# JavaBean简介

## 认识JavaBean

JavaBean是使用Java语言开发的一个可重用的组件，也就是遵循一定规则的Java类。

此类满足以下条件：

1. 所有的类都必须放在包中，Web开发中没有包的类是不存在的
2. 所有的类都必须声明为public class，这样才能被外部访问
3. 类中所有属性必须全部封装，即使用private声明，并且需要为每一个属性编写setter

和getter方法

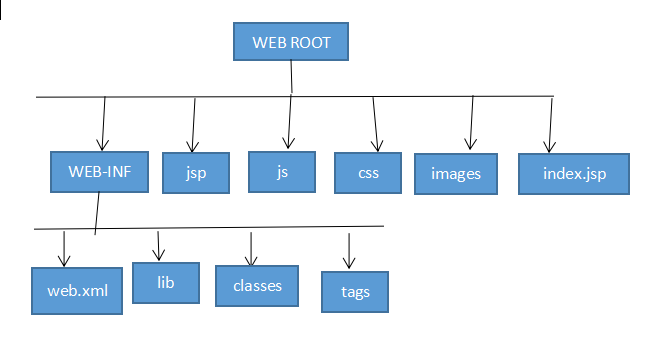
1. 类中至少存在一个无参的构造方法，用于给JSP标签实例化使用（后面会讲）。

## JavaBean的优点

1. 将HTML代码与Java代码分离，便于维护。
2. 可以重复使用，减少重复的代码。

# 在JSP中使用JavaBean

## Web开发的标准目录结构

一个JavaBean编写完成后，需要进行打包编译，而对于编译好的JavaBean保存的存放位置，则需要掌握Web开发的标准目录结构。

1. 在WEB-INF目录中定义了一个名为classes文件夹，如果没有则自行创建。在此文件

夹中保存所有的\*.class文件；

1. 除了目录结构外，所有项目基本上都会在根目录中存放一个首页文件，一般以 index.jsp、index.html、等形式命名。可以在web.xml中配置默认首页。

各个文件或目录的介绍如下：

|  |  |  |
| --- | --- | --- |
| NO | 目录或文件文件 | 作用 |
| 1 | WEB ROOT | Web的根目录，一般虚拟目录直接指向此文件夹，此文件中必然存在WEB-INF目录 |
| 2 | WEB-INF | Web目录中最安全的文件夹，保存各种类、第三方jar包、配置文件等 |
| 3 | web.xml | Web项目的部署描述符 |
| 4 | classes | 保存所有的JavaBean，即\*.class类，如该目录不存在，可以手工创建 |
| 5 | lib | 保存所有的第三方jar文件包 |
| 6 | tags | 保存所有的标签文件，如JSTL |
| 7 | jsp | 存放\*.jsp文件，一般根据子功能可再建立子文件夹 |
| 8 | js | 存放所有的\*.js文件 |
| 9 | css | 样式表文件的保存路径 |
| 10 | images | 存放项目的所有图片 |

在学习Java的时候讲解过，所有的class要进行访问都必须在classpath中进行配置之后才可以使用。如Tomcat安装目录的lib文件夹，Web项目目录中的WEB-INF/lib、WEB-INF/class文件夹实际上都表示了classpath，所以直接将类或jar包或\*.class文件复制到这些文件夹中就可以直接使用。

## 通过page指令使用JavaBean

在JSP中可以**使用<%@page%>指令的import属性**导入指定的classpath里所需要的包和类，那么一个JavaBean开发完成后，也可以按照此方式导入。语法为：

**<%@page import=”包.类” %>**

使用这种方式导入JavaBean后，在代码中需要使用new关键字实例化对象。

## 通过<jsp:useBean/>标签使用JavaBean

除了使用<%@page%>指令外，也可以使用JSP中提供的<jsp:useBean>指令使用JavaBean。语法如下：

**<jsp:useBean id=”对象名” scope=”保存范围” class=”包.类”/>**

·id：表示对象的名称

·scope：表示此对象的保存范围（page、request、session、application）

·class：对象所对应的包.类名称，即这个对象时哪个类的实例

在使用<jsp:useBean>标签时，会默认调用JavaBean类中的无参构造方法进行对象的实例化，这就是为什么要在JavaBean类中定义一个无参构造的原因。<jsp:useBean/>标签可以自动实例化依靠的是反射机制完成的，也就是为什么JavaBean中要有一个无参构造方法和在使用<jsp:useBean/>标签时需要指定对象所在的“包.类”名称。

## 关于修改JavaBean需要重新启动服务器的问题

在进行代码的开发过程中，JavaBean肯定是需要不断修改的，但修改JavaBean后必须 重启服务器所作的修改才会生效。但这样一来非常的麻烦。为此我们可以使用自动加载。

在web.xml文件中的配置的虚拟目录<Context/>标签中加入reloadable属性，使其 值为true。如：

**<Context path=”/web” docBase=”d:\myWebProject” reloadable=”true” />**

这样每次修改JavaBean后服务器会自动加载，保存所作的修改。但是这种情况是适合于开发。因为使用reloadable自动加载后，服务器始终处于监视状态，一旦发现类修改后就要立刻重新加载，这样的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的内容设置为false，以提升服务器的运行性能。另外，当Tomcat服务器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

# JavaBean与表单

在JavaBean应用中，可以将JavaBean应用到表单的交互上，使用<jsp:useBean/>标签和<jsp:setProperty/>标签完成。即将表单的内容提交给JSP页面，在JSP页面使用<jsp:useBean/>标签声明JavaBean对象，再使用<jsp:setProperty/>标签传送过来的参数值设置为JavaBean对象的属性值（可以自动将字符串型自动转换为数值型）。表单的name属性值与JavaBean对象的属性名相同。

# 设置属性值<jsp:setProperty/>

<jsp:setProperty/>标签可以为JavaBean对象属性设置值，会自动调用JavaBean对象的setter方法。<jsp:setProperty/>标签可以方便的将表单提交的值设置为JavaBean对象的属性值。它一共有4种用法：

|  |  |
| --- | --- |
| **类型** | **语法格式** |
| 自动匹配（自省机制） | <jsp:setProperty name=”对象名” property=”\*”/> |
| 指定要匹配的属性 | <jsp:setProperty name=”对象名” property=”属性名”/> |
| 指定要匹配的属性和对应的参数名 | <jsp:setProperty name=”对象名” property=”属性名” param=”参数名”/> |
| 为指定的属性匹配指定的值 | <jsp:setProperty name=”对象名” property=”属性名” value=”值”/> |

name属性的值是使用<jsp:useBean/>标签定义的对象的名称，即id属性的值，表示使用这个对象接收参数。

自动匹配（自省机制）的方式使用最多，也是最方便的。

# 取得属性值<jsp:getProperty/>

使用<jsp:getProperty/>标签可以取得JavaBean对象指定的属性值 ，它会自动调用JavaBean对象的getter( )方法。与设置属性值标签<jsp:setPropety/>相比，取得属性值的标签只有如下一种语法：

**<jsp:getProperty name=”对象名” property=”属性名称”/>**

name属性的值是使用<jsp:useBean/>标签定义的对象的名称，即id属性的值，表示使用这个对象接收参数。

# JavaBean对象的保存范围

<jsp:useBean> 指令上存在一个scope属性，用于指定该JavaBean对象的保存范围。对象的保存范围有4种：

·page：保存在一页的范围中，跳转后此JavaBean对象无效

·request：保存在此一页中，服务器跳转后有效，但客户端跳转后无效

·session：在一个用户的操作范围中保存，重新打开浏览器才会声明新的JavaBean

对象

·application：在整个服务器上保存，服务器关闭后消失。

1）page范围的JavaBean只在本页有效，跳转后无效。

2）request范围的JavaBean只在本页有效，服务器端跳转后有效，客户端跳转后无效。

3）session范围的JavaBean对象任何页面中都有效。重新打开一个浏览器则无效。

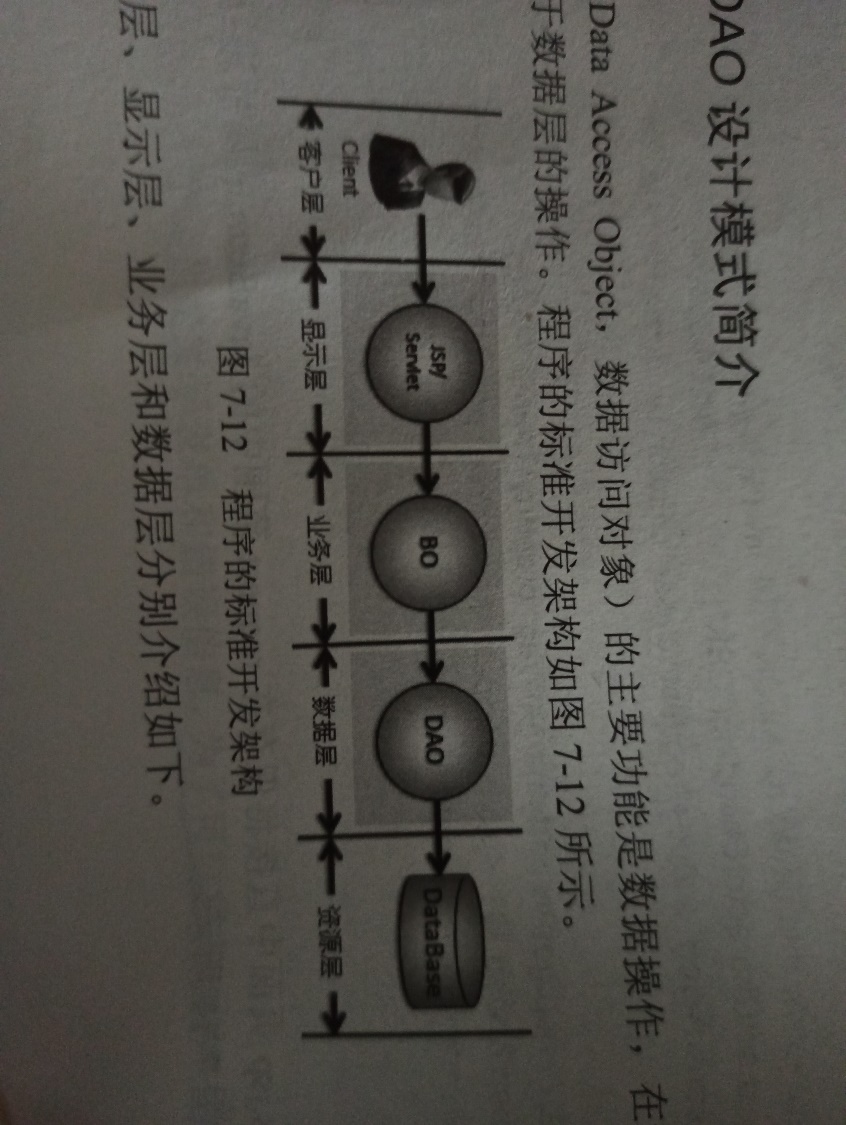
4）application范围的JavaBean对象是所有用户共同拥有的，是要声明后就会在服务器 中保存，所有的用户都可以直接访问此对象。除非服务器关闭后，JavaBean对象才 会消失重置。

# JavaBean对象的删除

JavaBean对象虽然是使用 <jsp : useBean>标签创建的，但是其操作核心依靠的仍然是4种属性范围的概念。如果一个JavaBean对象不再使用的话，则可以直

接使用4种范围对应内置对象的removeAttribute( )方法进行删除。

1. 删除page范围的JavaBean对象使用pageContext.removeAttribute( JavaBea对象名);
2. 删除request范围的JavaBean对象使用request.removeAttribute( JavaBean对象名);
3. 删除session范围的JavaBean对象使用session.removeAttribute( JavaBean对象名 );
4. 删除application范围的JavaBean对象使用application.removeAttribute( JavaBean对象 名);



# DAO设计模式

**1、介绍**

DAO（Data Access Object，数据访问对象）的主要功能是完成数据操作，在程序的标准开发结构中属于数据库操作。程序的开发标准开发结构如下图所示：

·客户层：因为现在都采用B/S开发模式，所以一般客户都使用浏览器进行访问，当然

也可以使用其他程序访问。

·显示层：使用JSP/Servlet进行页面效果的显示。

·业务层（Business Object，业务对象）：会将多个原子性的DAO操作进行组合，组合

成一个完整的业务逻辑。

·数据层（DAO）：提供多个原子性的DAO操作，如增加、修改、删除等，都属于原子

性的操作。

·资源层：存放数据的地方，一般是数据库。

书上说：对于一些大的系统，并且关联较多的系统，业务层才会发挥作用；而如果业务操作简单较为简单，可以不建立业务层，而完全通过DAO完成操作。但我不认为是这样，为了架构层的层次性，业务层还是有必要建立的，不管业务复杂与否。

**2、DAO设计模式的结构**

在整个DAO中实际上是以接口为操作标准的，即客户端依靠DAO实现接口进行操作，而服务端要将接口进行具体的实现。DAO由以下几个部分组成。

·DatabaseConnection类：专门负责数据库的打开与关闭操作的类。

·VO：主要由属性、setter、getter方法组成，VO类中的属性与表中的字段相对应，每

一个VO类的对象表示表中的每一条记录。

·DAO：主要定义操作的接口，定义一系列数据库的原子性操作标准，如增加、修改、

删除、查询操作。因为是接口，所以名称以大写I字母开头。

·Impl：DAO接口的真正实现类，完成具体的数据库操作，但是不负责数据库的打开与

关闭。

·Proxy：代理实现类，主要完成数据库的打开和关闭，并且调用真实实现类对象的操

作。

·Factory：工厂类，通过工厂类取得一个DAO接口的实例化对象。