第3章

Java Web开发环境搭建

Tomcat是一款小型的轻量级应用服务器发布软件，在中小型系统和并发访问用户不是很多的场合下被普遍使用。对Java初学者来说，Tomcat是开发和调试JSP程序的首选，随着Tomcat新版本的不断发布，Tomcat在企业Web应用服务器市场上的份额也越来越大。

在Tomcat中，应用程序的部署很简单，只需将WAR或整个应用程序目录放到Tomcat的webapps目录下即可。Tomcat会自动检测文件，如果是WAR包，Tomcat将自动将其解压。在浏览器中访问这个应用的JSP时，通常第一次会很慢，因为Tomcat要将JSP转换成Servlet类文件，然后再对其编译。后面访问编译后的文件时速度会很快。另外Tomcat也提供了一个应用管理器（访问这个应用管理器需要用户名和密码），将用户名和密码存储在一个XML文件中。通过这个应用管理器并借助FTP协议，用户可以在远端通过Web部署和撤销应用。

实验3.1　第一个JSP动态网页

【实验任务】

编写JSP动态网页，要求页面运行时，根据当前系统时间输出相应的问候语。

【实验步骤】

（1）编辑date.jsp程序，代码如下。

```
<%@ page language="java" import="java.util.*" pageEncoding="gbk"%>
    <html>
     <head>
          <title>Date</title>
     </head>
    <body>
     <%
       //Calendar类表示时间，现在其更为常用，所以选择使用它
```

```
        Calendar date = Calendar.getInstance();  //Calendar对象不能简单地用new创建
        int year = date.get(Calendar.YEAR);                                    //年
        int month = date.get(Calendar.MONTH) + 1;                              //月,需+1调整
        int day = date.get(Calendar.DATE);                                     //日
        int weekDay = date.get(Calendar.DAY_OF_WEEK) - 1;                      //星期几
        String[] sweekDay = {"日","一","二","三","四","五","六"};    //将英文转换为中文
        int hour = date.get(Calendar.HOUR_OF_DAY);                             //得到现在的时间
        String welcome = "";
          if(hour >= 0&&hour < 12){
              welcome = "上午";  }
          elseif(hour > 12&&hour < 18){
               welcome = "下午"; }
          else{
               welcome = "晚上"; }
    %>
    <% -- 将时间按格式显示出来 -- %>
    <% = welcome %>好!今天是<% = year %>年<% = month %>月<% = day %>日,星期<% = sweekDay
[weekDay] %>
  </body>
</html>
```

(2) 在浏览器输入本程序页面的 URL,请求资源,观察显示效果。

实验 3.2 WAR 包及发布

【实验任务】

将 MyEclipse 下建立的 JSP Web 项目打包,生成 WAR 包,在 Tomcat 下部署并发布。

【实验步骤】

(1) 首先是使用 MyEclipse 将 Web 项目打包。

① 右击项目,选择 Export 选项,如图 3-1 所示。

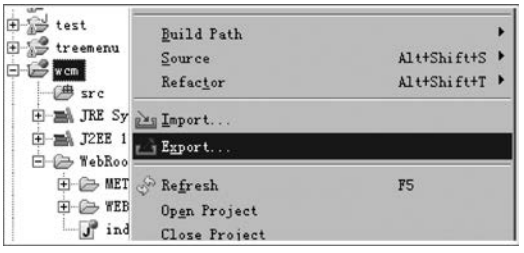

图 3-1 项目生成 WAR 包

② 然后选择 Java EE→WAR File 选项,单击 Next 按钮。

③ 指定 WAR 包的存放路径,如图 3-2 所示。

图 3-2　指定 WAR 包的存放路径

（2）打包完成以后将 WAR 文件直接放到 Tomcat 的 webapps 目录下即可，如图 3-3 所示。

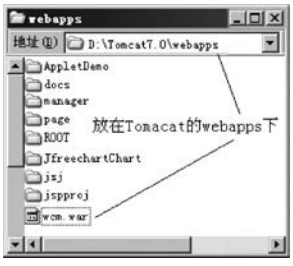

图 3-3　发布 WAR 包

（3）运行 Tomcat。在浏览器输入该项目的 URL 进行测试，可先视其效果和正常的 Web 项目发布相同。

第4章

JSP技术基础

一个 JSP 页面可由普通的 HTML 标记、JSP 标记、成员变量与方法的声明、Java 程序片和 Java 表达式组成。JSP 引擎把 JSP 页面中的 HTML 标记交给客户的浏览器执行显示，而该引擎自身则负责处理 JSP 标记、变量和方法声明，并负责运行 Java 程序片、计算 Java 表达式，最终将需要显示的结果发送给客户的浏览器。

在 JSP 页面中，成员变量是被所有用户共享的变量。Java 程序片可以操作成员变量，任何一个用户对 JSP 页面成员变量操作的结果。都会影响其他用户。如果多个用户访问一个 JSP 页面，那么该页面中的 Java 程序片就会被执行多次，分别运行在不同的线程中，即运行在不同的时间片内。运行在不同线程中的 Java 程序片的局部变量互不干扰，即使一个用户改变 Java 程序片中的局部变量的值也不会影响其他用户在 Java 程序片中的局部变量。

【实验目的】

让学生掌握在 JSP 页面中使用 JSP 标签的方法，重点掌握 Java 程序片段、JSP 表达式及 JSP 内置对象的使用方法，熟练掌握 JSP 的程序设计方法。

实验 4.1　JSP 程序段

【实验任务】

编写两个 JSP 页面，其 JSP 文件分别为 inputName.jsp 和 visitperson.jsp。

inputName.jsp 的具体要求：该页面有一个表单，用户通过该表单输入自己的姓名并提交给 visitperson.jsp 页面。

visitperson.jsp 页面的具体要求：该页面有名字为 person、类型是 StringBuffer 及名字为 count、类型是 int 的成员变量。页面包含 public void judge()方法。该方法负责创建 person 对象，当 count 的值是 0 时，judge()方法创建 person 对象。页面包含 public void addPerson(String p)方法，该方法将参数 p 指定的字符串尾加到操作成员变量 person，同时

将 count 作自增运算。

visitperson.jsp 页面在 JSP 程序段中获取 inputName.jsp 页面提交的姓名，然后调用 judge()方法创建 person 对象、调用 addPerson()方法将用户的姓名追加到成员变量 person。

如果 inputName.jsp 页面没有提交姓名或姓名含有的字符个数大于 10，就使用<jsp:forward>标记将用户请求转到 inputName.jsp 页面。

通过 JSP 表达式输出 person 和 count 的值。

【实验步骤】

(1) 分别编写 inputName.jsp 和 visitperson.jsp 两个文件的代码。

inputName.jsp 参考代码如下所示。

```
<%@ page contentType="text/html;charset=utf-8" %>
<html>
    <body bgcolor=cyan>
        <font size=3>
            <form action="visitperson.jsp" method="get">
                请输入姓名:
                <input type="text" name="name"><br>
                <input TYPE="submit" value="加入访问者行列">
            </form>
    </body>
</html>
```

visitperson.jsp 参考代码如下所示。

```
<%@ page contentType="text/html;charset=utf-8" %>
<html>
    <body bgcolor=yellow>
        <%!int count;
           StringBuffer person;
           public void judge() {
             if (count == 0)
                 person = new StringBuffer();
           }
           public void addPerson(String p) {
             if (count == 0) {
                person.append(p);
             } else {
                person.append("," + p);
           }
             count++;
         }%>
     <%
           String name = request.getParameter("name");
           byte bb[] = name.getBytes("iso-8859-1");
           name = new String(bb,"utf-8");
```

```
            if (name == null||name.length() == 0 || name.length() > 10) {
        %>
    <jsp:forward page = "inputName.jsp" />
     <%
         }
         judge();
         addPerson(name);
    %>
         <br> 目前共有<% = count %>人访问过该站点,他们的名字是: <br>
         <font size = 3 ><% = person %></font ><p>
         <a href = "inputName.jsp">返回 inputName.jsp 页面</a ><p>
     </body>
</html>
```

（2）在地址栏输入 inputName.jsp 文件的 URL 地址，观察浏览器显示效果，如图 4-1 所示。

图 4-1　统计访问站点人数

（3）拓展实验。

修改程序，将访问者显示方式改为表格。

实验 4.2　JSP 指令标记

page 指令："<%@ page 属性 1="属性 1 的值" 属性 2= "属性 2 的值" …%>"用来定义整个 JSP 页面的一些属性和这些属性的值。比较常用的两个属性是 contentType 和 import。page 指令只能为 contentType 属性指定一个值，但可以为 import 属性指定多个值。

contentType 是让浏览器知道下载的文件要保存为什么类型。当然，真正的文件还是在网络数据流里面的数据，设定一个下载类型并不会去改变网络数据流里的内容。

include 指令标记："<%@ include file= "文件的 URL" %>"的作用是在 JSP 页面出现该指令的位置处静态插入一个文件。被插入的文件必须是可访问和可使用的，如果该文件和当前 JSP 页面在同一 Web 服务目录中，那么"文件的 URL"就是文件的名字；如果该文件在 JSP 页面所在的 Web 服务目录的一个子目录中，如 fileDir 子目录中，那么"文件的 URL"就是"fileDir/文件的名字"。include 指令标记在编译阶段就会处理所需要的文件，被处理的文件在逻辑和语法上依赖当前 JSP 页面，其优点是页面的执行速度快。

【实验任务】

编写三个 JSP 页面：first.jsp、second.jsp 和 third.jsp。另外，要求用"记事本"编写一

个文本文件 hello.txt。hello.txt 的每行有若干个英文单词，单词之间用空格分隔，每行之间用< br >分隔。

first.jsp 的具体要求：使用 page 指令设置 contentType 属性的值是“text/plain”，使用 include 指令静态插入 hello.txt 文件。

second.jsp 的具体要求：second.sp 使用 page 指令设置 contentType 属性的值是“application/mspowerpoint”，使用 include 指令静态插入 hello.txt 文件。

third.jsp 的具体要求：使用 page 指令设置 contentType 属性的值是“application/msword”，使用 include 指令静态插入 hello.txt 文件。

本实验的目的是让学生掌握在 JSP 页面中使用 include 指令在 JSP 页面中静态插入一个文件内容的方法，体会 page 指令 contentType 属性值的作用。

【实验步骤】

(1) 编写文本文件 hello.txt 和三个 JSP 页面：first.jsp、second.jsp 和 third.jsp。

hello.txt 文件参考代码如下所示。

```
nantong university jiangsu China
< br >
private throw class hello welcome
```

first.jsp 文件参考代码如下所示。

```
<%@ page contentType = "text/plain" %>
<HTML>
 <BODY>
   <FONT size = 4 color = blule>
      <%@include file = "hello.txt" %>
  </FONT>
 </BODY>
</HTML>
```

second.jsp 文件参考代码如下所示。

```
<%@ page contentType = "application/vnd.ms - powerpoint" %>
<HTML>
 <BODY>
   <FONT size = 2 color = blule>
      <%@include file = "hello.txt" %>
   </FONT>
 </BODY>
</HTML>
```

third.jsp 文件参考代码如下所示。

```
<%@ page contentType = "application/msword" %>
<HTML>
 <BODY>
```

```
    <FONT size = 4 color = blule>
      <%@include file = "hello.txt" %>
    </FONT>
  </BODY>
</HTML>
```

（2）在地址栏分别输入这些文件的 URL 并观察浏览器显示效果，体会 Page 指令 contentType 属性的作用。

first.jsp 运行效果如图 4-2 所示。

图 4-2　first.jsp 运行效果

second.jsp 运行效果如图 4-3 所示。

图 4-3　second.jsp 运行效果

third.jsp 运行效果如图 4-4 所示。

图 4-4　third.jsp 运行效果

实验 4.3　JSP 表格实验

【实验任务】

使用 JSP 程序段动态生成表格(表格数据来自数组或集合类容器),为从数据库获取数据做准备,如图 4-5 所示。

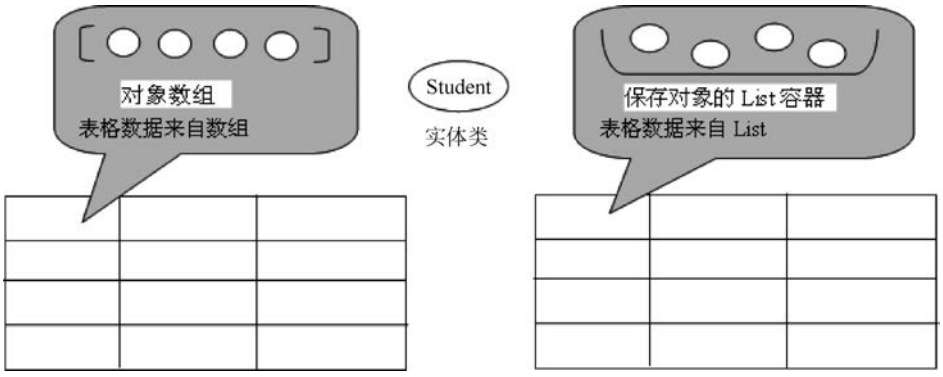

图 4-5　表格数据来自数组或 List 容器

集合类是容器类的数据结构,包括 List、Map、Set 等类。

List:按对象进入的顺序保存对象,不向对象做排序或编辑操作。List 类容器中的值允许重复,因为其为有序的数据结构。

Map:是基于"键"的成对数据结构,Map 类容器的元素键值必须具有唯一性(键不能相同,否则元素的值会被替换)。

Set:对每个对象只接收一次,并使用自己内部的排序方法。Set 类容器中的值不允许重复,且其是无序的。

List 和 Set 是由 Collection 接口派生的两个接口。

【实验步骤】

(1) 编写实体类 bean/Student.java。

Student.java 参考代码如下所示。

```
package bean;
public class Student {
    private String xh;
    private String name;
    private String teleno;
    public Student(){ }
    public Student(String xh,String name,String teleno){
        this.xh = xh;
        this.name = name;
        this.teleno = teleno;
```

```
    }
    public String getXh() {return xh;}
    public void setXh(String xh) {this.xh = xh;}
    public String getName() {return name;}
    public void setName(String name) {this.name = name;  }
    public String getTeleno() {return teleno;}
    public void setTeleno(String teleno) {this.teleno = teleno;}
}
```

（2）编写 JSP 表格程序，表格中的数据分别来自数组和 List 容器。table.jsp 参考代码如下所示。

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@ page import="bean.Student"%>
<html>
    <body>
        <%
            //在数组中预先准备好数据
            Student[] stu = { new Student("001", "欧巴马", "13844488101"),
                    new Student("002", "李刚好", "13848888108"),
                    new Student("003", "胡规范", "18844488158") };
            //在 List 中预先准备好数据
            List lstu = new ArrayList();
            lstu.add(new Student("2101", "黄晓敏", "18843488111"));
            lstu.add(new Student("2103", "季试第", "18844488103"));
            lstu.add(new Student("2104", "章里好", "18745488145"));
        %>
        从数组中取出数据放入表格中
        <table border="1" bgcolor="#aa8899">
            <tr><td>学号</td><td>姓名</td><td>联系电话</td>
            </tr>
            <%
                for (int i = 0; i < stu.length; i++) {
            %>
            <tr>
                <td><%=stu[i].getXh()%></td>
                <td><%=stu[i].getName()%></td>
                <td><%=stu[i].getTeleno()%></td>
            </tr>
            <%
                }
            %>
        </table>
          从 List 中取出数据放入表格中
        <table border="1" bgcolor="#ddaa99">
            <tr><td>学号</td><td>姓名</td><td>联系电话</td></tr>
            <%
                for (int i = 0; i < lstu.size(); i++) {
            %>
            <tr>
                <td><%=((Student) lstu.get(i)).getXh()%></td>
                <td><%=((Student) lstu.get(i)).getName()%></td>
                <td><%=((Student) lstu.get(i)).getTeleno()%></td>
            </tr>
            <%
```