模块二　使用 JSP 内置对象

模块简介：

上一个模块通过安装并配置 JSP 开发环境、在 MyEclipse 环境中部署运行 JSP 项目掌握了开发 JSP 动态网站的步骤，也知道了 JSP 的基本页面组成元素，对 JSP 有了初步认识。

本模块将带领大家进一步学习 JSP 程序开发的重点——JSP 内置对象。JSP 内置对象是指不需要 JSP 页面编写者来实例化，在所有的 JSP 页面中都可以直接使用并由 JSP 容器实现和管理的组件，它起到了简化页面的作用。JSP 内置对象被广泛应用于获取客户端数据、输出信息、保存数据等操作中。

技能目标：

（1）会使用 out 对象向客户端输出内容。
（2）会使用 request 对象处理客户端请求。
（3）会使用 response 对象响应各种信息。
（4）会使用 session 对象实现访问控制。
（5）会使用 application 对象实现数据共享。

知识目标：

（1）掌握 JSP 常用内置对象的使用方法。
（2）掌握表单处理的编程模式。
（3）掌握访问控制的实现流程。
（4）了解 URL 请求参数。

工作任务：

（1）认识 JSP 的内置对象。
（2）使用 out 对象输出信息。
（3）获取客户端请求数据。
（4）实现页面跳转。
（5）实现访问控制。
（6）制作网页计数器。

【问题引入】

JSP 的一个重要特征就是它自带了功能强大的内置对象，那么有哪些内置对象呢？它们的功能又是如何的？

2.1　任务一　认识 JSP 的内置对象

【实现思路】

JSP 的内置对象包括 request、response、out、session、pageContext、application、config、page 和 exception。比较常用的有 out、request、response、session 和 application。通过学习 JSP 内置对象的概念、分类、可见范围及功能这些知识点来帮助我们了解 JSP 的内置对象。

2.1.1　JSP 内置对象概述

Java 是面向对象的，由于 JSP 是使用 Java 作为脚本语言，所以 JSP 具有强大的对象处理能力。在 Java 语法中使用一个对象前，需要先实例化这个对象，这其实是一件比较繁琐的事。为了简化开发，JSP 提供了一些内置对象，这些内置对象在使用时不需要实例化，直接使用即可。内置对象也称为隐含对象或固有对象。内置对象是被 JSP 容器自动定义的对象变量，可以在 JSP 页面的 jspService()方法中自动实例化这些隐含对象。JSP 内置对象大致可分为如下四类：

- 与输入输出有关的内置对象：out、request、response 对象。
- 与上下文（Context）有关的内置对象：session、application、pageContext 对象。
- 与 Servlet 有关的内置对象：page、config 对象。
- 与错误（Error）处理有关的内置对象：exception 对象。

2.1.2　JSP 内置对象的范围

表 2-1 列出了 JSP 的 9 个内置对象的可见范围及其功能描述。

表 2-1　JSP 的内置对象

对象名	对象类型	可见范围	描述
out	javax.servlet.jsp.JspWriter	page	提供对输出流的访问
request	javax.servlet.http.HttpServletRequest	request	提供对 HTTP 请求数据的访问，同时还提供用于加入特定请求数据的上下文
response	javax.servlet.http.HttpServletResponse	page	允许直接访问 HttpServletResponse 对象，可用来向客户端输入数据
session	javax.servlet.http.HttpSession	session	可用来保存在服务器与一个客户端之间需要保存的数据，当客户端关闭网站的所有网页时，session 变量会自动消失
application	javax.servlet.ServletContext	application	代表应用程序上下文，它允许 JSP 页面与包括在同一应用程序中的任何 Web 组件共享信息
pageContext	javax.servlet.jsp.PageContext	page	是 JSP 页面本身的上下文，它提供了唯一一组方法来管理具有不同作用域的属性，这些 API 在实现 JSP 自定义标签处理程序时非常有用

续表

对象名	对象类型	可见范围	描述
page	java.lang.Object，即 HttpJspBase	page	代表 JSP 页面对应的 Servlet 类实例
config	javax.servlet.ServletConfig	page	允许将初始化数据传递给一个 JSP 页面
exception	java.lang.Throwable，即 Exception	page	含有只能由指定的 JSP“错误处理页面”访问的异常数据

在选择范围时，应遵循如下原则：

- 如果数据只在一个页面用到，就用 page 范围。
- 如果数据在多个页面用到，就用 request 范围。
- 如果数据在多个请求中用到，就用 session 范围。
- 如果数据在多个 session 中用到，就用 application 范围。

【问题引入】

out 对象是常用的 JSP 内置对象之一，通过 out 对象可以向客户端浏览器输出信息，并且管理应用服务器上的输出缓冲区，如何使用 out 对象呢？

2.2 任务二 使用 out 对象输出信息

【实现思路】

在 JSP 文件中使用 out 对象的 print()或 println()方法可在页面上输出信息，out 对象的 clear()方法可用来管理缓冲区。

2.2.1 向客户端输出数据

out 对象一个最基本的应用就是向客户端浏览器输出信息。out 对象可以输出各种数据类型的数据，在输出非字符串类型的数据时，会自动转换为字符串进行输出。out 对象提供了 print()和 println()两种向页面中输出信息的方法，不同的是，println()还在后面添加一个空行，不过这个空行被浏览器解析时忽略。

【例 2.1】使用 out 对象向页面输出数据。

清单 2-1 out1.jsp

```
<%@ page language="java" contentType="text/html; charset=GBK"
    pageEncoding="GBK"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>使用 out 对象案例 1</title>
</head>
<body>
<%
out.print("使用 out 对象的 print 方法输出信息。");
```

```
out.println("使用 out 对象的 println 方法输出字符串。");
out.println("实现换行了<br>");
out.print("<br>");
out.print(3.14159);
 %>
<br>
 <%="这是使用表达式输出" %>
</body>
</html>
```

本案例的运行效果如图 2-1 所示。可见在使用 print()方法和 println()方法向页面输出信息时并不能很好地区分出两者的区别，为了能真正实现在页面中换行，需要通过 print("
")或 println("
")方法实现。通过 out 对象的 print 和 println 方法向客户端浏览器输出信息与使用 JSP 表达式输出信息相同。

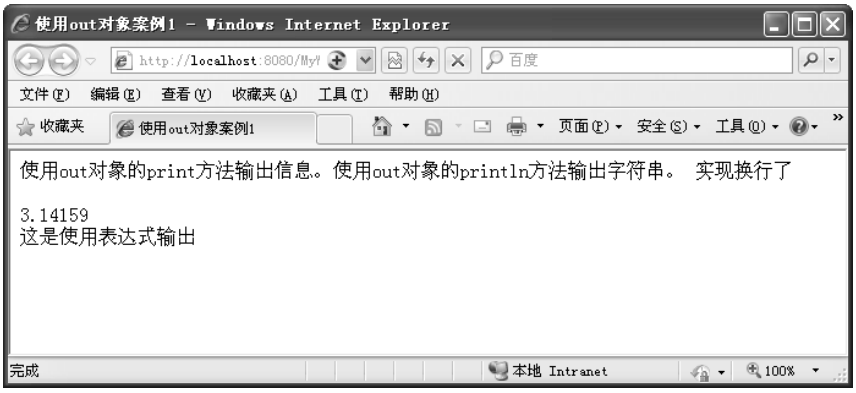

图 2-1　案例运行结果

2.2.2　管理缓冲区

out 对象的一个重要功能就是对缓冲区进行管理。表 2-2 列出了 out 对象管理缓冲区的方法。

表 2-2　out 对象管理缓冲区的方法

方法	说明
void clear()	清除缓冲区的内容，但不把数据输出到客户端
void clearBuffer()	清除缓冲区的当前内容，并把数据输出到客户端
void close()	关闭输出流，清除所有的内容
void flush()	立即输出缓冲区里的数据
int getBufferSize()	返回缓冲区以字节数的大小（KB），如不设缓冲区则为 0
int getRemaining()	返回缓冲区中剩余空间的大小
boolean isAutoFlush()	返回缓冲区满时是自动清空还是抛出异常。若返回 true，则缓冲区满时是自动清空；若返回 false，则缓冲区满时抛出异常

【问题引入】

动态网页具有交互性，而能够处理客户端请求是与用户进行信息交互的基础。那么服务器是如何获取客户端请求数据的呢？

2.3 任务三 获取客户端请求数据

【实现思路】

request 对象是 JSP 中最常用的对象之一，用于封装客户端的请求信息，通过调用相应的方法可以获取客户端提交的信息。用户可以使用 HTML 表单提交客户端数据，也可以采用请求参数的方式将客户端数据提交给服务器。下面通过案例方式来深入学习如何使用 request 对象的相关方法获取表单数据及请求参数。

2.3.1 获取 HTML 表单提交的数据

用户通常使用 HTML 表单向服务器的某个 JSP 页面提交信息，一般格式为：

```
<form method="get/post" action="表单要提交到的地点">
[接收数据的表单组件]
[数据提交控件]
</form>
```

【例 2.2】用户在图 2-2 所示的注册页面 reginput.jsp 中填写注册信息，单击“提交”按钮后显示输入的注册信息，如图 2-3 所示。

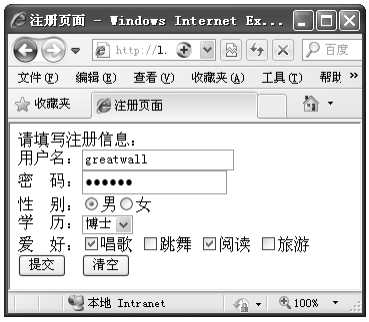

图 2-2 输入注册信息

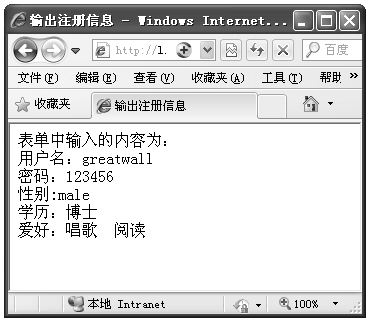

图 2-3 页面提交后显示注册信息

实现如下：

（1）编写 reginput.jsp 页面，包含注册表单，该表单数据提交到 doreg.jsp 页面。

清单 2-2 reginput.jsp

```
<%@ page language="java" contentType="text/html; charset=GBK"
    pageEncoding="GBK"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
```

```
<title>注册页面</title>
</head>
<body>
请填写注册信息：<br>
<form action="doreg.jsp" method="post">
用户名：<input type="text" name="uname"><br>
密  码：<input type="password" name="pwd"><br>
性  别：<input type="radio" name="sex" value="male">男<input type="radio" name="sex"
value="female">女<br>
学  历：<select name="study">
<option value="本科">本科</option>
<option value="硕士">硕士</option>
<option value="博士">博士</option>
</select><br>
爱  好：<input type="checkbox" name="like" value="唱歌">唱歌
<input type="checkbox" name="like" value="跳舞">跳舞
<input type="checkbox" name="like" value="阅读">阅读
<input type="checkbox" name="like" value="旅游">旅游<br>
<input type="submit" value="提交">  <input type="reset" value="清空">
</form>
</body>
</html>
```

（2）编写 doreg.jsp 页面，用于获取 reginput.jsp 页面中填写的注册信息并显示。

清单 2-3　doreg.jsp

```
<%@ page language="java" contentType="text/html; charset=GBK"
    pageEncoding="GBK"%>
<html>
<head>
<title>输出注册信息</title>
</head>
<%
request.setCharacterEncoding("GBK");                    //设置请求的编码
String name=request.getParameter("uname");
String password=request.getParameter("pwd");
String xb=request.getParameter("sex");
String xl=request.getParameter("study");
String ah[]=request.getParameterValues("like");         //获取多选项的值
 %>
<body>
表单中输入的内容为：<br>
用户名：<%=name %><br>
密码：<%=password %><br>
性别：<%=xb %><br>
学历：<%=xl %><br>
爱好：<% if(ah!=null){
```

```
    for(int i=0;i<ah.length;i++)
    out.print(ah[i]+"  ");
    } %><br>
    </body>
    </html>
```

reginput.jsp 中表单信息提交至 doreg.jsp 文件处理，表单信息的发送方式采用 post 方式。post 方式会将表单的内容通过 http 发送，在地址栏中看不到表单的提交信息，而且使用 post 方式发送信息没有字符长度的限制。图 2-4 所示就是采用 post 方式发送信息并使用 doreg.jsp 接收信息后的页面。

若把 reginput.jsp 中表单信息的发送方式改为 get（即 method="get"），那么表单内容经过编码之后通过 URL 发送（可以在地址栏中看到表单信息，不安全，一般不建议使用 get 方式）。使用 get 方式发送信息时有 255 个字符的限制。图 2-5 所示就是采用 get 方式发送信息。

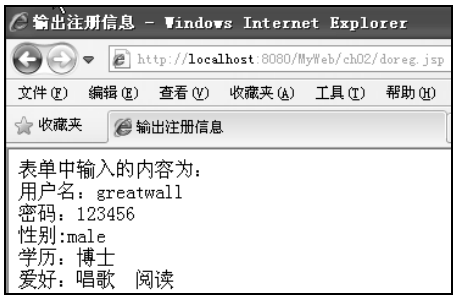

图 2-4 采用 post 方式发送信息

图 2-5 采用 get 方式发送信息

doreg.jsp 中使用 request 对象的 getParameter 方法获得上一个页面表单中文本框、密码框、单选框、下拉框所提交的单个参数值，如使用 getParameter ("uname")方法可以获取到一个字符串，其中 uname 为表单控件的名称；使用 request 对象的 getParameterValues 方法获得上一个页面中复选框所提交的多个参数值，如使用 getParameterValues("like")方法可以获取到一个字符串数组，这个数组中存储的就是所有选中的复选项对应的值。

表单中输入或选择的数据有可能是中文的，为了避免出现乱码问题，需要使用 request 对象的 setCharacterEncoding 方法，指定请求的编码方式，一般为 GBK 或 UTF-8。调用 request 对象的 setCharacterEncoding 方法的语句一定要在页面中没有调用任何 request 对象的方法时才能使用，否则该语句将不起作用。

2.3.2 获取访问请求参数

request 对象用于处理 HTTP 请求中的各项参数。在这些参数中，最常用的就是获取访问请求参数。在表单中采用 get 方式提交数据时，在地址栏中显示的地址为 http://localhost:8080/MyWeb/ch02/doreg.jsp?uname=greatwall&pwd=123456&sex=male&study=%B2%A9%CA%BF&like=%B3%AA%B8%E8&like=%D4%C4%B6%C1，我们发现客户端要传递给目标文件的数据在“?”后面，也就是数据和目标文件之间用“?”隔开，数据的格式为“请求参数名=参数值”。如果有多个请求参数要传递，多个参数值对之间通过“&”分隔开。

使用请求参数形式传递数据的方法通常用在超级链接中，当传递数据不多时，可以直接

通过链接来传递数据。

【例 2.3】进行翻页时，通过超级链接传递当前页码。

```
<a href="index.jsp?page=<%=prep %>">上一页</a><br>
<a href="index.jsp?page=<%=nextp %>">下一页</a>
```

当点击上一页超级链接时，会打开 index.jsp 页面，同时通过 page 参数传递当前页码（表达式 prep 的值），于是可以在 index.jsp 中使用 request 对象的 getParameter 方法获得传递的 page 参数的值，具体代码如下：

```
<% String p=request.getParameter("page");%>
```

在使用 request 对象的 getParameter 方法获得传递的参数值时，如果指定的参数不存在，将返回 null；如果指定了参数名，但未指定参数值，将返回空的字符串""。

【例 2.4】处理获取请求参数时乱码。

（1）编写 req1.jsp 页面，包含一个超级链接，通过该链接传递若干参数，其中包含中文参数值。

清单 2-4　req1.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<title>超级链接传递参数</title>
</head>
<body>
<a href="req2.jsp?user=王&id=2">到 req2.jsp</a>
</body>
</html>
```

（2）编写 req2.jsp 页面，用于获取 req1.jsp 页面传递过来的请求参数并显示。

清单 2-5　req2.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<title>获取请求参数</title>
</head>
<body>
<%
String n=new String(request.getParameter("user").getBytes("iso-8859-1"),"utf-8") ;
int id=Integer.parseInt(request.getParameter("id"));          //类型转换为 int
%>
<%=n %><br>
<%=id %>
</body>
</html>
```

使用超级链接进行数据传递时采用 get 方式提交请求，如果在传递数据中存在中文，由于

请求参数采用的是 ISO-8859-1 编码，不支持中文，若使用 request 对象直接获取时容易产生乱码问题，为避免乱码问题，需要对获取的数据重新进行编码。由于使用 request 对象获取的数据类型均是 String 类型，因而可以将获取到的数据通过 String 构造方法使用 UTF-8 或 GBK 编码重新构造一个 String 对象，才可以正确显示出中文，见 reg2.jsp 中粗体字显示代码部分。

2.3.3 实训 获取用户留言数据

训练要点：request 对象的使用。

需求说明：在留言管理系统中，用户要发表留言，请编写程序获取并在页面中显示用户提交的留言数据，系统运行效果如图 2-6 和图 2-7 所示。

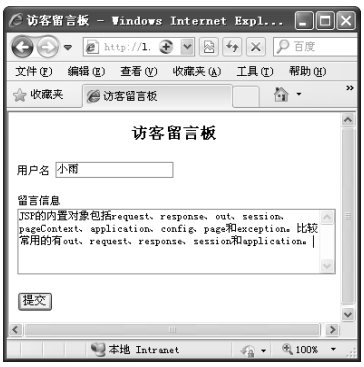

图 2-6 留言页面图

图 2-7 显示留言页面

实现思路：编写留言 JSP 页面；编写显示留言 JSP 页面（获取留言页面表单输入的信息并显示）。

【问题引入】

使用 JSP 处理客户端请求时一般遵循这样一种模式：首先，用户通过表单控件输入并提交信息或程序通过请求参数方式提交信息；接着，JSP 页面获得客户端的请求数据并进行处理；最后，JSP 页面根据处理结果转向不同的结果页面。获取客户端请求信息我们使用 request 对象的 getParameter 方法或 getParameterValues 方法，那么如何实现页面转向呢？

2.4 任务四 实现页面跳转

【实现思路】

在 JSP 中可以使用重定向及转发方式来实现页面转向。

2.4.1 转发与重定向

转发简单地说就是通过一个中介将甲方的请求传递给乙方。从程序运行的角度解答就是当客户端发送一个请求到服务器后，Web 服务器调用内部的方法在容器内部完成请求处理和转发动作，然后将目标资源发送给浏览器，整个过程都是在一个 Web 容器内完成，因而可以共享

request 范围内的数据。而对于到客户端，不管服务器内部如何处理，作为浏览器都只是提交了一个请求，因而客户端的 URL 地址不会发生改变。使用 request 的 getRequestDispatcher()方法可以实现转发，通过转发能在多个页面交互过程中实现请求数据的共享。

使用 response 对象的 sendRedirect()方法可以实现重定向。重定向是指客户端重新向服务器请求一个地址链接，由于是发送新的请求，因而上次请求中的数据将随之丢失。由于服务器重新定向了 URL，因而在客户端浏览器中显示的是新的 URL 地址，所以重定向可以理解为是浏览器至少提交了两次请求。

转发和重定向都能实现页面的跳转，它们的区别如表 2-3 所示。

表 2-3　转发和重定向的区别

名称	使用的对象	发挥作用的位置	地址栏情况	数据共享
重定向	response	客户端	显示转向后的地址	不共享 request 范围内的数据
转发	request	服务器端	不显示目标地址	共享 request 范围内的数据

2.4.2　使用 response 对象实现重定向

response 对象用于响应客户请求并向客户端输出信息，sendRedirect ()方法是 response 对象最常用的方法，用来将请求重定向到一个不同的 URL 上。sendRedirect ()方法的语法格式为：

response.sendRedirect(String path);

path：用于指定目标路径，可以是相对路径，也可以是不同主机的其他 URL 地址。

【例 2.5】模拟用户登录，登录成功跳转到欢迎页面，登录失败跳转到登录页面。

实现如下：

（1）编写 login.jsp 页面，包含登录表单，表单数据提交到 dologin.jsp 页面，如图 2-8 所示。

清单 2-6　login.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<title>用户登录</title>
</head>
<body>
请输入用户名和密码：
<form action="dologin.jsp" method="post">
用户名：<input type="text" name="uname"><br>
密&amp;nbsp;&amp;nbsp;码：<input type="password" name="pwd"><br>
<input type="submit" value="登录"><input type="reset" value="重置">
</form>
</body>
</html>
```

（2）编写 dologin.jsp 页面，用来获取登录表单中的数据，并验证用户名和密码，根据验证的结果进行跳转。

清单 2-7　dologin.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<title>登录处理</title>
</head>
<body>
<%request.setCharacterEncoding("utf-8");
String name=request.getParameter("uname");
String pwd=request.getParameter("pwd");
if(name.equals("张三")&&pwd.equals("123")) response.sendRedirect("welcome.jsp");
else response.sendRedirect("login.jsp");
 %>
</body>
</html>
```

清单 2-8　welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<title>登录成功</title>
</head>
<body>
<%String name=request.getParameter("uname"); %>
欢迎<%=name %>!
</body>
</html>
```

当输入正确的用户名及密码时，本案例的运行效果如图 2-9 所示。可见重定向显示转向后的地址，但不共享 request 范围内的数据。

图 2-8　login.jsp 页面

图 2-9　welcome.jsp 页面

2.4.3　使用 request 对象实现转发

使用 request 的 getRequestDispatcher()方法可实现转发。通过转发能在多个页面交互过程

中实现请求数据的共享。

【例 2.6】修改 dologin.jsp，使用转发方式实现例 2.5 的功能。

仅需修改例 2.5 中的 dologin.jsp 代码，如清单 2-9 所示。

清单 2-9　修改后的 dologin.jsp 代码

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<title>登录处理</title>
</head>
<body>
<%request.setCharacterEncoding("utf-8");
String name=request.getParameter("uname");
String pwd=request.getParameter("pwd");
if(name.equals("张三")&&pwd.equals("123")) request.getRequestDispatcher("welcome.jsp").
   forward(request,response);
else response.sendRedirect("login.jsp");
 %>
</body>
</html>
```

当输入正确的用户名及密码时，本案例的运行效果如图 2-10 所示。可见转发不显示转向后的地址，但共享 request 范围内的数据。

图 2-10　使用转发后的欢迎页面

2.4.4　实训　猜数游戏

训练要点：request 和 response 对象的使用。

需求说明：编写 JSP 程序，实现猜数功能。系统随机生成一个 1～100 之间的整数，要求用户在输入框中输入数进行游戏，根据判断的结果转向不同的页面，若没猜对，继续猜数。系统运行效果如图 2-11 至图 2-14 所示。

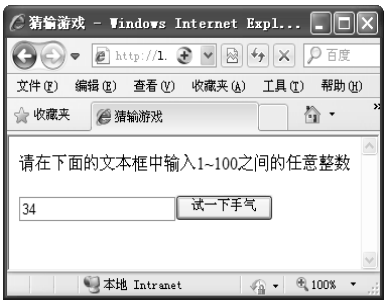

图 2-11　猜数游戏首页

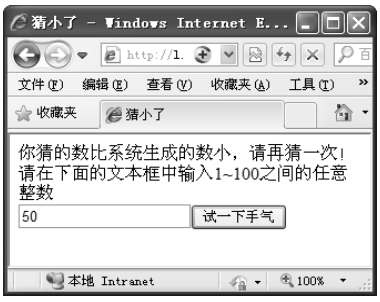

图 2-12　猜小了页面

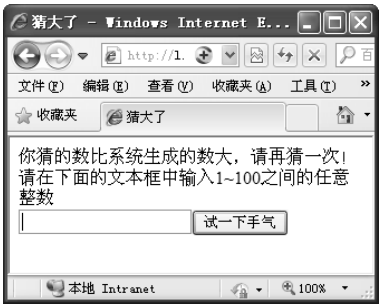

图 2-13　猜大了页面

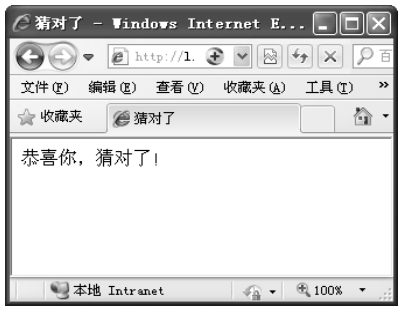

图 2-14　猜对了页面

实现思路：实现该游戏功能需要 5 个页面：

- guess.jsp 页面：效果如图 2-11 所示，在此页面中生成随机整数 (int)(Math.random()*100)+1，通过表单传递此随机整数到其他页面中。
- control.jsp 页面：获取 guess.jsp 页面输入的数，同传递过来的随机数比较，转向不同的页面。
- ok.jsp 页面：效果如图 2-14 所示。
- smaller.jsp 页面：效果如图 2-12 所示。
- larger.jsp 页面：效果如图 2-13 所示。

在 control.jsp、smaller.jsp 和 larger.jsp 这些页面间共享请求数据（随机数）。

【问题引入】

大家可能经常碰到这样的问题，比如我们在使用百度文库时，当想下载某文档时，系统会自动转入登录页面，提示用户登录后才能下载。当然，如果是已登录用户，就不会面临这样的问题了。那么，网络应用系统是如何判断用户是否已经登录过的呢？也就是说，系统如何实现对网站的访问控制呢？

2.5　任务五　实现访问控制

【实现思路】

通常情况下，用户登录后的整个访问过程都会用到用户登录信息，JSP 的 session 对象能

在一段时间内保存用户的登录信息，所以通常会在用户登录之后把用户信息保存在 session 中。而在其他页面中访问 session 中的用户信息，从而实现访问控制。

2.5.1　访问控制流程

系统进行访问控制有以下两种情形：

- 用户通过登录页面登录网站，如果该用户是已注册用户，系统会保存该用户的登录信息，并让用户进入其欲访问的页面。
- 用户直接访问网站的某个页面，系统会去查询是否保存有该用户的登录信息，如果有，则显示该页面的内容；如果没有，就转入登录页面，要求用户登录网站。

HTTP 协议是一种无状态的协议，也就是说，它不能有效地记录整个访问过程中的一些状态。JSP 提供了一套会话跟踪机制，该机制可以维持每个用户的会话信息，可以为不同的用户保存不同的数据。对 Web 开发来说，一个会话就是用户通过浏览器与服务器之间进行的一次通话，它包含浏览器与服务器之间的多次请求、响应过程。当浏览器关闭时，相应的用户会话结束。

2.5.2　使用 session 对象保存信息

JSP 中，session 内置对象用来存储有关用户会话的所有信息，一个用户对应一个 session，并且随着用户的离开 session 中的信息也会消失。访问控制就是使用 session 对象完成的。

session 对象的 setAttribute()方法用来保存用户信息，其语法格式为：

```
session.setAttribute(String key,Object value);
```

key 表示键，value 表示值，value 必须是一个 Object 类型，该方法表示以键/值的方式将一个对象的值存放到 session 中去。

【例 2.7】依据访问控制流程完善例 2.5 中的用户登录代码，用户登录成功后保存用户信息（用户名和密码）。

实现如下：

（1）编写 User 类，该类在 entity 包下，用于存放用户名及密码。

清单 2-10　entity.User.java

```
package entity;
public class User {
private String name;
private String password;
public String getName() {
        return name;
}
public void setName(String name) {
        this.name = name;
}
public String getPassword() {
        return password;
}
public void setPassword(String password) {
        this.password = password;
```

```
}
}
```

（2）编写 login.jsp 页面，为登录页面。

清单 2-11　login.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<title>用户登录</title>
</head>
<body>
请输入用户名和密码：
<form action="dologin.jsp" method="post">
用户名：<input type="text" name="uname"><br>
密&amp;nbsp;&amp;nbsp;码：<input type="password" name="pwd"><br>
<input type="submit" value="登录"><input type="reset" value="重置">
</form>
</body>
</html>
```

（3）编写 dologin.jsp 页面，为登录处理页面，用来进行登录验证，依据访问控制流程，需要将成功登录的用户信息存放在 session 中，然后再转向欲访问的 welcome.jsp 页面。

清单 2-12　dologin.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="entity.*"%>
<html>
<head>
<title>登录处理</title>
</head>
<body>
<%request.setCharacterEncoding("utf-8");
String name=request.getParameter("uname");
String pwd=request.getParameter("pwd");
User user=new User();
user.setName(name);              //将用户名封装在 user 中
user.setPassword(pwd);           //将密码封装在 user 中
if(name.equals("张三")&&pwd.equals("123")) {
    //把 user 对象存放到 session 中去，它对应的键为 loginuser
    session.setAttribute("loginuser",user);
    response.sendRedirect("welcome.jsp");
}
else response.sendRedirect("login.jsp");
 %>
</body>
</html>
```

2.5.3　使用 session 对象获取信息

session 对象的 getAttribute()方法用来获取 session 对象中存放的对象的值，其语法格式为：

(要强制转换的类型)session.getAttribute(String key);

key 表示键，该方法表示以键的方式获取 session 对象中存放的对象的值，此方法的返回值是一个 Object，必须要进行强制类型转换。

【例 2.8】依据访问控制流程完善例 2.7，要求非登录的用户不能访问欲访问的 welcome.jsp 页面。

实现：仅需修改例 2.7 中的 welcome.jsp，增加清单 2-13 中加粗的代码部分。

清单 2-13　welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="entity.*"%>
<html>
<head>
<title>登录成功</title>
</head>
<body>
<%User user=(User)session.getAttribute("loginuser");        //获取用户对象
if(user==null)response.sendRedirect("login.jsp");           //未登录的情况
 %>
欢迎<%=user.getName() %>!
</body>
</html>
```

清单 2-13 中的粗体部分即为判断 session 是否保存了用户登录信息。session 只存储已登录的用户信息。至此，访问控制已实现。现在可通过以下几步来验证访问控制的效果：

（1）直接在浏览器地址栏中输入 URL，访问 welcome.jsp 页面。

（2）通过登录页面进入 welcome.jsp 页面。

（3）重新开启一个浏览器窗口，直接访问 welcome.jsp 页面。

2.5.4　从 session 中移除指定的对象

对于存储在 session 会话中的对象，如果想将其从 session 会话中移除，可以使用 session 对象的 removeAttribute()方法，其格式如下：

session.removeAttribute(String key);

key 表示键，该方法表示以键的方式移除 session 对象中存放的对象，一般用在注销登录功能中。

2.5.5　实训　为留言管理系统增加访问控制

训练要点：session 对象的使用。

需求说明：在留言管理系统中为后台管理页面增加访问控制。

实现思路及关键代码：编写 doLogin.jsp 页面，按照访问控制流程进行登录处理，使用

session 保存成功登录的用户对象，再转向后台管理页面（admin.jsp），代码参考例 2.7；在后台管理页面获取 session 中的用户对象并判断，若为空则进行登录，否则显示后台管理页面内容，代码参考例 2.8。

【问题引入】

在网站上经常可以见到网页计数器，用来统计网页被用户访问的次数，我们也来给留言管理系统增加网页计数器功能，该如何实现呢？

2.6 任务六 制作网页计数器

【实现思路】

在项目开发中经常使用 application 对象实现网页计数器，application 对象是 JSP 的内置对象，它类似于系统的全局变量，用于实现用户之间的数据共享。

2.6.1 application 对象

application 对象用于保存所有应用程序中的公有数据。它在服务器启动时自动创建，在服务器停止时销毁。与 session 对象不同的是，在 JSP 服务器运行时，仅有一个 application 对象，当 application 对象没有被销毁时，所有用户都可以共享该 application 对象。与 session 对象相同的是，application 对象也有 setAttribute 和 getAttribute 方法。

2.6.2 应用 application 对象实现网页计数器

每次访问页面时，先获取 application 对象中保存的访问次数。如果没有获取到，表示页面被第一次访问，将数字 1 保存到 application 对象中；如果获取到了，则将访问次数增 1，再次存储到 application 对象中，具体代码如清单 2-14 所示。

清单 2-14 application.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<title>网页计数器</title>
</head>
<body>
<%
int num=0;        //定义一个保存访问次数的变量
if(application.getAttribute("number")==null){//当用户第一次访问时
num=1;
}
else {
num=Integer.parseInt(application.getAttribute("number").toString());
```

```
num=num+1;     //访问次数加 1
}
out.println("该页面已被浏览了"+num+"次！");
application.setAttribute("number",num);
%>
</body>
</html>
```

模块二小结

1．在动态网页的开发中，HTML 表单是与用户交互信息的主要手段。

2．使用 JSP 处理表单请求时，一般遵循这样一种模式：

（1）用户通过表单控件输入并提交信息。

（2）JSP 页面获取表单数据，进行逻辑处理。

（3）JSP 页面根据处理结果转向不同的结果页面。

3．所谓内置对象就是由 Web 容器加载的一组类的实例，它不像一般的 Java 对象那样用 new 去获取实例，而是可以直接在 JSP 页面中使用的对象。

4．常用的内置对象，如表 2-4 所示。

表 2-4　常用内置对象

名称	说明	常用方法
out	用于向客户端输出数据	println、print
request	用于客户端的请求处理	getParameter、getParameterValues、setCharacterEncoding、getRequestDispatcher
response	用于响应客户请求并向客户端输出信息	setCharacterEncoding、sendRedirect
session	用来存储有关用户会话的所有信息	setAttribute、getAttribute、removeAttribute、invalidate
application	类似于系统的全局变量，用于实现用户之间的数据共享	setAttribute、getAttribute

5．访问控制流程，如图 2-15 和图 2-16 所示。

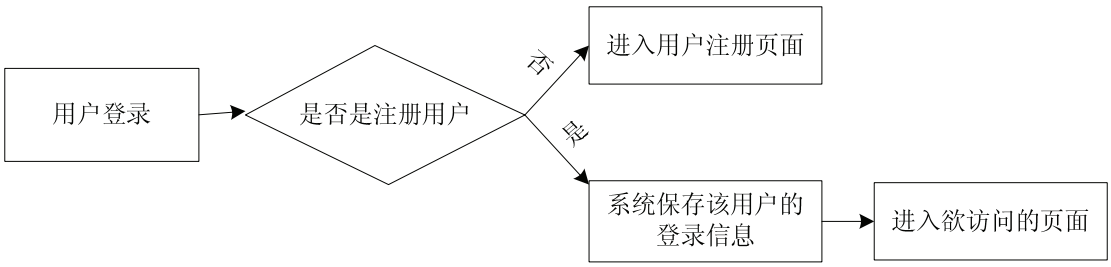

图 2-15　访问控制流程（1）

图 2-16　访问控制流程（2）

就 Web 开发来说，一个会话就是用户通过浏览器与服务器之间的一次通话，它包含浏览器与服务器之间的多层请求、响应过程。

习题二

一、填空题

1．JSP 中常用的内置对象有 request 对象、response 对象、session 对象、application 对象、out 对象等，其中_____对象的生命周期为用户访问过程，_____对象的生命周期为站点运行的全过程，_____对象包含了一次请求中的所有信息，_____对象包含了响应请求的所有信息，_____对象用于输出其他对象信息。

2．客户端通过表单 Form 向服务器提交数据有两种方法：_____和_____。

3．通过调用 request 对象的 getParameter 方法返回的数据类型为_____，通过调用 session 对象的 getAttribute 方法返回的数据类型为_____，通过调用 application 对象的 getAttribute 方法返回的数据类型为_____。

4．page 指令的_____属性指明想要引入的包和类。

5．_____指令可用于包含另一个文件。

二、选择题

1．在 JSP 页面中，下列（　　）表达式语句可以获取页面请求中名字为 title 的文本框的内容。

A．<%=request.getParameter("title")%>

B．<%=request.getAttribute("title")%>

C．<%=request.getParameterValues("title")%>

D．<%=request.getParameters("title")%>

2．在用户登录的 JSP 页面上包含如下代码所示的表单，用户希望提交表单时在地址栏中不显示提交的信息，则应该在下划线处填写的代码是（　　）。

```
<form action="loginAction.jsp" name="loginForm" method="_____">
    用户名：<input type="text" name="name"/><br>
```

```
    密码：<input type="password" name="pwd"/>
    <input type="submit" value="登录"/>
</form>
```

A．get

B．post

C．不填写任何内容

D．以上选项均可

3．在 JSP 中，要在 page 指令中设置使用的脚本语言是 Java，且导入了 java.io 和 java.util 包，下列语句中正确的是（　　）。

A．<%@ page language="java" import="java.io.*,java.util.*"%>

B．<%@ page language="java" import="java.io, java.util "%>

C．<%@ page language="java" import="java.io" import="java.util"%>

D．<%@ page language="java"%>

<%@ page import="java.io.*,java.util.*"%>

4．试图运行如下 JSP 代码，以下说法正确的是（　　）。

```
<html>
<%
String str="hello ACCP";
session.setAttribute("title",str);
String getStr=session.getAttribute("title");
out.println(getStr);
%>
</html>
```

A．运行成功，页面上输出 hello ACCP

B．运行成功，页面上输出 title

C．代码行 session.setAttribute("title",str);有错误，无法运行

D．代码行 String getStr=session.getAttribute("title");有错误，无法运行

5．启动 IE 窗口运行如下 JSP，如果连续刷新 5 次，输出的结果是 X，紧接着重新启动一个新的 IE 窗口运行该 JSP，连续刷新 3 次，输出的结果是 Y，X 和 Y 的值分别是（　　）。

```
<%@ page contentType="text/html;charset=GBK"%>
<html>
<%
Integer cnt=(Integer)application.getAttribute("hitcount ") ;
if(cnt= =null){
   cnt=new Integer(1);
}else{
    cnt=new Integer(cnt.intValue()+1);
}
application.setAttribute("hitcount",cnt);
%>
<%=cnt%>
</html>
```

A．5、8

B．5、3

C．1、2

D．1、1

6．JSP 提供了一个可以在多个请求之间持续有效的内置对象（　　），该对象与浏览器一一对应。

A．request

B．response

C．session

D．application

7．在 helloapp 应用中有一个 hello.jsp，它的文件路径为 WebRoot/hello/hello.jsp，那么在浏览器端访问 hello.jsp 的 URL 是（　　）。

A．http://localhost:8080/hello.jsp

B．http://localhost:8080/helloapp/hello.jsp

C．http://localhost:8080/helloapp/hello/hello.jsp

D．http://localhost:8080/hello/hello.jsp

8．从 HTTP 请求中获得请求参数，应该调用（　　）。

A．request 对象的 getAttribute()方法

B．request 对象的 getParameter()方法

C．session 对象的 getAttribute()方法

D．session 对象的 getParameter()方法

9．在 helloapp 应用中 hello.jsp 和 welcome.jsp 在同一目录中，index.jsp 在应用的根目录中，index.jsp 使用下面的代码可以跳转到 hello.jsp 页面：request.getRequestDispatcher("/hello.jsp").forward(request,response);，用下列方式中的（　　）代替上述代码后可以跳转到 welcome.jsp 页面。

A．response.sendRedirect("/helloapp/welcome.jsp");

B．response.sendRedirect("helloapp/welcome.jsp");

C．response.sendRedirect("welcome.jsp");

D．response.sendRedirect("/welcome.jsp");

10．下面关于 JSP 作用域对象的说法错误的是（　　）。

A．request 对象可以得到请求中的参数

B．session 对象可以保存用户信息

C．application 对象可以被多个应用共享

D．作用域范围从小到大是 request、session、application

三、编程题

1．编写静态页面用于输入年份，编写 JSP 用于判断该年是否是闰年。

2．编写一个 JSP 页面，要求提供一个包含各职业名称的下拉列表框，让用户选择其职业，提交后，判断用户的职业是否是医生，如果是，则跳转进入一个欢迎页面；如果不是，则在页面上显示该用户的职业。

3．编写一个 JSP 页面 luncknum.jsp，产生 0～9 之间的随机数作为用户幸运数字，将其保存到会话中，并重定向到另一个页面 shownum.jsp 中，在该页面中将用户的幸运数字显示出来。提示：Math 类的 random()方法生成 0.0～1.0 之间的随机数。