**JSP(JavaServer Pages )是什么?**

JavaServer Pages(JSP)是一种支持动态内容开发的网页技术它可以帮助开发人员通过利用特殊的JSP标签，其中大部分以<%开始并以%>作为结束标志插入Java代码到HTML页面。

JavaServer Pages组件是一个Java servlet的类型，旨在满足Java Web应用程序用户界面的一个角色。Web开发人员编写JSP为文本文件，结合HTML或XHTML代码，XML元素，并嵌入JSP动作和命令。

使用JSP可以通过网页的形式，从数据库或其他来源的记录收集来自用户的输入，并动态地创建Web页面。

JSP标签可用于各种目的，如从数据库中检索信息或登记的用户偏好，访问JavaBeans组件，页面之间传递控制和共享请求，网页等之间的信息。

**为什么使用JSP?**

JavaServer页面往往服务于同一目的，使用通用网关接口(CGI)执行的方案。但相比CGI，JSP提供了几个优点。

* 性能更好，因为JSP允许嵌入动态元素在HTML页面中，而不是只有一个单独的CGI文件。
* JSP在请求处理之前总是经过了编译，它不同于 CGI/Perl 服务器，需要服务器在加载时都要解释，并在每次请求页面时才处理目标脚本。
* JavaServer Pages是建立在Java Servlet的API之上，所以就像Servlet，JSP也可以访问强大的企业 Java API 的所有功能，包括JDBC，JNDI，EJB，JAXP等。
* JSP页面可以结合使用servlet处理业务逻辑，通过Java servlet模板引擎所支持的模型。

最后，JSP是J2EE，企业级应用的完整平台的一个组成部分。这意味着JSP可以开发最简单应用程序，或作为最复杂的应用程序一部分。

**安装JSP开发环境**

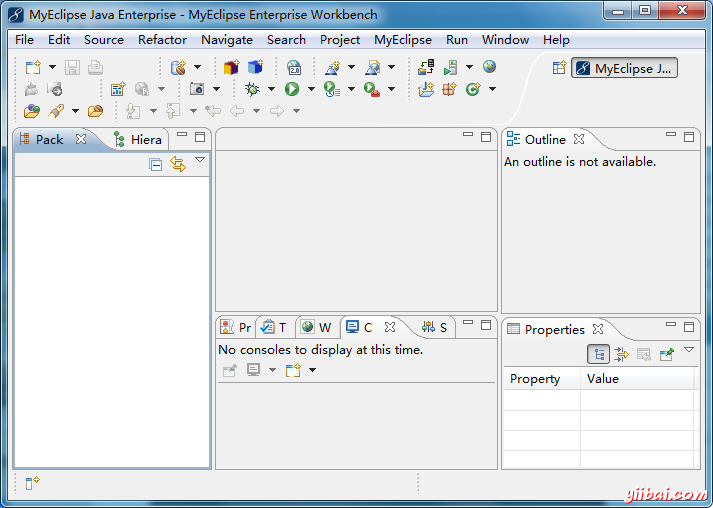
这个步骤包括下载Java软件开发工具包(SDK)和安装，并适当设置PATH环境变量。

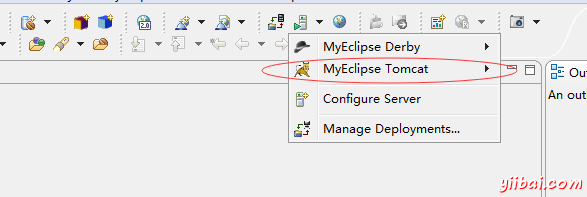
可以从Oracle的Java网站下载SDK： [Java SE 下载](http://www.oracle.com/technetwork/java/javase/downloads/index.html)

另外，如果使用集成开发环境(IDE)，如：Borland公司的JBuilder，Eclipse，IntelliJ IDEA，或Sun ONE Studio，编译并运行一个简单的程序，以确认IDE知道系统上在哪里安装了 Java 。

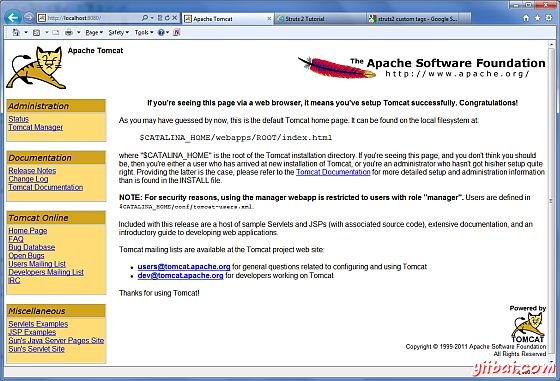
在本教程中，我们使用的是：MyEclise 8.5 试用版本，为了与本教程同步以减少出现代码或其它的意外的错误，建议在使用 MyEclipse 8.5 以同步本教程。

**安装MyEclipse开发环境**

安装MyEclipse开发环境比较简单，可以从网上搜索下载一个MyEclipse 8.5，然后安装在一个自己喜欢的位置，在教程中安装的位置是：D:\Program Files\MyEclipse 8.5，双击“myeclipse.exe”(或桌面的图标)，启动，启动完成后画面如下：  
  


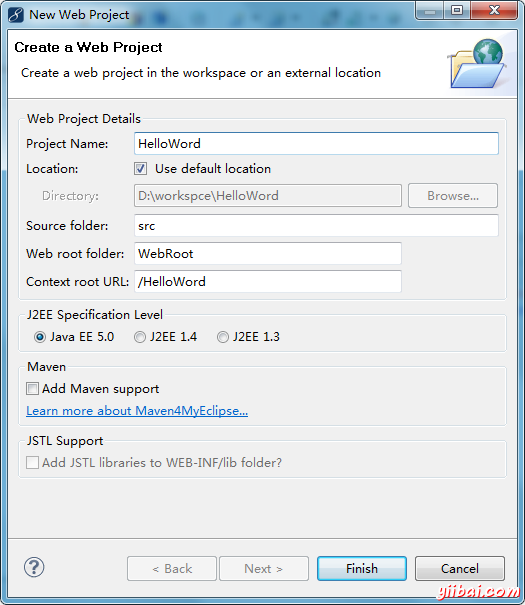
第二步，启动 MyEclipse 中的Tomcat：  
  


成功启动后，Tomcat 默认的Web应用程序将可以通过：<http://localhost:8080/> 访问。如果一切正常，那么它应该显示以下结果：

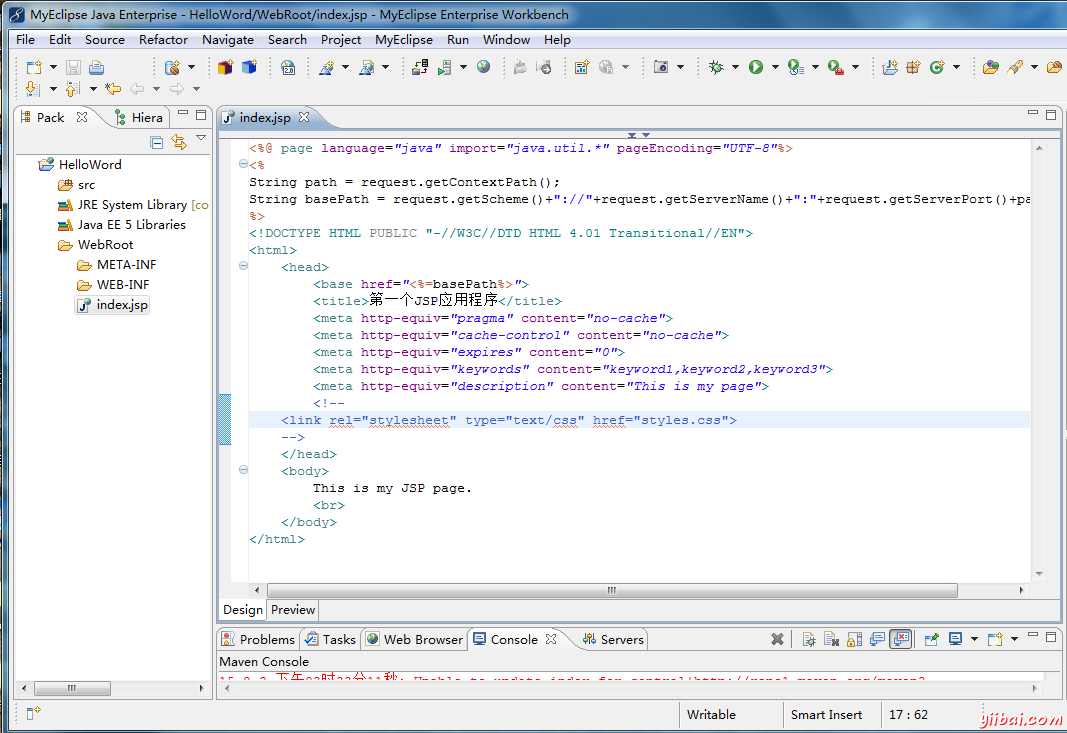


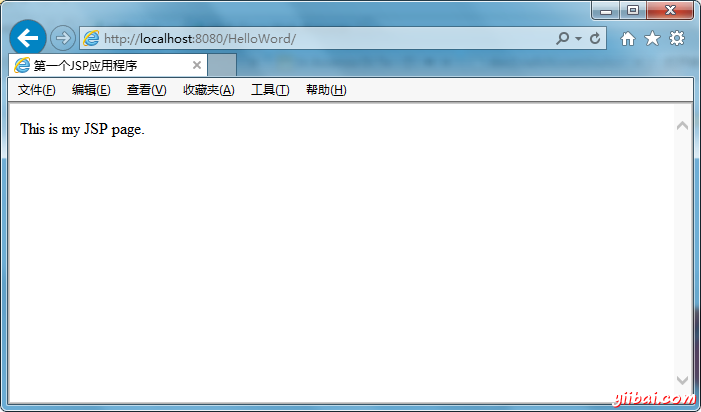
这里包括，Tomcat网站有关配置和运行Tomcat文档信息，更多的信息可访问：[http://tomcat.apache.org](http://tomcat.apache.org/)

**JSP第一个应用:**

第一步，点击右上角中的 “File"->"New"->"Web Project" 创建一个工程为：“HelloWord"，如下：  
  


第二步：点击“Finish"，然后完成工程创建，整个工程结构如下图所示：



运行工程，右键点击工程名称 ”HelloWord"，选择“Run As”->“MyEclipse Server Application" 或在浏览器中输入：[http://localhost](http://localhost/):8080/HelloWord/ 运行结果如下图所示：  
  


到这里，JSP的开发运行环境已经建立起来了。接下来可以进一步开发学习JSP的其它知识。

**JSP处理：**

下面的步骤是用来说明Web服务器是如何使用JSP创建网页：

* 对于一个正常的页面，浏览器发送一个HTTP请求到万维网服务器。
* 网站服务器识别HTTP请求是一个JSP页面，并将其转发给JSP引擎。这是通过使用URL或JSP页面，而这个页面使用 .jsp 后缀，而不是 .html 后缀。
* JSP引擎从磁盘加载JSP页面，并将其转换成servlet的内容。这种转换是很简单的，所有的模板文本被转换给println()语句，所有JSP元素被转换为实现该页面相应动态行为的Java代码。
* JSP引擎编译成servlet的一个可执行类并转发原始请求到servlet引擎。
* 一部分Web服务器调用servlet引擎加载Servlet类并执行它。在执行期间，Servlet产生HTML的格式输出，其servlet引擎传递到web服务器响应HTTP请求。
* Web服务器响应HTTP转发静态的HTML内容到浏览器。
* 最后的 web 浏览器处理HTTP响应动态生成的HTML页面完全就像一个静态页面内容。

下面的图显示所有上述的步骤：

**脚本**

scriptlet可以包含任意数量的JAVA语言的语句，变量或方法声明，或者是在页面的脚本语言有效表达式。

以下是Scriptlet中的语法：

<% code fragment %>

可以编写XML代码相当于上述语法如下：

<jsp:scriptlet>

code fragment

</jsp:scriptlet>

写的任何文本，HTML标签或JSP元素一定在脚本之外。以下是简单第一个JSP的例子：

<html>

<head><title>Hello World - By yiibai.com</title></head>

<body>

Hello World!<br/>

<%

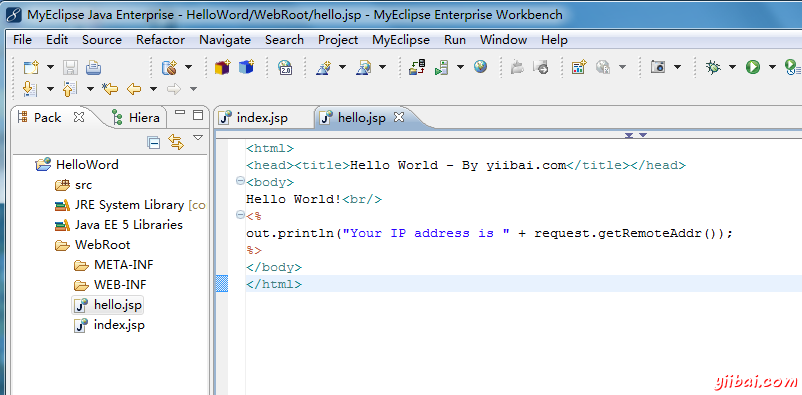
out.println("Your IP address is " + request.getRemoteAddr());

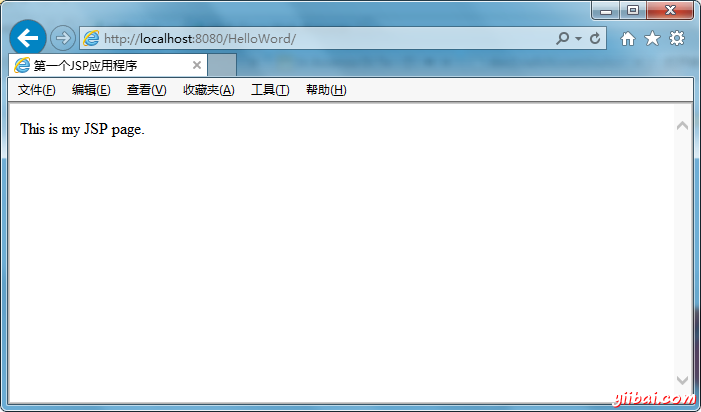
%>

</body>

</html>

现在我们来创建一个JSP文件页面，这个文件名称为：hello.jsp，让我们保存上面的代码到hello.jsp 中，整个完整的代码如下：



启动Tomcat，在浏览器地址栏中输入：http://localhost:8080/HelloWord/hello.jsp，这将产生以下结果：  
  


**JSP 声明**

声明一个或多个变量，或者方法可以在Java代码后面的JSP文件中使用。 在JSP文件中使用变量或方法它们之前，必须先声明它们。

以下是JSP声明的语法：

<%! declaration; [ declaration; ]+ ... %>

可以使用XML编写相当于上述语法如下：

<jsp:declaration>

code fragment

</jsp:declaration>

下面是简单的例子JSP声明：

<%! int i = 0; %>

<%! int a, b, c; %>

<%! Circle a = new Circle(2.0); %>

**JSP表达式**

JSP表达式元素包含计算，转换为字符串，并插入出现在JSP文件的脚本语言表达式。

因为一个表达式的值被转换为字符串，可以在文本一行内使用表达式，不管它是否被标记使用在HTML，JSP文件中。

表达元素可以包含任何Java语言规范有效的表达式，但是不能使用一个分号来结束表达式。

下面是JSP表达式的语法：

<%= expression %>

可以使用XML编写相当于上述语法如下：

<jsp:expression>

expression

</jsp:expression>

这里创建一个新的工程：jsp-base，并在index.jsp文件中写入以下代码，下面是简单的例子JSP表达式：

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>

<html>

<head>

<title>JSP表达式 - by yiibai.com</title>

</head>

<body>

<p>

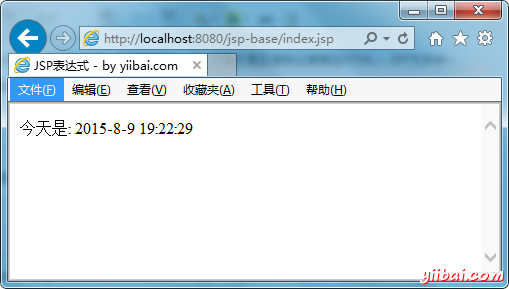
今天是:<%=(new java.util.Date()).toLocaleString()%>

</p>

</body>

</html>

打开浏览器，输入网址：http://localhost:8080/jsp-base/index.jsp，产生以下结果：



**JSP注释**

JSP注释标记的文字或语句都会被JSP容器忽略。当想要隐藏或“注释掉”JSP页面的一部分，JSP注释是很有用的。

以下是JSP注释语法：

<%-- This is JSP comment --%>

创建一个comment.jsp 并编写入下面的代码，以测试JSP注释：

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>

<html>

<head><title>注释 - 示例</title></head>

<body>

<h2>A Test of Comments</h2>

<%-- This comment will not be visible in the page source --%>

</body>

</html>

[http://localhost](http://localhost/):8080/jsp-base/comment.jsp

* 还有少数特殊的结构可以使用一些情况，插入注释或字符，将被特殊处理。这里有一个整理汇总：

|  |  |
| --- | --- |
| **语法** | **目的** |
| <%-- comment --%> | JSP注释，它将被JSP引擎忽略 |
| <!-- comment --> | HTML注释，它将被浏览器忽略 |
| <\% | 表示静态<%的字面量 |
| %\> | 表示静态%>的字面量 |
| \' | 在使用单引号在属性中的单引号 |
| \" | 双引号在属性使用双引号 |

* **JSP 指令**
* JSP指令影响的servlet类的整体结构。它通常有以下形式：
* <%@ directive attribute="value" %>
* 有三种类型的指令标记：

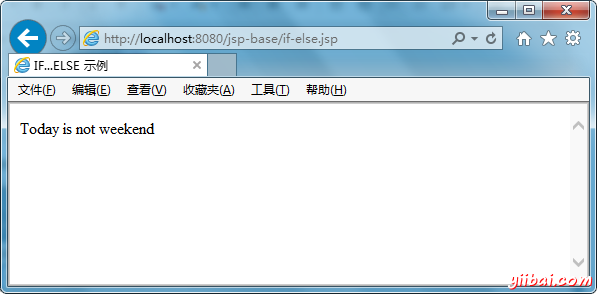
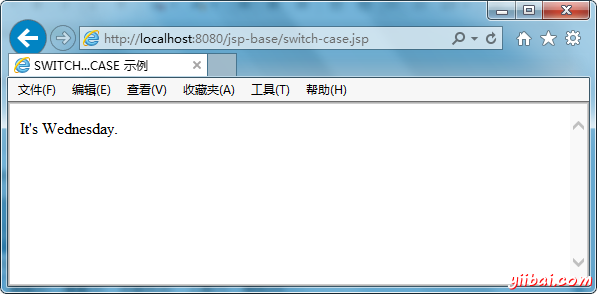
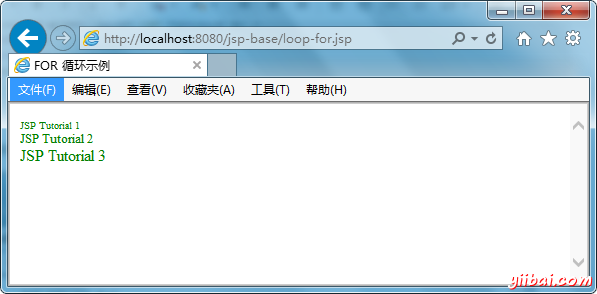
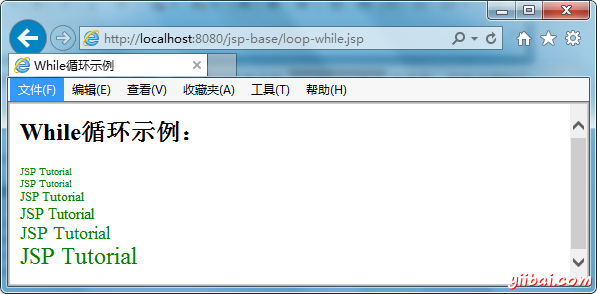
|  |  |
| --- | --- |
| **指令** | **描述** |
| <%@ page ... %> | 定义页面依赖属性，例如脚本语言，错误页面和缓冲的要求 |
| <%@ include ... %> | 包括在转换阶段的文件 |
| <%@ taglib ... %> | 声明了一个标签库，包含自定义动作，用在页面中 |

* **JSP 动作**
* JSP动作使用XML语法结构来控制Servlet引擎的行为。可以动态地插入文件，重用JavaBeans组件，用户转发到另一个页面，或为Java插件生成HTML。
* 只有一个用于动作元素的语法，因为它符合XML标准：
* <jsp:action\_name attribute="value" />
* 动作元素基本上都是预先定义函数并有以下可用的JSP操作：

|  |  |
| --- | --- |
| **语法** | **目的** |
| jsp:include | 包括页面被一次请求的文件 |
| jsp:include | 包括页面被一次请求的文件 |
| jsp:useBean | 查找或实例化一个JavaBean |
| jsp:setProperty | 设置一个JavaBean的属性 |
| jsp:getProperty | 插入一个JavaBean的属性到输出 |
| jsp:forward | 转发请求到一个新的页面 |
| jsp:plugin | 生成特定浏览器的代码，使对象或嵌入标签Java插件 |
| jsp:element | 定义XML元素动态 |
| jsp:attribute | 定义动态定义XML元素的属性 |
| jsp:body | 定义动态定义的XML元素主体 |
| jsp:text | 用于编写模板文本在JSP页面和文档 |

* 我们将在单独的章节解释JSP动作 [JSP动作](http://www.yiibai.com/jsp/jsp_actions.html)
* **JSP 隐式对象：**
* JSP支持九种自动定义的变量，这也被称为隐式对象。这些变量是：

|  |  |
| --- | --- |
| **对象** | **描述** |
| request | 这是与请求相关联的HttpServletRequest对象 |
| response | 这是用于响应客户端相关联的HttpServletResponse对象 |
| out | 这是用于将输出发送给客户端的PrintWriter对象 |
| session | 这是与请求相关联的HttpSession对象 |
| application | 这是应用程序上下文关联的ServletContext对象 |
| config | 这是与页面关联的ServletConfig对象 |
| pageContext | 这封装采用类似更高的性能JspWriters服务器特定的功能 |
| page | 这是一个简单的代名词，是用来调用由转换servlet类中定义的方法 |
| Exception | Exception对象允许例外的数据由JSP指定访问 |

* 我们将在单独一章介绍JSP隐式对象 [**JSP隐式对象**](http://www.yiibai.com/jsp/jsp_implicit_objects.html)
* **控制流语句**
* JSP提供了Java的全部功能可以嵌入在Web应用程序。可以使用Java的所有API和构建块在JSP编程，包括决策语句，循环等。
* **决策声明**
* 在if ... else块开头时就像一个普通的Scriptlet，但 Scriptlet 每一行是封闭的，包括scriptlet标记之间的HTML文本。创建一个JSP文件为：if-else.jsp，并写入以下代码：
* <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>
* <%! int day = 3; %>
* <html>
* <head><title>IF...ELSE 示例</title></head>
* <body>
* <% if (day == 1 | day == 7) { %>
* <p> Today is weekend</p>
* <% } else { %>
* <p> Today is not weekend</p>
* <% } %>
* </body>
* </html>
* 在浏览器中打开网址：[http://localhost](http://localhost/):8080/jsp-base/if-else.jsp， 产生结果如下：
* 
* 现在来看以下 switch...case 块，编写通过使用out.println()和内小脚本有一点不同，把下面代码保存到 switch-case.jsp文件中，代码详细如下：
* <%! int day = 3; %>
* <html>
* <head><title>SWITCH...CASE 示例</title></head>
* <body>
* <%
* switch(day) {
* case 0:
* out.println("It\'s Sunday.");
* break;
* case 1:
* out.println("It\'s Monday.");
* break;
* case 2:
* out.println("It\'s Tuesday.");
* break;
* case 3:
* out.println("It\'s Wednesday.");
* break;
* case 4:
* out.println("It\'s Thursday.");
* break;
* case 5:
* out.println("It\'s Friday.");
* break;
* default:
* out.println("It's Saturday.");
* }
* %>
* </body>
* </html>
* 在浏览器中打开网址：[http://localhost](http://localhost/):8080/jsp-base/switch-case.jsp， 产生结果如下：  
    
  
* **循环语句**
* 还可以使用在Java中三种基本循环类型块：for, while,and do.while 块在JSP编程中。
* 让我们来看看下面的for循环示例，把下面代码保存到 loop-for.jsp 文件中，代码详细如下：
* <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>
* <%! int fontSize; %>
* <html>
* <head><title>FOR 循环示例</title></head>
* <body>
* <%for ( fontSize = 1; fontSize <= 3; fontSize++){ %>
* <font color="green" size="<%= fontSize %>">
* JSP Tutorial
* </font><br />
* <%}%>
* </body>
* </html>
* 在浏览器中打开网址：[http://localhost](http://localhost/):8080/jsp-base/loop-for.jsp， 产生结果如下：  
    
  
* 上面的示例可以使用 while 循环编写，把下面代码保存到 loop-while.jsp 文件中，代码详细如下：
* <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>
* <%! int fontSize; %>
* <html>
* <head><title>WHILE循环示例</title></head>
* <body>
* <h2>While循环示例：</h2>
* <%while ( fontSize <= 5){ %>
* <font color="green" size="<%= fontSize %>">
* JSP Tutorial
* </font><br />
* <%fontSize++;%>
* <%}%>
* </body>
* </html>
* 在浏览器中打开网址：[http://localhost](http://localhost/):8080/jsp-base/loop-while.jsp， 产生结果如下：  
    
  
* **JSP 运算符**
* JSP支持所有支持Java的逻辑和算术运算符。下表给出了所有的运算符，具有最高优先级将排在表的顶部， 运算级别最低的放在底部。
* 在一个表达式中，具有更高的优先级运算符将首先计算评估。

|  |  |  |
| --- | --- | --- |
| **分类** | **操作符** | **关联** |
| 后缀 | () [] . (点运算符) | 左->右 |
| 一元 | ++ - - ! ~ | 右->左 |
| 乘法 | \* / % | 左->右 |
| 加法 | + - | 左->右 |
| 位移 | >> >>> << | 左->右 |
| 关系 | > >= < <= | 左->右 |
| 相等 | == != | 左->右 |
| 位与/AND | & | 左->右 |
| 位XOR | ^ | 左->右 |
| 位OR | | | 左->右 |
| 逻辑AND | && | 左->右 |
| 逻辑OR | || | 左->右 |
| 关系 | ?: | 右->左 |
| 赋值 | = += -= \*= /= %= >>= <<= &= ^= |= | 右->左 |
| 逗号 | , | 左->右 |

* **JSP 字面量**
* JSP表达式语言定义了以下字面量：
  + **Boolean**: true 或 false
  + **Integer**: 与Java中的一样
  + **Float**: 与Java中的一样
  + **String**: 单引号和双引号; " 转义为 \"。' 转义为 \'， 以及 \ 转义为 \\
  + **Null**: null

JSP对象范围

定义为一个 JSP 对象范围

说明

JSP对象所使用的可用性通常是由该对象的范围限定。

**Page 范围:**

使用此JSP对象可以在其中创建的页面内使用。

**Request 范围:**

使用该JSP对象可以在请求服务任何地方使用。

**Session 范围:**

使用该JSP的对象可用于在属于同一个会话页面。

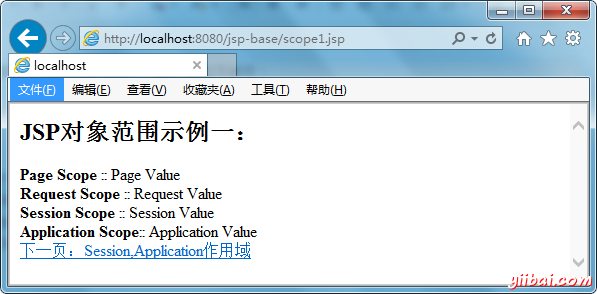
**Application 范围:**

使用该JSP的对象可以在整个应用程序页面中使用。

将下面的代码编写到 scope1.jsp 文件，代码内容如下所示：

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  
<c:set var="Pag" value="Page Value" scope="page" />  
<c:set var="Req" value="Request Value" scope="request" />  
<c:set var="Ses" value="Session Value" scope="session" />  
<c:set var="App" value="Application Value" scope="application" />  
<html>  
<body>  
    <h2>JSP对象范围示例一：</h2>  
    <b>Page Scope</b> ::<c:out value="${Pag}" /><br>  
    <b>Request Scope</b> ::<c:out value="${Req}" /><br>  
    <b>Session Scope</b> ::<c:out value="${Ses}" /><br>  
    <b>Application Scope</b>::<c:out value="${App}" /><br>  
    <a href="scope2.jsp">下一页Session,Application范围</a>  
</body>  
</html>

在浏览器中打开网址：[http://localhost](http://localhost/):8080/jsp-base/scope1.jsp， 产生结果如下：

****

将下面的代码编写到 scope2.jsp 文件，代码内容如下所示：

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  
<html>  
<body>  
<b>Page Scope</b> ::<c:out value="${Pag}" /><br>  
<b>Request Scope</b> ::<c:out value="${Req}" /><br>  
<b>Session Scope</b> ::<c:out value="${Ses}" /><br>  
<b>Application Scope</b>::<c:out value="${App}" /><br>  
</body>  
</html>

上面的代码运行结果如下：

Page Scope ::  
Request Scope ::  
Session Scope ::Session Value  
Application Scope ::Application Value

在上面的例子“scope1.jsp”中，变量在四个作用域设置的值都被显示出来。当点击一个链接，并跳转到下一个页面“scope2.jsp”只显示会话和应用范围的变量值，因为第二页没有页面，请求作用域。

在JSP页面中创建方法

学习在一个JSP页面里面创建方法

解释

方法即是可以在一个JSP页面被用于执行特定操作的一个代码段。

创建一个方法示例：

<%!  
public int mul(int a, int b){  
    return a \* b;  
}  
%>  
两个数相乘的结果是：<%= mul(2, 2) %>

以上代码保存，并执行结果如下：

两个数相乘的结果是：4

另外，在上述例子中使用的方法是：mul，将返回一个整数值作为输出。 它需要两个整数“a”，“b”作为参数，以产生两个数字的乘积作为输出。

在JSP页面中使用数组

如何在JSP页面中使用数组？

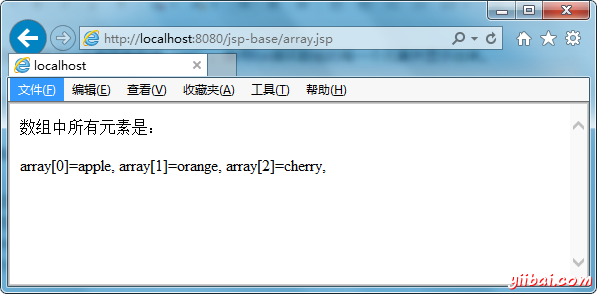
解释说明

由于JSP不是一个完整的编程语言不具有数组的声明，但使用Java中的数据结构在JSP中使用是完全没有问题的。

把下面的代码保存到 array.jsp 文件中，启动 Tomcat 容器，并在浏览器中找开网址测试显示结果。

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  
<%  
String[] arr={"apple","orange","cherry"};  
%>  
<%  
int j;  
out.println("<p>数组中所有元素是：</p>");  
for(j=0;j<arr.length;j++){  
    out.println(arr[j]);  
}  
%>

把上面的代码保存到文件：arr\_demo.jsp，并在浏览器中打开网址：<http://localhost:8080/jsp-base/array.jsp>，得到以下结果：



在这个例子中，已经声明了一个字符串数组“arr”。使用for循环数组的每一个元素并显示出来。

在JSP中使用Java Bean

如何在JSP中使用Java Bean？

Java bean只不过是一个实现java.io.Serializable接口，并且使用set/get方法来投射类的属性。因为它们是一个Java bean类的实例并可重复使用，在JSP页面中提供了灵活性，。  
嵌入一个Java bean到JSP网页，有三个基本动作或标签：<jsp:useBean>, <jsp:setProperty>, <jsp:getProperty>

**<jsp:useBean>**

这个标签是用来给bean指定“id”和“scope”属性相关联。

**<jsp:setProperty>**

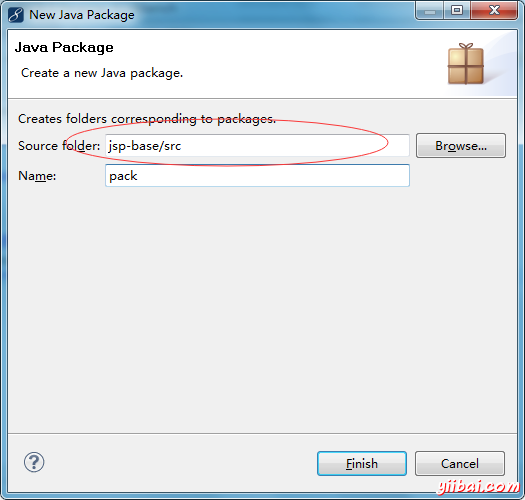
这个标签被用于设置一个beans属性的值，主要使用“name”属性已经定义了相同的范围对象。其他属性是 "property", "param", "value"

**<jsp:getProperty>**

这个标签是用来获取引用Bean实例属性并将其存储到隐式out对象。

**Beans的规则：**

* 包应该是java bean的第一行
* Bean应该有一个空的构造
* 所有的bean中的变量应该设置有“get”，“set”方法。
* 属性名应以大写字母开头在使用“set”，“get”方法时。
* 例如变量“名称”的get，set方法就是getName(), setName(String)
* 设置方法应该返回像一个空(void)值： "return void()"

创建一个包，名称是：pack，打开工程，在 jsp-base 工程下，右键 "src" -> "new" -> "Package"，然后出现对话框，写入包名：pack，如下所示：  
  


现在在 pack 包创建一个类，类的名称是：Counter，对应文件是：src/pack/Counter.java，文件的详细代码如下图所示：

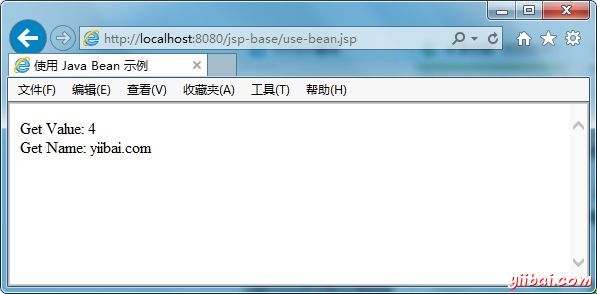
package pack;  
public class Counter  
{  
int count = 0;  
String name;  
public Counter() { }  
public int getCount()  
{  
    return this.count;  
}  
public void setCount(int count){  
    this.count = count;  
}   
public String getName(){  
    return this.name;   
}   
public void setName(String name){  
    this.name=name;   
}  
}

创建一个文件：use-bean.jsp，其代码内容如下所示：

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  
<%@ page language="java" %>  
<%@ page import="pack.Counter" %>  
</body>  
<jsp:useBean id="counter" scope="page" class="pack.Counter" />  
<jsp:setProperty name="counter" property="count" value="4" />  
Get Value: <jsp:getProperty name="counter" property="count" /><BR>  
<jsp:setProperty name="counter" property="name" value="yiibai.com" />  
Get Name: <jsp:getProperty name="counter" property="name" /><BR>

另外，在上述示例正在包含的是Counter.java，已为“counter”，“name”属性定义对象的set/get方法。使用 use-bean.jsp页面使用<jsp：useBean>动作包含第一个bean，使用<jsp;setProperty>设置属性 "counter" 的值为 "4" ，以及设置 "name" 的值为 "yiibai.com". 并使用 <jsp:getProperty> 来获取它们的值。

把上面的代码保存到文件：use-bean.jsp，并在浏览器中打开网址：http://localhost:8080/jsp-base/use-bean.jsp，得到以下结果：



JSP Session

在 JSP中如何处理 Session 数据?

**解释说明**

会话处理变得必不可少，当一个请求数据需要被持续保持以供进一步使用。 由于HTTP协议认为每个请求是一个新的请求，它不能保持过去访问的数据，因此会话处理就变得很重要。以下是一些来处理会话的方法。

* JSP中每当发起一个请求，服务器产生一个存储在客户机的唯一会话ID。
* Cookies存储信息在客户端浏览器
* URL重写会话信息附加到URL的末尾
* 隐藏的表单域将SessionID嵌入到GET和POST命令。

创建一个页面 session.jsp，代码如下：

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  
<html>  
<head><title>Session示例</title></head>  
<body>  
<h2>Session示例：</h2>   
<form method = "post" action="session2.jsp">  
<font>Username<input type = "text" name = "name"></font>  
</font><br><br>  
<input type = "submit" name = "submit" value = "提交" >  
</form>  
</body>  
</html>

创建另一个页面 session2.jsp，代码如下：

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  
<html>

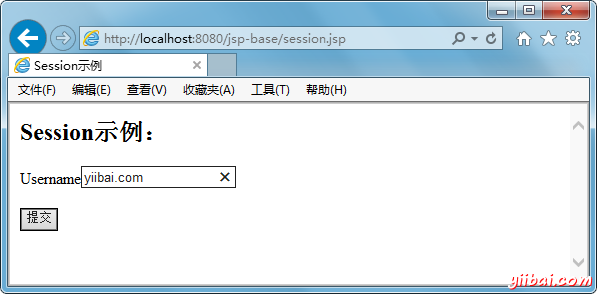
<head><title>Session示例2</title></head>

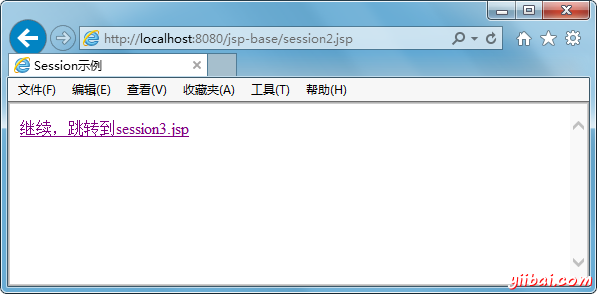
<body>   
<%  
String name = request.getParameter("name");  
if((name!=null))  
{  
    session.setAttribute("username",name);  
}  
%>  
<a href="session3.jsp">继续，跳转到session3.jsp</a>

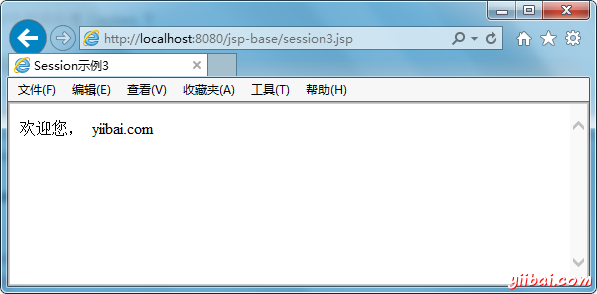
创建另一个页面 session3.jsp，代码如下：

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  
<html>  
<head><title>Session示例3</title></head>   
<body>  
<font>欢迎您，</font> <%= session.getAttribute("username") %>  
</body>  
</html>

在第一个示例中的 "session1.jsp" 是用来提供一个表单以获得用户名。 当提交表单时它转到第二个文件session2.jsp，它调用表单的“action”属性。一个Session的属性使用 "session.setAttribute"设置在这里. 在第三个文件 "session3.jsp" 相同的值使用"session.getAttribute" 来显示出来。

在浏览器中打开网址：http://localhost:8080/jsp-base/session.jsp，得到以下结果：  
  


填写好用户名，点击“提交”，进入第二个过度页面，如下：  
  


然后再点击页面中的链接，进入第三个页面，测试Session记录的结果信息，如下图：  
  


JSP Cookies信息处理

在JSP中如何处理 Cookies ？

**解释说明**

Cookies和会话有一些相似的地方，唯一的区别是cookies 存储在浏览器。而会话存储在服务器端。让我们来看看在JSP中处理Cookies的一些例子：

示例代码: cookie1.jsp

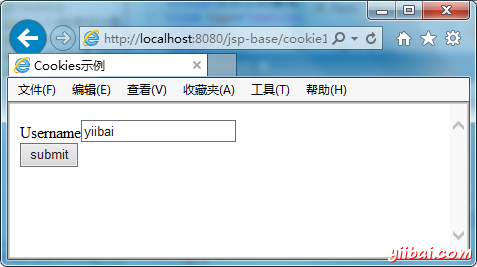
<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  
<html>  
<body>  
<form method = "post" action="cookie2.jsp">  
<font>Username<input type = "text" name = "name"></font>  
</font><br>  
<input type = "submit" name = "submit" value = "submit" >  
</form>  
</body>  
</html>

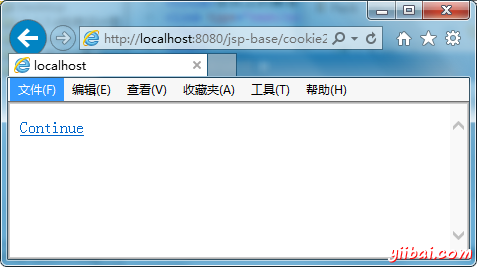
示例代码: cookie2.jsp

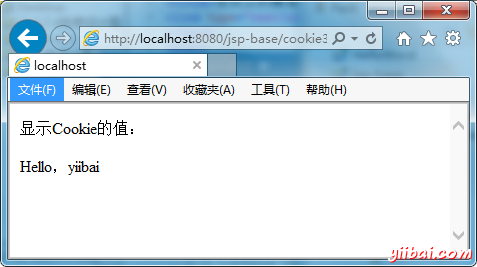
<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  
<%@ page language="java" import="java.util.\*"%>  
<%  
String name=request.getParameter("name");  
Cookie cookie = new Cookie ("name",name);  
response.addCookie(cookie);  
cookie.setMaxAge(50 \* 50); //Time is in Minutes  
%>  
<a href="cookie3.jsp">Continue</a>

示例代码: cookie3.jsp

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  
<p>显示Cookie的值：</p>  
<%  
Cookie[] cookies = request.getCookies();  
for (int i=0; i<cookies.length; i++){  
    if(cookies[i].getName().equals("name"))  
        out.println("Hello"+cookies[i].getValue());  
}  
%>

在第一个例子“cookie1.jsp”中，使用表单得到用户名。当表单提交进入第二页“cookie2.jsp”其中Cookie使用“cookie.setMaxAge”函数设置过期时间。在第三个页面“cookie3.jsp”中，cookie使用函数“request.getCookies()”，并使用循环是得到了字段“username”的值并显示。在浏览器中输入网址：http://localhost:8080/jsp-base/cookie1.jsp ，测试输出结果如下：  
  


点击"submit"按钮后，得到以下界面：  
  


点击"Continue"链接后，得到结果如下界面所示：  
  


**删除Cookies**

**示例代码: cookie4.jsp**

<%  
Cookie cookie = new Cookie( "name", "" );  
cookie.setMaxAge( 0 );  
%>

另外，在上述例子中，我们已经创建的cookie的新实例并使用一个“null”值并将cookie的age设定为“0”。 这将删除cookie。

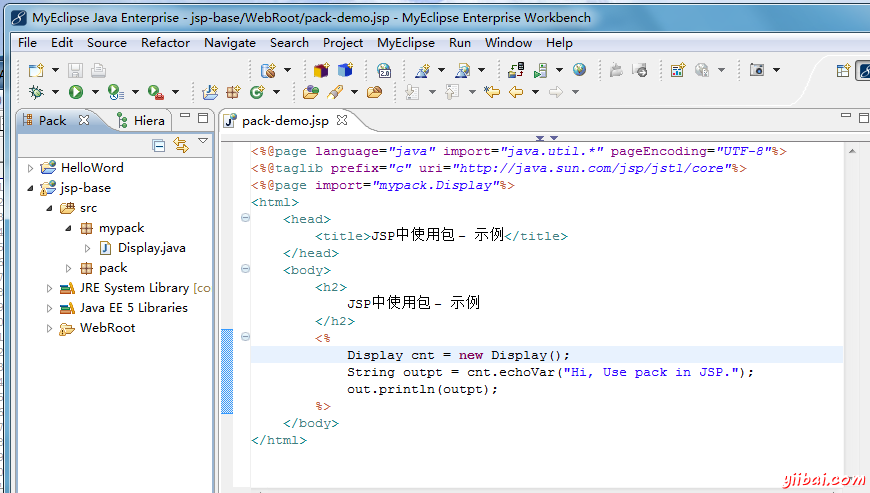
在JSP中使用包

如何在JSP中使用包? 这里给出一个示例...

**解释说明**

Java类可以存储在包中，在JSP中可得到有效使用。 下面是解释Java类用法的一个例子。

**pack-demo.jsp**



**JSP中使用包 - 示例**

**Counter.java**

package mypack;

public class Display {

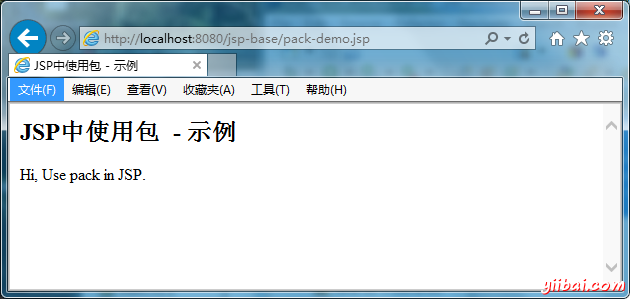
public String echoVar(String var) {

return var;

}

}

  在上面的例子中，我们使用包“mypack”中的类“Counter”。在这个例子中，我们在包创建一个类“Counter”，并实例化一个实例使用“echovar”方法中显示输入的文本。在浏览器中输入网址：http://localhost:8080/jsp-base/pack-demo.jsp ，测试输出结果如下：



**JSP 动作元素**

与JSP指令元素不同的是，JSP动作元素在请求处理阶段起作用。JSP动作元素是用XML语法写成的。

利用JSP动作可以动态地插入文件、重用JavaBean组件、把用户重定向到另外的页面、为Java插件生成HTML代码。

动作元素只有一种语法，它符合XML标准：

<jsp:action\_name attribute="value" />

动作元素基本上都是预定义的函数，JSP规范定义了一系列的标准动作，它用JSP作为前缀，可用的标准动作元素如下：

|  |  |
| --- | --- |
| **语法** | **描述** |
| jsp:include | 在页面被请求的时候引入一个文件。 |
| jsp:useBean | 寻找或者实例化一个JavaBean。 |
| jsp:setProperty | 设置JavaBean的属性。 |
| jsp:getProperty | 输出某个JavaBean的属性。 |
| jsp:forward | 把请求转到一个新的页面。 |
| jsp:plugin | 根据浏览器类型为Java插件生成OBJECT或EMBED标记。 |
| jsp:element | 定义动态XML元素 |
| jsp:attribute | 设置动态定义的XML元素属性。 |
| jsp:body | 设置动态定义的XML元素内容。 |
| jsp:text | 在JSP页面和文档中使用写入文本的模板 |

**常见的属性**

所有的动作要素都有两个属性：id属性和scope属性。

* **id属性：**

id属性是动作元素的唯一标识，可以在JSP页面中引用。动作元素创建的id值可以通过PageContext来调用。

* **scope属性：**

该属性用于识别动作元素的生命周期。 id属性和scope属性有直接关系，scope属性定义了相关联id对象的寿命。 scope属性有四个可能的值： (a) page, (b)request, (c)session, 和 (d) application。

**<jsp:include>动作元素**

<jsp:include>动作元素用来包含静态和动态的文件。该动作把指定文件插入正在生成的页面。语法格式如下：

<jsp:include page="相对 URL 地址" flush="true" />

　前面已经介绍过include指令，它是在JSP文件被转换成Servlet的时候引入文件，而这里的jsp:include动作不同，插入文件的时间是在页面被请求的时候。

以下是include动作相关的属性列表。

|  |  |
| --- | --- |
| **属性** | **描述** |
| page | 包含在页面中的相对URL地址。 |
| flush | 布尔属性，定义在包含资源前是否刷新缓存区。 |

**实例**

以下我们定义了两个文件 **date.jsp** 和 **main.jsp**，代码如下所示：

date.jsp文件代码：

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<p>

今天的日期是: <%= (new java.util.Date()).toLocaleString()%>

</p>

main.jsp文件代码：

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>菜鸟教程(runoob.com)</title>

</head>

<body>

<h2>include 动作实例</h2>

<jsp:include page="date.jsp" flush="true" />

</body>

</html>

现在将以上两个文件放在服务器的根目录下，访问main.jsp文件。显示结果如下：

include 动作实例

今天的日期是: 2016-6-25 14:08:17

**<jsp:useBean>动作元素**

**jsp:useBean** 动作用来加载一个将在JSP页面中使用的JavaBean。

这个功能非常有用，因为它使得我们可以发挥 Java 组件复用的优势。

jsp:useBean动作最简单的语法为：

<jsp:useBean id="name" class="package.class" />

在类载入后，我们既可以通过 jsp:setProperty 和 jsp:getProperty 动作来修改和检索bean的属性。

以下是useBean动作相关的属性列表。

|  |  |
| --- | --- |
| **属性** | **描述** |
| class | 指定Bean的完整包名。 |
| type | 指定将引用该对象变量的类型。 |
| beanName | 通过 java.beans.Beans 的 instantiate() 方法指定Bean的名字。 |

在给出具体实例前，让我们先来看下 jsp:setProperty 和 jsp:getProperty 动作元素：

**<jsp:setProperty>动作元素**

jsp:setProperty用来设置已经实例化的Bean对象的属性，有两种用法。首先，你可以在jsp:useBean元素的外面（后面）使用jsp:setProperty，如下所示：

<jsp:useBean id="myName" ... />

...

<jsp:setProperty name="myName" property="someProperty" .../>

此时，不管jsp:useBean是找到了一个现有的Bean，还是新创建了一个Bean实例，jsp:setProperty都会执行。第二种用法是把jsp:setProperty放入jsp:useBean元素的内部，如下所示：

<jsp:useBean id="myName" ... >

...

<jsp:setProperty name="myName" property="someProperty" .../>

</jsp:useBean>

此时，jsp:setProperty只有在新建Bean实例时才会执行，如果是使用现有实例则不执行jsp:setProperty。

jsp:setProperty动作有下面四个属性,如下表：

|  |  |
| --- | --- |
| **属性** | **描述** |
| name | name属性是必需的。它表示要设置属性的是哪个Bean。 |
| property | property属性是必需的。它表示要设置哪个属性。有一个特殊用法：如果property的值是"\*"，表示所有名字和Bean属性名字匹配的请求参数都将被传递给相应的属性set方法。 |
| value | value 属性是可选的。该属性用来指定Bean属性的值。字符串数据会在目标类中通过标准的valueOf方法自动转换成数字、boolean、Boolean、 byte、Byte、char、Character。例如，boolean和Boolean类型的属性值（比如"true"）通过 Boolean.valueOf转换，int和Integer类型的属性值（比如"42"）通过Integer.valueOf转换。 　　value和param不能同时使用，但可以使用其中任意一个。 |
| param | param 是可选的。它指定用哪个请求参数作为Bean属性的值。如果当前请求没有参数，则什么事情也不做，系统不会把null传递给Bean属性的set方法。因此，你可以让Bean自己提供默认属性值，只有当请求参数明确指定了新值时才修改默认属性值。 |

**<jsp:getProperty>动作元素**

jsp:getProperty动作提取指定Bean属性的值，转换成字符串，然后输出。语法格式如下：

<jsp:useBean id="myName" ... />

...

<jsp:getProperty name="myName" property="someProperty" .../>

下表是与getProperty相关联的属性：

|  |  |
| --- | --- |
| **属性** | **描述** |
| name | 要检索的Bean属性名称。Bean必须已定义。 |
| property | 表示要提取Bean属性的值 |

**实例**

以下实例我们使用了Bean:

package com.runoob.main;

public class TestBean {

private String message = "菜鸟教程";

public String getMessage() {

return(message);

}

public void setMessage(String message) {

this.message = message;

}

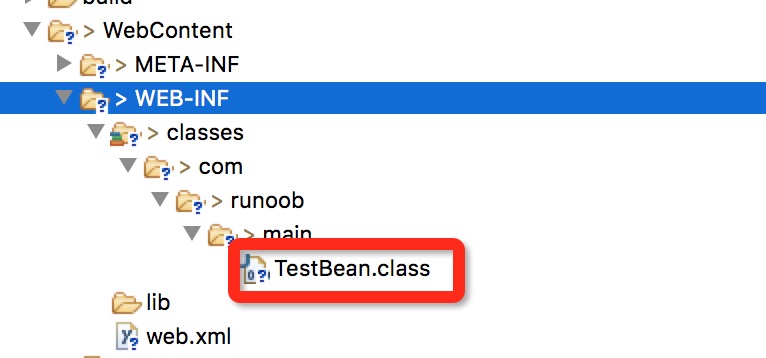
}

编译以上实例文件 TestBean.java ：

$ javac TestBean.java

编译完成后会在当前目录下生成一个 **TestBean.class** 文件， 将该文件拷贝至当前 JSP 项目的 **WebContent/WEB-INF/classes/com/runoob/main** 下( com/runoob/main 包路径，没有需要手动创建)。

下面是一个 Eclipse 中目录结构图：



下面是一个很简单的例子，它的功能是装载一个Bean，然后设置/读取它的message属性。

现在让我们在main.jsp文件中调用该Bean:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>菜鸟教程(runoob.com)</title>

</head>

<body>

<h2>Jsp 使用 JavaBean 实例</h2>

<jsp:useBean id="test" class="com.runoob.main.TestBean" />

<jsp:setProperty name="test"

property="message"

value="菜鸟教程..." />

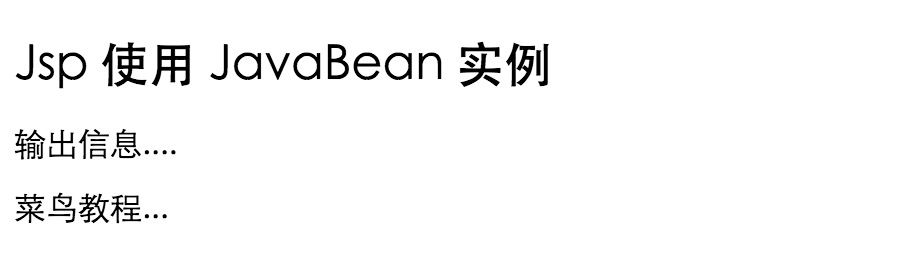
<p>输出信息....</p>

<jsp:getProperty name="test" property="message" />

</body>

</html>

浏览器访问，执行以上文件，输出如下所示：



**<jsp:forward> 动作元素**

　jsp:forward动作把请求转到另外的页面。jsp:forward标记只有一个属性page。语法格式如下所示：

<jsp:forward page="相对 URL 地址" />

以下是forward相关联的属性：

|  |  |
| --- | --- |
| **属性** | **描述** |
| page | page属性包含的是一个相对URL。page的值既可以直接给出，也可以在请求的时候动态计算，可以是一个JSP页面或者一个 Java Servlet. |

**实例**

以下实例我们使用了两个文件，分别是： date.jsp 和 main.jsp。

date.jsp 文件代码如下：

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<p>

今天的日期是: <%= (new java.util.Date()).toLocaleString()%>

</p>

main.jsp文件代码：

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>菜鸟教程(runoob.com)</title>

</head>

<body>

<h2>forward 动作实例</h2>

<jsp:forward page="date.jsp" />

</body>

</html>

现在将以上两个文件放在服务器的根目录下，访问main.jsp文件。显示结果如下：

今天的日期是: 2016-6-25 14:37:25

**<jsp:plugin>动作元素**

jsp:plugin动作用来根据浏览器的类型，插入通过Java插件 运行Java Applet所必需的OBJECT或EMBED元素。

如果需要的插件不存在，它会下载插件，然后执行Java组件。 Java组件可以是一个applet或一个JavaBean。

plugin动作有多个对应HTML元素的属性用于格式化Java 组件。param元素可用于向Applet 或 Bean 传递参数。

以下是使用plugin 动作元素的典型实例:

<jsp:plugin type="applet" codebase="dirname" code="MyApplet.class"

width="60" height="80">

<jsp:param name="fontcolor" value="red" />

<jsp:param name="background" value="black" />

<jsp:fallback>

Unable to initialize Java Plugin

</jsp:fallback>

</jsp:plugin>

如果你有兴趣可以尝试使用applet来测试jsp:plugin动作元素，<fallback>元素是一个新元素，在组件出现故障的错误是发送给用户错误信息。

**<jsp:element> 、 <jsp:attribute>、 <jsp:body>动作元素**

<jsp:element> 、 <jsp:attribute>、 <jsp:body>动作元素动态定义XML元素。动态是非常重要的，这就意味着XML元素在编译时是动态生成的而非静态。

以下实例动态定义了XML元素：

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>菜鸟教程(runoob.com)</title>

</head>

<body>

<jsp:element name="xmlElement">

<jsp:attribute name="xmlElementAttr">

属性值

</jsp:attribute>

<jsp:body>

XML 元素的主体

</jsp:body>

</jsp:element>

</body>

</html>

浏览器访问以下页面，输出结果如下所示：



**<jsp:text>动作元素**

<jsp:text>动作元素允许在JSP页面和文档中使用写入文本的模板，语法格式如下：

<jsp:text>模板数据</jsp:text>

以上文本模板不能包含其他元素，只能只能包含文本和EL表达式（注：EL表达式将在后续章节中介绍）。请注意，在XML文件中，您不能使用表达式如 ${whatever > 0}，因为>符号是非法的。 你可以使用 ${whatever gt 0}表达式或者嵌入在一个CDATA部分的值。

<jsp:text><![CDATA[<br>]]></jsp:text>

如果你需要在 XHTML 中声明 DOCTYPE,必须使用到<jsp:text>动作元素，实例如下：

<jsp:text><![CDATA[<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"DTD/xhtml1-strict.dtd">]]>

</jsp:text>

<head><title>jsp:text action</title></head>

<body>

<books><book><jsp:text>

Welcome to JSP Programming

</jsp:text></book></books>

</body>

</html>

你可以对以上实例尝试使用<jsp:text>及不使用该动作元素执行结果的区别。

← [JSP 指令](http://www.runoob.com/jsp/jsp-directives.html)

[JSP 隐式对象](http://www.runoob.com/jsp/jsp-implicit-objects.html) →