JavaBean详解---JavaBean与WEB开发标准目录结构

作者：[SkewRain](http://my.csdn.net/SkewRain)

JavaBean简介

JavaBean是使用Java语言开发的一个可重用的组件，在JSP的开发中可以使用JavaBean减少重复代码，使整个JSP代码的开发更简洁。JSP搭配JavaBean来使用，有以下的优点：

1）：可将HTML和Java代码分离，这主要是为了日后维护的方便。如果把所有的程序代码（HTML和Java）写到JSP页面中，会使整个程序代码又多又复杂，造成日后维护上的困难。

2）：可利用JavaBean的优点。将常用到的程序写成JavaBean组件，当在JSP要使用时，只要调用JavaBean组件来执行用户所要的功能，不用再重复写相同的程序，这样一来也可以节省开发所需的时间。

JavaBean开发要求

JavaBean本身就是一个类，属于Java的面向对象编程。

在JSP中如果要应用JSP提供的Javabean的标签来操作简单类的话，则此类必须满足如下的开发要求。

1）：所有的类必须放在一个包中，在WEB中没有包的类是不存在的；

2）：所有的类必须声明为public class，这样才能够被外部所访问；

3）：类中所有的属性都必须封装，即：使用private声明；

4）：封装的属性如果需要被外部所操作，则必须编写相应的setter、getter方法。

5）：一个JavaBean中至少存在一个无参的构造方法，此为JSP中的标签所使用。

以下是一个简单的JavaBean。【skewrain.java的内容如下：】

package com.skewrain.demo;

public class SimpleBean {

    private String name;

    private int age;

    public void setName(String name){

        this.name = name;

    }

    public void setAge(int age){

        this.age = age;

    }

    public String getName(){

        return this.name;

    }

    public int getAge(){

        return this.age;

    }

}

也许你会疑问，现在的JavaBean中怎么没有无参构造方法？

现在的类中是存在构造方法的，因为如果一个类中没有明确的定义一个构造方法的话，则会自动生成一个无参的，什么都不做的构造方法。

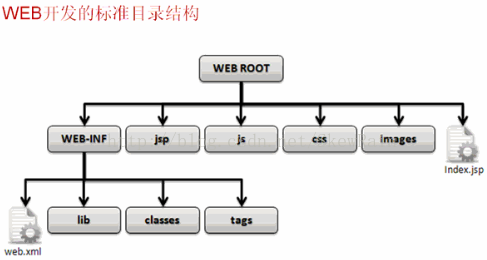
但是，在这里需要说明的是，对于这样的简单JavaBean也有几个名词：

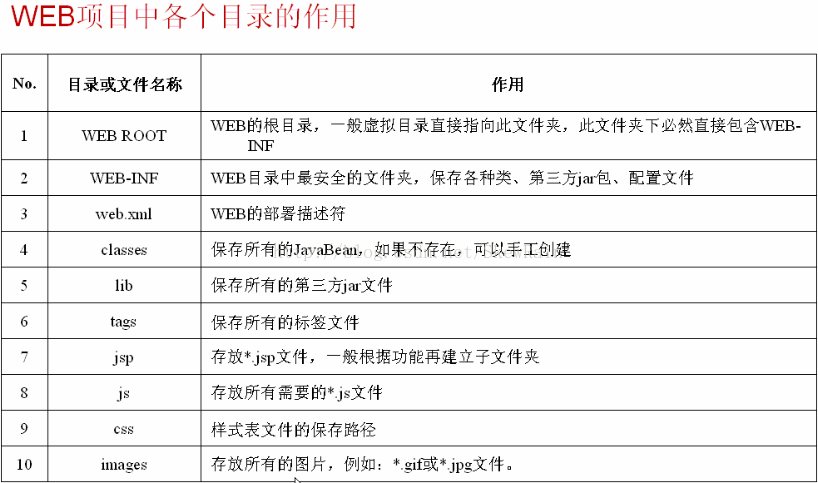
1）：VO：值对象，存放所有的传递数据的操作上；

2）：POJO：简单Java对象；

3）：TO：传输对象，一个类必须实现Serializable接口后才称为传输对象。

JavaBean本身是一个\*.java的文件，按照最原始的方式解释，现在的Java程序编译成\*.class文件才可以使用，但是放在哪里呢？





实际上，在WEB-INF中的llib和classes文件夹就相当于一个默认的classpath。

Classpath是类执行时所需要的一个重要的环境属性，当Tomcat启动之后，WEB-INF/lib和WEB-INF/classes都会自动配置到classpath之中。

既然现在可以成功的进行编译了，则下面就可以使用两种方式使用JavaBean，因为WEB-INF/classes文件夹就是一个classpath的路径，所以可以直接使用page指令导入所需要的JavaBean。

用法一：【javabean\_01.jsp】

<%@ page contentType="text/html" pageEncoding="GBK"%>

<%@ page import="com.skewrain.demo.\*"%>

<html>

<head><title>javabean</title></head>

<body>

<%

    SimpleBean simple = new SimpleBean();

    simple.setName("斜雨");

    simple.setAge(30);

%>

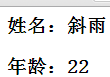
<h3>姓名：<%=simple.getName()%></h3>

<h3>年龄：<%=simple.getAge()%></h3>

</body>

</html>

执行此段代码的结果如下：



用法二：

除了这第一种使用方法之外，也可以使用专门提供的JSP标签完成操作，在整个JavaBean对于这种简单bean的操作提供了—useBean标签。

使用<jsp:useBean>指令

除了使用import的语句之外，也可以使用JSP中提供的：<jsp:useBean>指令完成操作，指令的操作语法如下所示：

<jsp:useBean id=”实例化对象名称” scope=”保存范围” class=”包.类名称”/>

主要属性：

1）：id：表示实例化对象的名称；

2）：scope：表示此对象保存的范围，一共有四种属性范围：page、request、session、application。

3）：class：对象所对应的包.类名称。

【javabean\_02.jsp的内容如下：】

<%@ page contentType="text/html" pageEncoding="GBK"%>

<jsp:useBean id="simple" scope="page" class="com.skewrain.demo.SimpleBean"/>

<html>

<head><title>javabean</title></head>

<body>

<%

    simple.setName("斜雨");

    simple.setAge(22);

%>

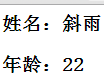
<h3>姓名：<%=simple.getName()%></h3>

<h3>年龄：<%=simple.getAge()%></h3>

</body>

</html>

执行此程序的结果如下：



实际上在使用以上的标签的时候，都会自动调用JavaBean中无参数的构造方法，那么所以之前才会要求在一个JavaBean中如果需要通过JSP标签调用的话，则必须存在一个无参的构造方法，就在这里了。

当一个JavaBean修改之后，服务器一定要重新启动，只有在重新启动之后才会将新的配置加载进来。但是，这个时候就出现一个问题，如果在开发中，肯定要重复的修改JavaBean，如果每次都要这样去做的话是不是太麻烦了？

所以，在Tomcat中提供了自动的加载机制，但是这种加载机制需要通过server.xml文件进行配置。

<Context path="/skewrain" docBase="D:\org" reloadable="true">

以后的Tomcat会自动进行检测，如果发现JavaBean修改之后会自动地进行重新加载。

但是有两点需要注意：

1.由于reloadable一直要处于监视状态，所以在服务器真正用于项目运行的时候，此属性一定要设置成false，否则会造成性能下降。

2.当每次进行完重新加载之后，之前所设置的全部的session范围的属性将彻底消失。

JavaBean是什么原理？

为什么直接通过这样的一个标签就可以完成对象实例化呢？

在useBean的标签中存在了一个class的定义，这里面写的是一个包.类名称，如果按照之前学习过的反射机制完成的话，现在已经存在了包.类，肯定可以通过class进行实例化，而且这个时候的类中还要求存在无参构造方法，那么就更加明确了，肯定是通过反射机制进行对象的实例化操作的。

小结：

1）：通过JavaBean可以封装JAVA的复杂操作；

2）：JSP中提供了专门的JavaBean操作标签，也可以使用page导入。