Double(88));
    %>
  <p>成绩单:
  <show:Sort title="姓名" item="高等数学"
              listName="<%=listName%>" listScore="<%=listScore%>"/>
 </BODY></HTML>
```

**Sort.tag**

```
<%@ attribute name="listName" required="true" type="java.util.LinkedList" %>
<%@ attribute name="listScore" required="true" type="java.util.LinkedList" %>
<%@ attribute name="title" required="true" %>
<%@ attribute name="item" required="true" %>
<% for(int i=0; i<listName.size(); i++){
       for(int j=i+1; j<listName.size(); j++){
          double a=((Double)listScore.get(i)).doubleValue();
          double b=((Double)listScore.get(j)).doubleValue();
          if(b<a){
```

```
                String temp = (String)listName.get(i);
                Double r = (Double)listScore.get(i);
                listName.set(i,(String)listName.get(j));
                listName.set(j,temp);
                listScore.set(i,(Double)listScore.get(j));
                listScore.set(j,r);
            }
        }
    }
    out.print("<table border = 1 >");
    out.print("<tr>");
        out.print("<th>" + title + "</th>");
        out.print("<th>" + item + "</th>");
    out.print("<tr>");
    for(int k = 0; k < listName.size(); k++ ){
        out.print("<tr>");
        double score = (Double)listScore.get(k);
        String name = (String)listName.get(k);
        if(score < 60){
            out.print("<td bgcolor = yellow>" + name + "</td>");
            out.print("<td bgcolor = yellow>" + score + "</td>");
        }
        else{
            out.print("<td>" + name + "</td>");
            out.print("<td>" + score + "</td>");
        }
        out.print("<tr>");
    }
    out.print("</table>");
%>
```

### 3.4.4 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 设计中,JSP 页面不仅希望向 Tag 文件传递数据,而且还希望 Tag 文件能返回某些数据给 JSP 页面。例如,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 页面对数据进行进一步处理的细节以及对数据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,并将这些对象返回给当前 JSP 页面,由 JSP 页面负责进一步处理和显示这些对象,这样做可以很好地实现代码的复用,将数据的基本处理和特殊数据以及显示相分离。

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。

**1. variable 指令的格式**

variable 指令的格式如下:

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

variable 指令中属性 name 的值用来指定对象的名字，名字必须符合标识符规定，即名字可以由字母、下划线、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 的值指定对象的类型。例如，可以是 java. lang. String、java. lang. Integer、java. lang. Float、java. lang. Double、java. util. Date 等类型。如果 variable 指令中没有使用 variable-class 指定对象的类型，那么对象的类型是 java. lang. String 类型。需要特别注意的是，对象的类型必须带有包名。例如，不可以将 java. util. Date 简写为 Date。

variable 指令中属性 scope 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，就可以使用 variable 指令中给出的对象。例如，JSP 页面可以在 Tag 标记的标记体中或 Tag 标记结束后的各个部分中使用 variable 指令给出的对象。当 scope 的值是 NESTED 时，JSP 页面只可以在 Tag 标记的标记体使用 variable 指令给出的对象。

当 scope 的值是 AT_END 时，JSP 页面只有在 Tag 标记结束后，才可以使用 variable 指令中给出的对象，也就是说 JSP 页面不可以在 Tag 标记的标记体中使用该 variable 指令给出的对象，必须在调用完 Tag 标记后才可以使用 variable 指令给出的对象。

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

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

**2. 对象的返回**

Tag 文件为了给 JSP 页面返回一个对象，就必须将对象的名字以及该对象的引用存储到 Tomcat 引擎提供的内置对象 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。

jspContext 调用

```
setAttribute("对象的名字",对象的引用);
```

方法存储对象的名字以及该对象的引用。例如：

```
jspContext.setAttribute("time",new Date());
```

将名字是 time 的 Date 对象存储到 jspContext 中。

以下的 variable 指令：

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

为 JSP 页面返回名字是 time 的 Date 对象。下列 Tag 文件 GiveDate. tag 使用了上述 variable 指令：

**GiveTag. tag**

```
<%@ tag import="java.util.*" %>
<%@ variable name-given="time" variable-class="java.util.Date" scope="AT_END" %>
  <%   jspContext.setAttribute("time",new Date());
   %>
```

需要特别注意的是，不能在 Tag 文件中的 Java 程序片中直接操作 variable 指令中的对

象,Tag 文件只能将对象的名字及其引用存储到 jspContext 中。例如,下列程序片中的代码是不允许的:

```
<%
    time = new Date();
 %>
```

以下的 JSP 页面使用 Tag 标记调用上述 GiveTag.tag 文件。在下面的 JSP 页面 lookTime.jsp 中,在出现 Tag 标记

```
<showTime: GiveDate />
```

之后,该 JSP 页面就可以在程序片中或表达式部分使用 GiveDate.tag 文件为它返回的 time 对象。以下是使用 GiveDate.tag 文件的 lookTime.jsp 的完整代码:

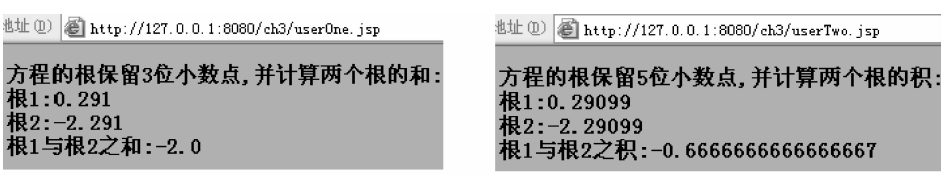
**lookTime.jsp**

```
<%@ page contentType = "text/html; Charset = GB2312" %>
<%@ page import = "java.text.*" %>
<%@ taglib tagdir = "/WEB-INF/tags" prefix = "showTime" %>
<HTML><BODY bgcolor = cyan>
    <showTime: GiveDate />        <%-- Tag 标记 --%>
 <h3>当前时间:</h3>
    <% = time %>                  <%-- 使用 GiveDate.tag 文件返回的 time 对象 --%>
</BODY></HTML>
```

**注意**: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(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 指令和 attribute 指令中的 name 指定的对象不能相同(将出现编译错误)。

在例 3-5 中,Tag 文件 GiveRoot.tag 负责求出一元二次方程的根。JSP 页面在调用 Tag 文件时,使用 attribute 指令将方程的系数传递给 Tag 文件;Tag 文件 GiveRoot.tag 使用 variable 指令返回一元二次方程的根给调用该 Tag 文件的 JSP 页面。例 3-5 中的 userOne.jsp 和 userTwo.jsp 都使用 Tag 标记调用 GiveRoot.tag,二者都可以得到 GiveRoot.tag 返回的方程的两个根,但是二者使用不同的方式来处理和显示方程的两个根。userOne.jsp 将方程的根保留最多 3 位小数,并计算方程的两个根之和,userTwo.jsp 将方程的根保留最多 5 位小数,并计算方程的两个根之积。userOne.jsp 和 userTwo.jsp 的效果如图 3-5 所示。




(a) userOne.jsp　　(b) userTwo.jsp

图 3-5　userOne.jsp 和 userTwo.jsp 的效果图

**例 3-5**

**userOne. jsp**

```
<%@ page contentType = "text/html; Charset = GB2312" %>
<%@ page import = "java.text.*" %>
<%@ taglib tagdir = "/WEB-INF/tags" prefix = "computer" %>
<HTML><BODY bgcolor = cyan>
  <computer:GiveRoot coefficientA = "3" coefficientB = "6" coefficientC = "-2"/>
  <h4>方程的根保留 3 位小数点,并计算两个根的和:
  <% NumberFormat f = NumberFormat.getInstance();
     f.setMaximumFractionDigits(3);
     if(rootOne!= null&&rootTwo!= null){
        double r1 = rootOne.doubleValue(); //rootOne 是 GetRoot.tag 文件返回的 Double 型对象
        double r2 = rootTwo.doubleValue(); //rootTwo 是 GetRoot.tag 文件返回的 Double 型对象
        String s1 = f.format(r1);
        String s2 = f.format(r2);
        out.println("<br>根 1:" + s1);
        out.println("<br>根 2:" + s2);
        double sum = r1 + r2;
        out.println("<br>根 1 与根 2 之和:" + sum);
     }
     else{
        out.println("<br>方程没有实根");
     }
  %>
</BODY></HTML>
```

**userTwo. jsp**

```
<%@ page contentType = "text/html; Charset = GB2312" %>
<%@ page import = "java.text.*" %>
<%@ taglib tagdir = "/WEB-INF/tags" prefix = "computer" %>
<HTML><BODY bgcolor = cyan>
<computer:GiveRoot coefficientA = "3" coefficientB = "6" coefficientC = "-2"/>
<h4>方程的根保留 5 位小数点,并计算两个根的积:
 <% NumberFormat f = NumberFormat.getInstance();
    f.setMaximumFractionDigits(5);
    if(rootOne!= null&&rootTwo!= null){
        double r1 = rootOne.doubleValue(); //rootOne 是 GetRoot.tag 文件返回的 Double 型对象
        double r2 = rootTwo.doubleValue(); //rootTwo 是 GetRoot.tag 文件返回的 Double 型对象
        String s1 = f.format(r1);
        String s2 = f.format(r2);
        out.println("<br>根 1:" + s1);
        out.println("<br>根 2:" + s2);
        double ji = r1 * r2;
        out.println("<br>根 1 与根 2 之积:" + ji);
    }
    else{
        out.println("<br>方程没有实根");
    }
  %>
</BODY></HTML>
```

**GiveRoot. tag**

```
<%@ tag import = "java.util. * " %>
<%@ attribute name = "coefficientA" required = "true" %>
<%@ attribute name = "coefficientB" required = "true" %>
<%@ attribute name = "coefficientC" required = "true" %>
<%@ variable name - given = "rootOne" variable - class = "java.lang.Double"
 scope = "AT_END" %>
<%@ variable name - given = "rootTwo" variable - class = "java.lang.Double" scope = "AT_END" %>
  <%   double disk,root1,root2;
        double a = Double.parseDouble(coefficientA);
        double b = Double.parseDouble(coefficientB);
        double c = Double.parseDouble(coefficientC);
        disk = b * b - 4 * a * c;
        if(disk >= 0&&a! = 0){
          root1 = ( - b + Math.sqrt(disk))/(2 * a);
          root2 = ( - b - Math.sqrt(disk))/(2 * a);
          jspContext.setAttribute("rootOne",new Double(root1)); //返回对象 rootOne
          jspContext.setAttribute("rootTwo",new Double(root2)); //返回对象 rootTwo
        }
        else{
          jspContext.setAttribute("rootOne",null);
          jspContext.setAttribute("rootTwo",null);
      }
  %>
```

### 3.4.5 taglib 指令

和 JSP 页面使用 Tag 文件一样,一个 Tag 文件也可以使用 Tag 标记来调用其他 Tag 文件。和 JSP 页面使用 Tag 标记的要求相同,Tag 文件必须使用<taglib>指令引入该 Web 服务目录下的标记库,才可以使用 Tag 标记来调用相应的 Tag 文件。<taglib>指令的格式如下:

```
<%@ 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" %>
```

在例 3-6 中,FirstTag. tag 文件使用 Tag 标记调用 SecondTag. tag 文件。SecondTag . tag 文件负责从四组数中随机取出 $m$ 个($m \leqslant 52$),这四组数相同,都是由 1～13 组成的 13 个整数。实际上,SecondTag. tag 就是模拟从一副扑克牌中(不包括大王和小王)随机抽取 $m$ 张牌,其中 $m$ 的值由 FirstTag. tag 文件提供。FirstTag. tag 将 SecondTag. tag 返回的 $m$ 个随机数从小到大排列,并计算出它们的和。example3_6. jsp 使用 Tag 标记调用 FirstTag. tag,example3_6. jsp 的效果如图 3-6 所示。

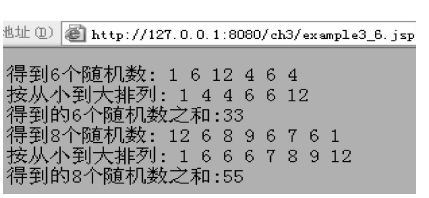



图 3-6　调用 Tag 文件得到随机数

**例 3-6**

**example3_6. jsp**

```
<%@ page contentType = "text/html; Charset = GB2312" %>
<%@ page import = "java.util.*" %>
<%@ taglib tagdir = "/WEB-INF/tags" prefix = "first" %>
<HTML><BODY bgcolor = cyan>
  <first:FirstTag number = "6"/>
  <first:FirstTag number = "8" />
</BODY></HTML>
```

**FirstTag. tag**

```
<%@ tag import = "java.util.*" %>
<%@ taglib tagdir = "/WEB-INF/tags" prefix = "getNumber" %>
<%@ attribute name = "number" required = "true" %>
   <getNumber:SecondTag number = "<% = number %>"/>
   <% out.println("得到" + number + "个随机数:");
       for(int i = 0; i < listNumber.size(); i++ ) //listNumber 是 SecondTag 文件返回的对象
           out.print((Integer)listNumber.get(i) + " ");
       out.println("<br>按从小到大排列:");
       for(int i = 0; i < listNumber.size(); i++ ){
             for(int j = i + 1; j < listNumber.size(); j++ ){
                int a = ((Integer)listNumber.get(i)).intValue();
                int b = ((Integer)listNumber.get(j)).intValue();
                if(b < a){
                  Integer temp = (Integer)listNumber.get(i);
                  listNumber.set(i,(Integer)listNumber.get(j));
                  listNumber.set(j,temp);
                }
            }
       }
       for(int i = 0; i < listNumber.size(); i++ )
            out.print((Integer)listNumber.get(i) + " ");
       int sum = 0;
       for(int i = 0; i < listNumber.size(); i++ )
           sum = sum + ((Integer)listNumber.get(i)).intValue();
       out.println("<br>得到的" + number + "个随机数之和:" + sum + "<br>");
%>
```

**SecondTag. tag**

```
<%@ tag import = "java.util.*" %>
<%@ attribute name = "number" required = "true" %>
<%@ variable name-given = "listNumber"
               variable-class = "java.util.LinkedList" scope = "AT_END" %>
<%    int count = Integer.parseInt(number);
       LinkedList listBox = new LinkedList(),
                  listNeeded = new LinkedList();
       for(int k = 1; k <= 4; k++ )
            for(int i = 1; i <= 13; i++ )
               listBox.add(new Integer(i));
```

```
        while(count > 0) {
            int m = (int)(Math.random() * listBox.size());
            Integer integer = (Integer)listBox.get(m);
            listNeeded.add(integer);
            listBox.remove(m);
            count -- ;
        }
        jspContext.setAttribute("listNumber",listNeeded); //返回 listNumber
%>
```

## 3.5 Tag 标记的嵌套

使用 Tag 标记时,可以带有标记体,标记体还可以是一个 Tag 标记,这就实现了 Tag 标记的嵌套。Tag 标记中的标记体由 Tag 文件的<jsp:doBody/>标记负责处理,而在 Tag 文件中,<jsp:doBody />标记被替换成对“标记体”进行处理后所得到的结果。

在例 3-7 中,JSP 页面使用 Tag 标记嵌套显示一个表格,example3_7.jsp 的效果如图 3-7 所示。


图 3-7 Tag 标记的嵌套

**例 3-7**

**example3_7.jsp**

```
<%@ page contentType = "text/html; Charset = GB2312" %>
<%@ taglib tagdir = "/WEB - INF/tags" prefix = "ok" %>
<html><body>
<p>
<Font size = 2>Tag 标记嵌套显示学生名单:</Font>
<table border = 1>
   <ok:Biaoge color = "#a9f002" name = "姓名" sex = "性别">
       <ok:Biaoge color = "cyan" name = "张三" sex = "男"/>
       <ok:Biaoge color = "#afc0ff" name = "李小花" sex = "女"/>
       <ok:Biaoge color = "pink" name = "孙六" sex = "男"/>
       <ok:Biaoge color = "#ffaaef" name = "赵扬" sex = "女"/>
   </ok:Biaoge>
</table>
</body></html>
```

**Biaoge.tag**

```
<%@ attribute name = "color" %>
<%@ attribute name = "name" %>
<%@ attribute name = "sex" %>
 <tr bgcolor = "<% = color %>">
     <td width = 60><% = name %></td>
     <td width = 60><% = sex %></td>
 </tr>
 <jsp:doBody/>
```

## 3.6 实验 1：使用标记体

**1. 相关知识点**

Tag 文件是扩展名为.tag 的文本文件，其结构几乎和 JSP 文件相同，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、成员变量和方法的声明、Java 程序片和 Java 表达式。JSP 页面使用 Tag 标记动态执行一个 Tag 文件。当 JSP 页面调用一个 Tag 文件时可能希望动态地向该 Tag 文件传递信息，那么就可以使用带有标记体的 Tag 标记来执行一个 Tag 文件，Tag 标记中的“标记体”就会传递给相应的 Tag 文件。标记体由 Tag 文件的＜jsp：doBody/＞标记负责处理，即＜jsp：doBody /＞标记被替换成对“标记体”处理后所得到的结果。

**2. 实验目的**

本实验的目的是让学生灵活掌握在 Tag 标记中使用标记体。

**3. 实验要求**

编写一个 JSP 页面：putImage.jsp 和一个 Tag 文件 Image.tag。JSP 页面通过调用 Tag 文件来显示若干幅图像，通过使用标记体将 HTML 图像标记传递给被调用的 Tag 文件。

1）putImage.jsp 的具体要求

要求 putImage.jsp 页面使用带标记体 Tag 标记调用 Tag 文件来显示一幅图像，即标记体是“显示图像的 HTML 标记”，如下所示：

```
<pic：Image>
      <image src = "a.jpg" />
</pic：Image>
```

putImage.jsp 的页面效果如图 3-8 所示。

2）Image.tag 的具体要求

Image.tag 使用＜jsp：doBody/＞处理标记体，并将图像显示在表格的单元中，要求表格每行有三个单元，这三个单元重复显示一幅图像。

表格每行重复显示一幅图像

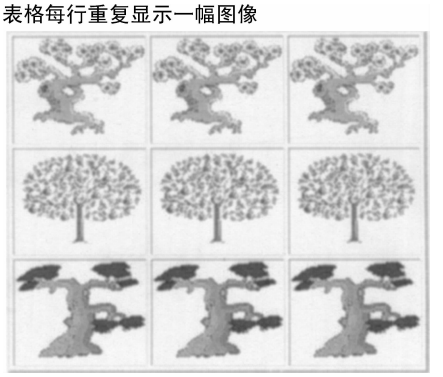


图 3-8　putImage.jsp 的页面效果

**4. 参考代码**

代码仅供参考，学生可按着实验要求，参考本代码编写代码。

JSP 页面参考代码如下：

**putImage.jsp**

```
<%@ page contentType = "text/html；Charset = GB2312" %>
<%@ taglib tagdir = "/WEB-INF/tags" prefix = "pic" %>
<html><body>
  <Font size = 2 color = blue>表格每行重复显示一幅图像</font>
  <table border = 2>
     <pic：Image>
        <image src = "a.jpg" width = 80 height = 60/>
```

```
    </pic:Image>
    <pic:Image>
        <image src="b.jpg" width=80 height=60/>
    </pic:Image>
     <pic:Image>
        <image src="c.jpg" width=80 height=60/>
    </pic:Image>
  </table>
  </body></html>
```

Tag 文件参考代码如下:

**Image.Tag**

```
<tr>
      <td><jsp:doBody/></td>
      <td><jsp:doBody/></td>
      <td><jsp:doBody/></td>
</tr>
```

## 3.7 实验 2:使用 attribute 指令和 variable 指令

**1. 相关知识点**

一个 Tag 文件中通过使用 attribute 指令:

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

使得 JSP 页面调用 Tag 文件时,可以向该 Tag 文件中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" />
```

或

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

Tag 文件可以把一个对象返回给调用它的 JSP 页面。步骤如下:

(1) Tag 文件使用 variable 指令:

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

给出返回的对象的名字、类型和有效范围。

(2) 将返回的对象的名字和引用存储在内置对象 jspContext 中:

```
jspContext.setAttribute("对象名字",对象的引用);
```

**2. 实验目的**

本实验的目的是让学生灵活掌握在 Tag 标记中使用 attribute 指令和 variable 指令。

**3. 实验要求**

编写一个 Tag 文件 GetArea.tag,负责计算三角形或梯形的面积,并将计算结果返回给

调用该 Tag 文件的 JSP 页面。编写一个 JSP 页面 inputAndShow.jsp，该页面负责向 Tag 文件提交三角形三边的长度或梯形的上底、下底和高，并负责显示 Tag 文件返回的相应面积。

1）inputAndShow.jsp 的具体要求

inputAndShow.jsp 提供一个表单。用户可以在表单中分别输入三个数值，并选择这三个数值代表三角形三边的长度或梯形的上底、下底和高，然后提交给当前页面。inputAndShow.jsp 通过 Tag 标记调用 GetArea.tag 文件，并向该 Tag 文件 GetArea.tag 传递三角形三边的长度或梯形的上底、下底和高。inputAndShow.jsp 负责显示 Tag 文件 GetArea.tag 返回的面积以及相关信息。inputAndShow.jsp 的效果如图 3-9 所示。

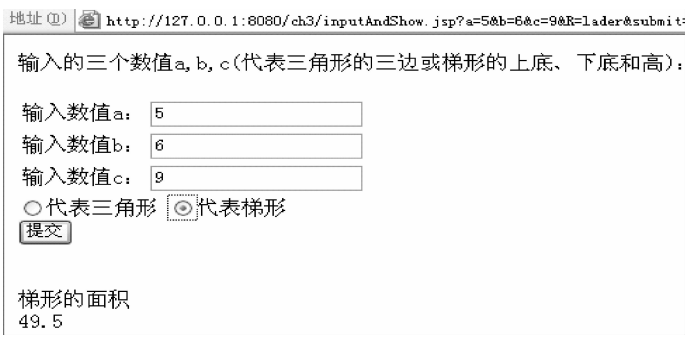



图 3-9　inputAndShow.jsp 的效果

2）GetArea.tag 的具体要求

要求 Tag 文件 GetArea.tag 使用 attribute 指令得到 JSP 页面传递过来三角形三边的长度或梯形的上底、下底和高，使用 variable 指令返回相应的面积以及字符串信息："三角形的面积"或"梯形的面积"。

**4. 参考代码**

代码仅供参考，学生可按着实验要求，参考本代码编写代码。

JSP 页面参考代码如下：

**inputAndShow.jsp**

```
<%@ page contentType="text/html; Charset=GB2312" %>
<%@ taglib tagdir="/WEB-INF/tags" prefix="computer" %>
<HTML>
输入的三个数值 a,b,c(代表三角形的三边或梯形的上底、下底和高):
<BODY color=cyan>
    <FORM action="" method=get name=form>
    <table>
    <tr><td>输入数值 a: </td>
          <td><INPUT type="text" name="a"></td>
    </tr>
   <tr><td>输入数值 b: </td>
          <td><INPUT type="text" name="b"></td>
    </tr>
   <tr><td>输入数值 c: </td>
          <td><INPUT type="text" name="c"></td>
   </tr>
```

```
    </table>
    <INPUT type="radio" name="R" value="triangle">代表三角形
    <INPUT type="radio" name="R" value="lader">代表梯形
    <br><INPUT TYPE="submit" value="提交" name=submit>
  </FORM>
  <%   String a=request.getParameter("a");
       String b=request.getParameter("b");
       String c=request.getParameter("c");
       String cd=request.getParameter("R");
       if(a==null||b==null||c==null){
          a="0";
          b="0";
          c="0";
          cd="0";
       }
       if(a.length()>0&&b.length()>0&&c.length()>0){
   %>  <computer:GetArea numberA="<%=a%>" numberB="<%=b%>"
                         numberC="<%=c%>" condition="<%=cd%>"/>
       <br><%=message%>
       <br><%=area %>
  <% }
   %>
</BODY></HTML>
```

Tag 文件参考代码如下：

**GetArea.Tag**

```
<%@ attribute name="numberA" required="true" %>
<%@ attribute name="numberB" required="true" %>
<%@ attribute name="numberC" required="true" %>
<%@ attribute name="condition" required="true" %>
<%@ variable name-given="area" variable-class="java.lang.Double" scope="AT_END" %>
<%@ variable name-given="message" scope="AT_END" %>
<%!
    public double getTriangleArea(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));
          return area;
       }
       else
         return -1;
    }
    public double getLaderArea(double above,double bottom,double h){
        double area=(above+bottom)*h/2.0;
        return area;
    }
%>
<% try{  double a=Double.parseDouble(numberA);
         double b=Double.parseDouble(numberB);
         double c=Double.parseDouble(numberC);
```

```
            double result = 0;
            if(condition.equals("triangle")){
               result = getTriangleArea(a,b,c);
               jspContext.setAttribute("area",new Double(result));
               jspContext.setAttribute("message","三角形的面积");
            }
            else if(condition.equals("lader")){
               result = getLaderArea(a,b,c);
               jspContext.setAttribute("area",new Double(result));
               jspContext.setAttribute("message","梯形的面积");
            }
      }
       catch(Exception e){
            jspContext.setAttribute("area",new Double(-1.0));
            jspContext.setAttribute("message","" + e.toString());
       }
%>
```

# 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 将返回的三角形的面积保留最多 3 位小数，two.jsp 将返回的三角形的面积保留最多 6 位小数。

8. 参照例 3-7，编写一个 JSP 页面 lianxi8.jsp，该页面通过使用 Tag 标记的嵌套显示的效果如图 3-10 所示。

| 姓名 | 电话 | email |
|---|---|---|
| 张三 | 12345678 | ss@163.com |
| 李小花 | 9876543 | cc@163.com |
| 孙六 | 11223355 | pp@163.com |
| 吴老二 | 66553377 | ee@163.com |

图 3-10　嵌套显示的效果