



用JSP脚本元素调用Java 代码

JSP, Servlet, & Struts Training Courses: <http://courses.coreservlets.com>
Available in US, China, Taiwan, HK, and Worldwide

JSP and Servlet Books from Sun Press: <http://www.coreservlets.com>
*Available in English, Chinese (simplified and traditional script),
and 12 other languages*

2

议程

- 静态和动态文本
- 动态代码和好的JSP设计
- JSP表达式
- servlet和JSP页面对类似任务的处理
- JSP scriptlet
- JSP声明
- 预定义变量
- 表达式、scriptlet和声明之间的对比

3

不同JSP构造的应用

简单应用

- 脚本元素直接调用servlet代码
- 脚本元素间接调用servlet代码（通过实用工具类）
- bean
- servlet/JSP的组合(MVC)
- MVC连同JSP表达式语言
- 定制标签

复杂应用



4

JSP/servlet/Struts/JSF training: <http://www.coreservlets.com>

设计策略：限制JSP页面内的Java代码

- 有两种选择：
 - 将25行Java代码直接放在JSP页面中
 - 将这25行代码放在单独的Java类中，在JSP页面中用一行代码调用它
- 为什么第二种选择更好？
 - 开发。我们可以在Java开发环境（编辑器或IDE）中编写单独的类，而非在HTML环境中。
 - 调试。如果代码中含有语法错误，在编译时就能够直接看到。简单的打印语句可以直接看到结果。
 - 测试。我们可以编写测试过程，每次更改后都通过循环进行上万次的测试。
 - 重用。多个页面都可以使用同一个类。

5

JSP/servlet/Struts/JSF training: <http://www.coreservlets.com>

基本语法

- **HTML文本**
 - `<H1>Blah</H1>`
 - 不加更改地传递给客户。实际上是由类似于下面的servlet代码生成：
 - `out.print("<H1>Blah</H1>");`
- **HTML注释**
 - `<!-- 注释内容 -->`
 - 和其他HTML一样：直接传递给客户
- **JSP注释**
 - `<%-- 注释内容 --%>`
 - 并不发送给客户
- **要输出`<%`，使用`<\%`**

6

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

脚本元素的类型

- **表达式**
 - 格式：`<%= expression %>`
 - 求值并插入到servlet的输出中。
也就是会产生类似于`out.print(expression)`的语句
- **scriptlet**
 - 格式：`<% code %>`
 - 逐字地插入到servlet的`_jspService`方法中(由service调用)
- **声明**
 - 格式：`<%! code %>`
 - 逐字地插入到servlet类的定义体中，不在任何方法之内

7

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

JSP表达式

- 格式

- `<%= Java Expression %>`

- 结果

- 表达式被求值，转换成字符串，将按照它在JSP页面中出现的位置插入到HTML页面中。

- 即，表达式在out.print内放置在That is, expression placed in _jspService inside out.print

- 示例

- 当前时间：`<%= new java.util.Date() %>`

- 您的主机名：`<%= request.getRemoteHost() %>`

- XML兼容的语法

- `<jsp:expression>Java Expression</jsp:expression>`

- 在单个页面中不能混合使用不同的版本。如果使用jsp:expression就必须整个页面都使用XML。

8

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

JSP/servlet的对应

- 原始的JSP

```
<H1>A Random Number</H1>
```

```
<%= Math.random() %>
```

- 所产生的具有代表性的servlet代码

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    HttpSession session = request.getSession(true);
    JspWriter out = response.getWriter();
    out.println("<H1>A Random Number</H1>");
    out.println(Math.random());
    ...
}
```

9

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

JSP表达式：示例

```
...<BODY>
<H2>JSP Expressions</H2>
<UL>
  <LI>Current time: <%= new java.util.Date() %>
  <LI>Server: <%= application.getServerInfo() %>
  <LI>Session ID: <%= session.getId() %>
  <LI>The <CODE>testParam</CODE> form parameter:
      <%= request.getParameter("testParam") %>
</UL>
</BODY></HTML>
```



10

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

预定义变量

- **request**
 - HttpServletRequest (service/doGet的第一个参数)
- **response**
 - HttpServletResponse (service/doGet的第二个参数)
- **out**
 - 用来将输出发送到客户程序的Writer (JspWriter带缓冲的版本)
- **session**
 - 与请求关联的HttpSession (除非用page指令的session属性将之禁止)
- **application**
 - ServletContext (用于数据共享) ， 可通过
getServletContext()获取

11

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

Servlet和JSP的比较：读取三个参数 (Servlet)

```
public class ThreeParams extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        ...
        out.println(docType +
            "<HTML>\n" +
            "<HEAD><TITLE>"+title + "</TITLE></HEAD>\n" +
            "<BODY BGCOLOR=\"#FDF5E6\">\n" +
            "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
            "<UL>\n" +
            "  <LI><B>param1</B>: "
            + request.getParameter("param1") + "\n" +
            "  <LI><B>param2</B>: "
            + request.getParameter("param2") + "\n" +
            "  <LI><B>param3</B>: "
            + request.getParameter("param3") + "\n" +
            "</UL>\n" +
            "</BODY></HTML>");
    }
}
```

12

JSP/Servlet/Struts/JSP training: <http://www.coreservlets.com>

读取三个参数 (Servlet)：结果



13

JSP/Servlet/Struts/JSP training: <http://www.coreservlets.com>

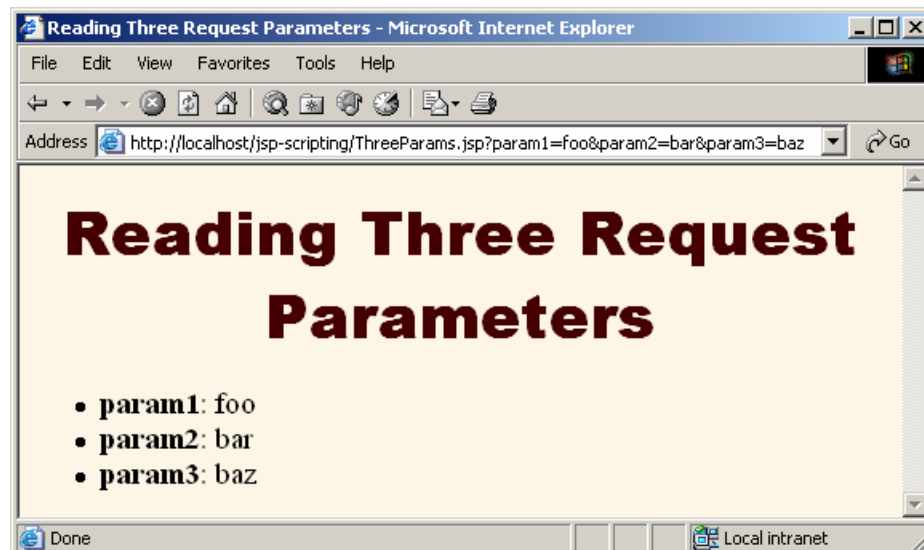
Servlet和JSP的比较：读取三个参数 (JSP)

```
<!DOCTYPE ...>
<HTML>
<HEAD>
<TITLE>Reading Three Request Parameters</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>Reading Three Request Parameters</H1>
<UL>
  <LI><B>param1</B>:
      <%= request.getParameter("param1") %>
  <LI><B>param2</B>:
      <%= request.getParameter("param2") %>
  <LI><B>param3</B>:
      <%= request.getParameter("param3") %>
</UL>
</BODY></HTML>
```

14

JSP/Servlet/Struts/JSP training: <http://www.coreservlets.com>

读取三个参数 (JSP) : 结果



15

JSP/Servlet/Struts/JSP training: <http://www.coreservlets.com>

JSP scriptlet

- 格式
 - `<% Java Code %>`
- 结果
 - 代码被逐字地插入到servlet的_jspService方法中
- 示例
 - `<%
String queryData = request.getQueryString();
out.println("Attached GET data: " + queryData);
%>`
 - `<% response.setContentType("text/plain"); %>`
- XML兼容的语法
 - `<jsp:scriptlet>Java Code</jsp:scriptlet>`

16

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

JSP/servlet之间的对应关系

- 原始的 JSP

```
<H2>foo</H2>  
<%= bar() %>  
<% baz(); %>
```
- 所产生的具有代表性的servlet代码

```
public void _jspService(HttpServletRequest request,  
                        HttpServletResponse response)  
    throws ServletException, IOException {  
    response.setContentType("text/html");  
    HttpSession session = request.getSession();  
    JspWriter out = response.getWriter();  
    out.println("<H2>foo</H2>");  
    out.println(bar());  
    baz();  
    ...  
}
```

17

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

JSP scriptlet : 示例

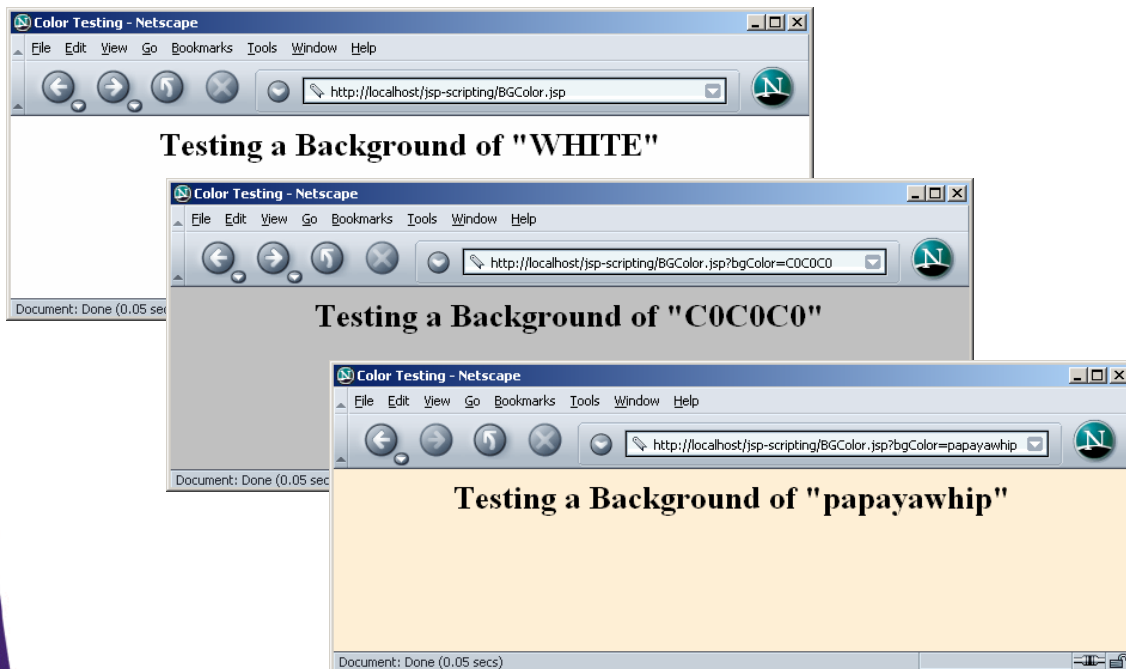
- 假定您希望能够让最终用户定制页面的背景色
 - 下面的代码有什么问题?

```
<BODY BGCOLOR=  
    "<%= request.getParameter("bgColor") %>">
```

JSP scriptlet : 示例

```
<!DOCTYPE ...>  
<HTML>  
<HEAD>  
    <TITLE>Color Testing</TITLE>  
</HEAD>  
<%  
String bgColor = request.getParameter("bgColor");  
if ((bgColor == null) ||  
    (bgColor.trim().equals("")) ) {  
    bgColor = "WHITE";  
}  
>%  
<BODY BGCOLOR="<%= bgColor %>">  
<H2 ALIGN="CENTER">Testing a Background of  
"<%= bgColor %>"</H2>  
</BODY></HTML>
```

JSP scriptlet : 结果



20

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

使用scriptlet将JSP文件的某些部分条件化

• 要点

- scriptlet会被严格按照所写的样子插入到servlet中
- 不需要是完整的Java表达式
- 但完整的表达式常常更清楚且易于维护

• 示例

```
- <% if (Math.random() < 0.5) { %>  
  Have a <B>nice</B> day!  
  <% } else { %>  
  Have a <B>lousy</B> day!  
  <% } %>
```

• 具有代表性的结果

```
- if (Math.random() < 0.5) {  
  out.println("Have a <B>nice</B> day!");  
} else {  
  out.println("Have a <B>lousy</B> day!");  
}
```

21

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

JSP声明

- 格式
 - `<%! Java Code %>`
- 结果
 - 代码被逐字地插入到servlet类的定义中，在任何现有方法以外
- 示例
 - `<%! private int someField = 5; %>`
 - `<%! private void someMethod(...) { ... } %>`
- 设计上的考虑
 - 字段当然有用，但对于方法而言，通常在单独的Java类中定义更好一些。
- XML兼容的语法
 - `<jsp:declaration>Java Code</jsp:declaration>`

22

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

JSP/servlet之间的对应关系

- 原始的JSP

```
<H1>Some Heading</H1>
<%!
    private String randomHeading() {
        return("<H2>" + Math.random() +
            "</H2>");
    }
%>
<%= randomHeading() %>
```

- (也可以使randomHeading成为单独Java类的静态方法)

23

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

JSP/servlet之间的对应关系

- 生成的servlet代码

```
public class xxxx implements HttpJspPage {
    private String randomHeading() {
        return("<H2>" + Math.random() + "</H2>");
    }

    public void _jspService(HttpServletRequest request,
                            HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        HttpSession session = request.getSession(true);
        JspWriter out = response.getWriter();
        out.println("<H1>Some Heading</H1>");
        out.println(randomHeading());
        ...
    }
}
```

24

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

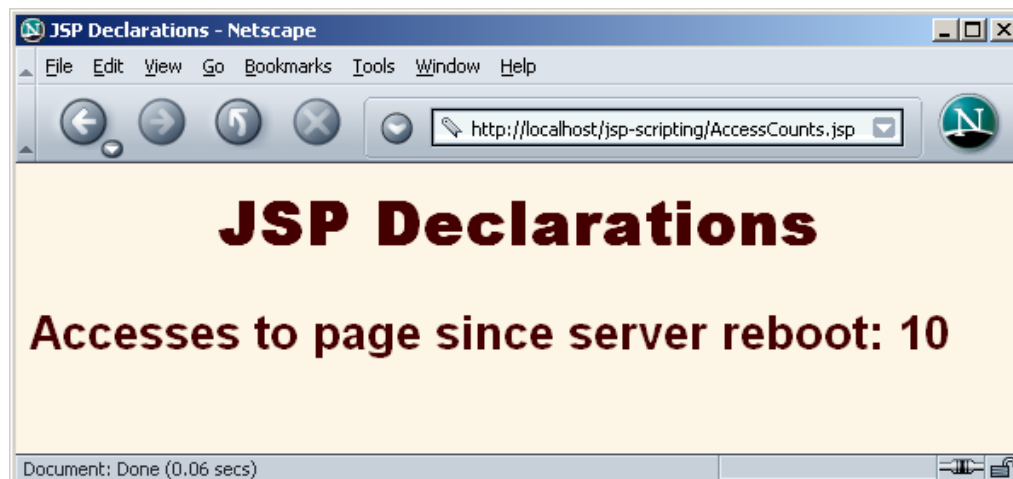
JSP声明：示例

```
<!DOCTYPE ...>
<HTML>
<HEAD>
<TITLE>JSP Declarations</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>JSP Declarations</H1>
<%! private int accessCount = 0; %>
<H2>Accesses to page since server reboot:
<%= ++accessCount %></H2>
</BODY></HTML>
```

25

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

JSP声明：结果



26

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

JSP声明：jspInit和jspDestroy

- JSP页面，如同常规servlet，有时需要用到init和destroy
- 问题在于：根据JSP页面构建的servlet有可能已经使用了init和destroy
 - 覆盖它们会引发问题。
 - 因此，使用JSP声明来声明init或destroy是不合法的。
- 解决方案：使用**jspInit**和**jspDestroy**。
 - 根据JSP页面生成的servlet一定会在init和destroy中调用这些方法，而标准版的jspInit和jspDestroy是空的（仅为占位符，由用户覆盖）。

27

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

JSP声明和预定义变量

- 问题
 - 预定义变量(request, response, out, session, 等)是_jspService方法中的局部变量。因而，由JSP声明定义的方法或辅助类中的方法都不能使用它们。怎么处理这种情况呢？
- 解决方案：将它们作为参数传递，例如

```
<%!  
private void someMethod(HttpSession s) {  
    doSomethingWith(s);  
}  
%>  
<% someMethod(session); %>
```
- 要注意：JspWriter的println方法抛出IOException异常
 - 使用println的方法要使用“throws IOException”

28

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

表达式、scriptlet和声明的对比

- 任务 1
 - 输出由从1到10之间的5个随机整数构成的项目列表。
 - 由于本页的结构固定，并且对于randomInt方法，我们使用单独的辅助类，JSP表达式完全能够胜任。
- 任务 2
 - 生成1到10数据项构成的列表（随机选定），每一项为1到10之间的整数。
 - 由于列表中的数据项是动态的，所以需要JSP scriptlet。
- 任务 3
 - 在首次请求时生成一个随机数，然后向所有的用户展示这个随机数，直到服务器重新启动。
 - 实例变量（字段）是完成这种持续性的自然方式。使用JSP声明来完成这项任务。

29

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

辅助类：RanUtilities

```
package coreservlets; // Always use packages!!

/** Simple utility to generate random integers. */
public class RanUtilities {

    /** A random int from 1 to range (inclusive). */

    public static int randomInt(int range) {
        return(1 + ((int)(Math.random() * range)));
    }

    public static void main(String[] args) {
        int range = 10;
        try {
            range = Integer.parseInt(args[0]);
        } catch(Exception e) { // Array index or number format
            // Do nothing: range already has default value.
        }
        for(int i=0; i<100; i++) {
            System.out.println(randomInt(range));
        }
    }
}
```

30

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

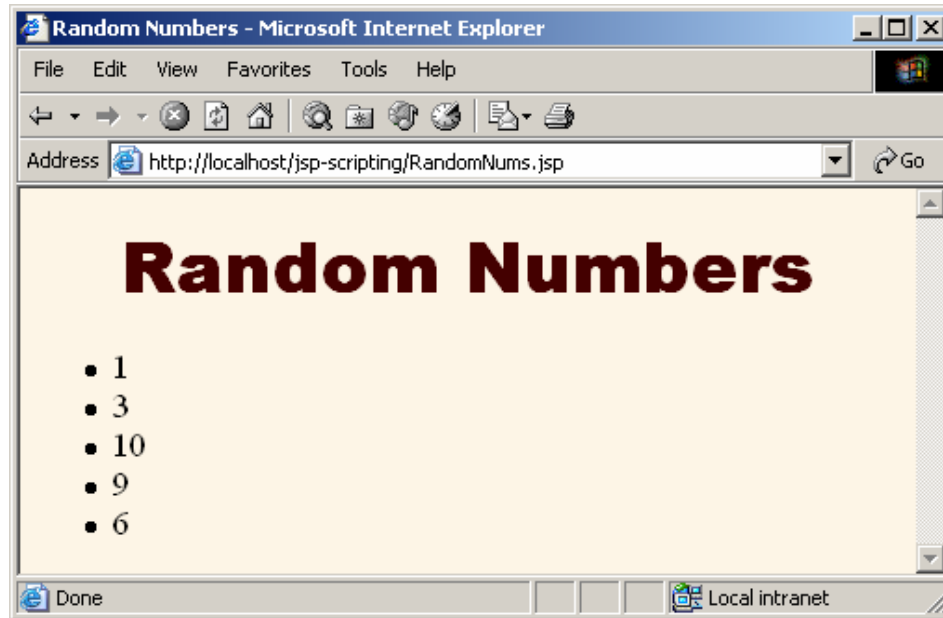
任务 1：JSP表达式（代码）

```
<!DOCTYPE ...>
<HTML>
<HEAD>
<TITLE>Random Numbers</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>Random Numbers</H1>
<UL>
  <LI><%= coreservlets.RanUtilities.randomInt(10) %>
  <LI><%= coreservlets.RanUtilities.randomInt(10) %>
  <LI><%= coreservlets.RanUtilities.randomInt(10) %>
  <LI><%= coreservlets.RanUtilities.randomInt(10) %>
  <LI><%= coreservlets.RanUtilities.randomInt(10) %>
</UL>
</BODY></HTML>
```

31

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

任务 1 : JSP表达式 (结果)



32

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

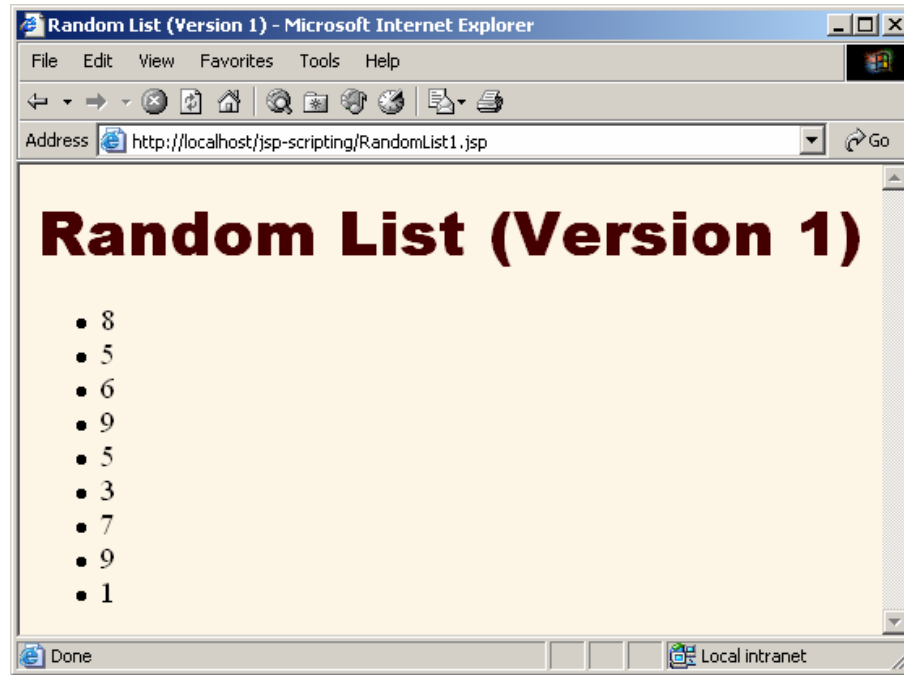
任务 2 : JSP scriptlet (代码 : 版本 1)

```
<!DOCTYPE ...>
<HTML>
<HEAD>
<TITLE>Random List (Version 1)</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>Random List (Version 1)</H1>
<UL>
<%
int numEntries = coreservlets.RanUtilities.randomInt(10);
for(int i=0; i<numEntries; i++) {
    out.println("<LI>" +
coreservlets.RanUtilities.randomInt(10));
}
%>
</UL>
</BODY></HTML>
```

33

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

任务 2 : JSP scriptlet (结果 : 版本1)



34

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

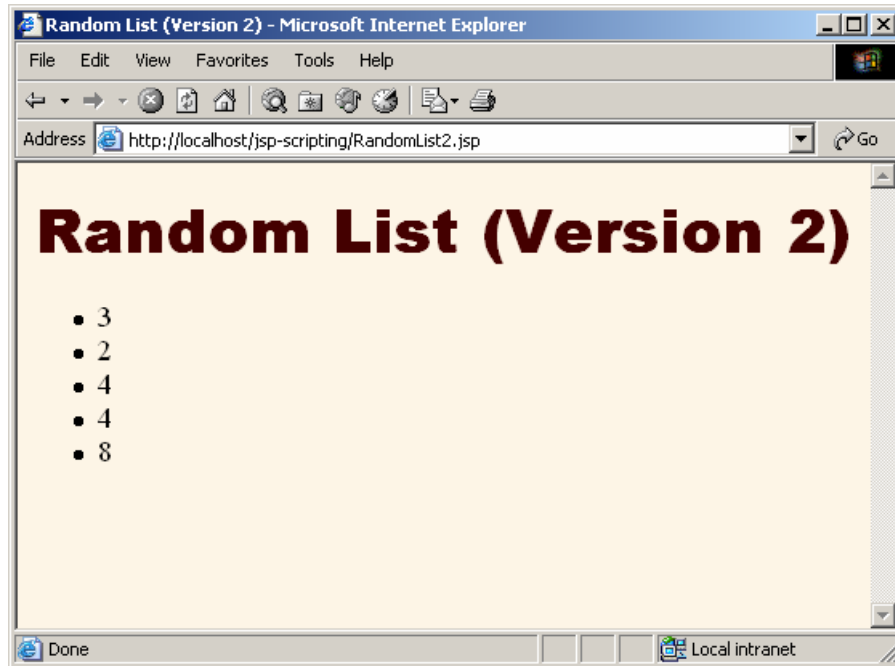
任务 2 : JSP scriptlet (代码 : 版本2)

```
<!DOCTYPE ...>
<HTML>
<HEAD>
<TITLE>Random List (Version 2)</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>Random List (Version 2)</H1>
<UL>
<%
int numEntries = coreservlets.RanUtilities.randomInt(10);
for(int i=0; i<numEntries; i++) {
%>
<LI><%= coreservlets.RanUtilities.randomInt(10) %>
<% } %>
</UL>
</BODY></HTML>
```

35

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

任务 2 : JSP scriptlet (结果 : 版本 2)



36

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

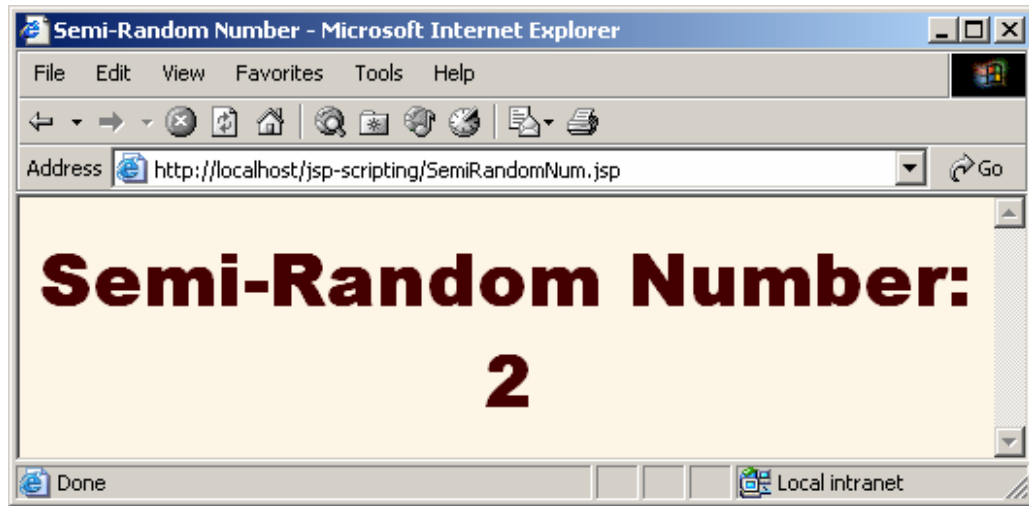
任务 3 : JSP声明 (代码)

```
<!DOCTYPE ...>
<HTML>
<HEAD>
<TITLE>Semi-Random Number</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<%!
private int randomNum =
    coreservlets.RanUtilities.randomInt(10);
%>
<H1>Semi-Random Number:<BR><%= randomNum %></H1>
</BODY>
</HTML>
```

37

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

任务 3 : JSP声明 (结果)



38

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

小结

- **JSP表达式**
 - 格式 : `<%= expression %>`
 - 封装到`out.print`中 , 并插入到`_jspService`
- **JSP scriptlet**
 - 格式 : `<% code %>`
 - 逐字地插入到`servlet`的`_jspService`方法中
- **JSP声明**
 - 格式 : `<%! code %>`
 - 逐字地插入到`servlet`类的定义体中
- **预定义变量**
 - `request`, `response`, `out`, `session`, `application`
- **限制直接放在页面中的Java代码**
 - 考虑使用辅助类、bean、servlet/JSP的组合(MVC)、JSP表达式语言、定制标签。

39

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>



问题?

JSP, Servlet, & Struts Training Courses: <http://courses.coreservlets.com>
Available in US, China, Taiwan, HK, and Worldwide

JSP and Servlet Books from Sun Press: <http://www.coreservlets.com>
*Available in English, Chinese (simplified and traditional script),
and 12 other languages*