javabean简单及使用：

参考：<http://www.cnblogs.com/mq0036/p/3873406.html>

**一、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。将日常用到的程序写成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，这样以来也可以节省开发所需的时间。

**二、JavaBean开发要求**

1.JavaBean本身就是一个类（只是这个类可以提供jsp开发的一些方法和属性的调用的组件），属于Java的面向对象编程。

2.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，则必须编写对应的setter、getter方法；

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

第一个简单JavaBean

1. **package blog.csdn.joywy;**
2. **public class SimpleBean{**
3. **private String name;   //封装**
4. **private int age;   //封装**
5. **public void setName(String name){   //封装属性，外部操作时的setter方法，就是属性访问器**
6. **this.name = name;**
7. }
8. **public void setAge(int age){**
9. **this.age = age;**
10. }
11. **public String getName(){   //属性访问器**
12. **return this.name;**
13. }
14. **public int getAge(){**
15. **return this.age;**
16. }
17. }

如果在一个类中只包含属性、setter、getter方法，那么这种类就成为简单JavaBean—简单的无规则的pojo（plain ordinary java object）。

对于简单的JavaBean也有几个名词：

(1)VO（value object）：与简单Java对象对应，专门用于传递值的操作上,简单理解为键值对

(2)POJO：简单Java对象

(3)TO(transfer object)：传输对象，进行远程传输时，对象所在的类必须实现java.io.Serializable接口(对象传输时要序列化，得到对象时要反序列化)。

**三、WEB开发的标准目录结构**

Web项目中各个目录的作用：

|  |  |  |
| --- | --- | --- |
| No. | 目录或文件名称 | 作用 |
| 1 | WEB ROOT | Web的根目录，一般虚拟目录直接指向此文件夹，此文件夹下必然直接包含WEB-INF |
| 2 | WEB-INF | Web目录中最安全的文件夹，保存各种类、第三方jar包、配置文件 |
| 3 | Web.xml | Web的部署描述符 |
| 4 | classes | 保存所有的JavaBean，如果不存在，可以手工创建 |
| 5 | lib | 保存所有的第三方jar文件 |
| 6 | tags | 保存所有的标签文件 |
| 7 | jsp | 存放\*.jsp文件，一般根据功能再建立子文件夹 |
| 8 | js | 存放所有需要的\*.js文件 |
| 9 | css | 样式表文件的保存路径 |
| 10 | images | 存放所有的图片，如\*.gif或\*.jpg文件 |

注：

1.打包编译：编译时直接使用javac -d . SimpleBean.java命令根据package的定义打包编译。

2.Tomcat中classpath的配置：

所有的class要进行访问时都必须在classpath中进行配置，在Tomcat中依然可以使用，在Web开发中，Tomcat安装目录的lib文件夹、WEB-INF\classes和WEB-INF\lib文件夹实际上都表示了classpath，所以直接将类或jar包复制到这些文件夹中就可以直接使用了。

1. <%@ page contentType="text/html" pageEncoding="GBK"%>
2. <%@ page import="blog.csdn.joywy.\*"%>    <!-- 导入blog.csdn.joywy包 -->
3. <html>
4. <head><title>欢迎来到望星空</title></head>
5. <body>
6. <%
7. SimpleBean simple = new SimpleBean();    //声明并实例化SimpleBean对象
8. simple.setName("Joywy");                 //设置name属性
9. simple.setAge(23);                       //设置age属性
10. %>
11. <h3>姓名：<%=simple.getName()%></h3>            <!-- 输出name属性的内容 -->
12. <h3>年龄：<%=simple.getAge()%></h3>             <!-- 输出age属性的内容 -->
13. </body>
14. </html>

**四、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：

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

2.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  
样例：

1. <%@ page contentType="text/html" pageEncoding="GBK"%>
2. <jsp:useBean id="simple" scope="page" class="blog.csdn.joywy.SimpleBean"/>
3. <html>
4. <head><title>欢迎来到望星空</title></head>
5. <body>
6. <%
7. simple.setName("Joywy");
8. simple.setAge(23);
9. %>
10. <h3>姓名：<%=simple.getName()%></h3>
11. <h3>年龄：<%=simple.getAge()%></h3>
12. </body>
13. </html>

在使用<jsp:useBean/>指令时，实际上会默认调用SimpleBean类中的无参构造方法进行对象实例化，当修改SimpleBean.java文件后，如果要想让其起作用，则需要重新启动服务器，我们可以通过在Tomcat中直接配置自动加载的操作，这样即使不重新启动服务器，JavaBean修改后也可以被立即加载进来。重新加载的配置如下：在Tomcat安装目录下conf/server.xml文件中进行配置

1. <Context path="/joywy" docBase="D:\myweb" reloadable="true"/>

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

注意：

1.直接在配置虚拟目录的位置上增加一个reloadable属性即可，但是这种情况只适合于开发，因为使用reloadable自动加载后，服务器将始终处于监视状态，一旦发现类修改后就要立刻重新加载，运行性能是比较低的，所以，当项目真正发布运行时一定要将reloadable的内容设置成false，以提升入屋企运行性能。

2.还要记住，当Tomcat重新加载了新的内容后，所有的操作都将初始化，所有设置过的session属性都将消失。

**四、<jsp:useBean>指令内部实现原理**

依靠的是反射机制完成的，在使用<jsp:useBean>指令时需要定义对象所在的“包.类“名称，而且又要求类中存在无参构造方法，这就是反射操作的前提条件，所以<jsp:useBean>的语法实际上属于反射机制的操作实现。

出处：http://blog.csdn.net/joywy/article/details/7773503