javabean:jsp里面调用java类，此类就称为javabean

·javaBean的主要作用

·javaBean的定义及使用

·javaBean四种保存范围

具体内容：

几年前，有一句话，如果不会使用JavaBean，就等于说不说Jsp，在Jsp程序设计的时候，为了让其他开发人员（Asp）能够更好的适应，是Jsp 的语言非常简单。

Java可以对一些业务进行封转，而且Java是以类的形式存在的，可以形成一个一个的小组件，正常的开发模式：Jsp+Jdbc+JavaBean

Web的目录结构

WEB Root

·Web –Inf:是整个Web中最安全的目录

|-class：保存的是有的\* .class文件，所有的class文件必须放在此目录中

|-lib:第三方的\*.jar文件

|-web.xml:Web的部署描述符

·css：存放所有的\*.css文件

·js:存放所有的\*.js文件

·jsp：保存jsp的文件

·index.htm,index.jsp

javaBean的使用，

为了说明javaBean的使用，我们使用简单JavaBean，即类中只包含属性和方法

JavaBean的编写要求：

·命名必须规范

·javaBean有以下几点要求：

|-所有的javaBean必须存放在包中

|-类必须使用public class声明，文件名称要和类名称一致

|-类中的属性必须封装，被封装的属性要使用getter和setter方法调用

|-如果直接使用jsp提供的javaBean的语法，必须在javabean中提供一个无参的构造方法

使用JavaBean：

在Web中使用javaBean有两种方式完成：

·通过page指令导入

·通过jsp中定义的javabean的语法进行访问

使用第一种语法：

|  |
| --- |
| <%@page contentType=*"text/html; charset=GBK"*%>  <%@page import=*"com.wanczy.jjm.Person"*%>  <%  Person p = **new** Person();  p.setAge(30);  p.setName("姜建民");  %>  <h2><%=p.getAge() %></h2>  <h2><%=p.getName() %></h2> |

直接使用jsp中的提供的javabean的语法完成javabean的使用：

|  |
| --- |
| <%@page contentType=*"text/html; charset=GBK"*%>  <jsp:useBean id=*"p"* class=*"com.wanczy.jjm.Person" scope=”page”*></jsp:useBean>  <%  p.setAge(30);  p.setName("姜建民");  %>  <h2><%=p.getAge() %></h2>  <h2><%=p.getName() %></h2> |

以上两种语法运行效果是一样，唯一的不同就是第二种使用jsp中的语法支持，我们来解释以下：

·<jsp:useBean id=*"p"* class=*"com.wanczy.jjm.Person" scope=”page”*>

|- jsp:useBean：表示在jsp 中要是用JavaBean

|- id：表示生成的实例化对象名称，以后凡是在标签中看到了id，就表示实例化了一个对象

|- class：此对象对应的javaBean的路径和名称，即 包。类

|- scope:表示此javaBean的保存范围，有四种：page,request,session,application

说明一：

JavaBean每次修改之后都必须重启服务器，因为重启了服务器之后才会重新加载这个class文件。

每次修改javaBean需要重启服务器，太麻烦了，为了解决这样的问题，在tomcat中提供了自动的加载功能，tomcat会侦查一个类是否改变了，如果改变，会立即加载

|  |
| --- |
| <Context path="/demoWeb3" docBase="F:\Workspaces\newWorkspace\DemoWeb03\WebRoot" reloadable="true"/> |

但是这样配置有一个问题：因为每次要重新加载，容器中的全部内容会清空，如果有session的话，就不再存在了。

说明二：

对于以上的javaBean（Person）文件，有以下几个名词，必须记住：

·POJO：最原始，最古来的定义类的统称

·VO：bean中为每个属性赋值

·TO：对象传输，实现了序列化接口的类

javaBean的应用：

之前的所有的操作，我们都会感觉没有任何必要，不是用javaBean的语法也可以完成程序，那么javaBean的最大的好处在哪里呢？

在于javaBean的参数传递上。

接收参数：

在表单中写入输入框，传递参数，观察javaBean 的语法特点：

之前方式接收参数是否存在问题呢？如果现在表单中的参数非常多，有一百个，一个一个的接收会显得非常的费劲，下面通过javaBean的语法完成

|  |
| --- |
| <%@page contentType=*"text/html; charset=GBK"*%>  <jsp:useBean id=*"p"* class=*"com.wanczy.jjm.Person"* scope=*"page"*></jsp:useBean>  <%  request.setCharacterEncoding("GBK");  %>  <jsp:setProperty property=*"\*"* name=*"p"*/>  <h2><%=p.getName() %></h2>  <h2><%=p.getAge() %></h2> |

设置javaBean的属性：

javaBean定义完成之后，可以直接使用设置属性的方式，为Bean中的内容赋值，设置属性的语法的有四种：

第一种：通过自省方式设置属性，通常就是使用这一种方式：

|  |
| --- |
| <jsp:setProperty property=*"\*"* name=*"p"*/> |

·name：对应javaBean的id

·porperty=”\*”：四种语法要改变的位置就是此处，表示根据参数的名称自动进行设置

第二种：

|  |
| --- |
| <jsp:setProperty property=*"具体的一个属性"* name=*"p"*/> |

|  |
| --- |
| <%@page contentType=*"text/html; charset=GBK"*%>  <jsp:useBean id=*"p"* class=*"com.wanczy.jjm.Person"* scope=*"page"*></jsp:useBean>  <%  request.setCharacterEncoding("GBK");  %>  <jsp:setProperty property=*"age"* name=*"p"*/>  <h2><%=p.getName() %></h2>  <h2><%=p.getAge() %></h2> |

第三种：给指定的属性设置参数：

|  |
| --- |
| <jsp:setProperty property=*"具体的一个属性"* param=”参数名称” name=*"p"*/> |

|  |
| --- |
| <%@page contentType=*"text/html; charset=GBK"*%>  <jsp:useBean id=*"p"* class=*"com.wanczy.jjm.Person"* scope=*"page"*></jsp:useBean>  <%  request.setCharacterEncoding("GBK");  String name = "Wanczy";  %>  <jsp:setProperty property=*"name"* param=*"age"* name=*"p"*/>  <jsp:setProperty property=*"age"* param=*"age"* name=*"p"*/>  <h2><%=p.getName() %></h2>  <h2><%=p.getAge() %></h2> |

第四种：给指定的属性设置值

|  |
| --- |
| <jsp:setProperty property=*"具体的一个属性"* value=”指定的值” name=*"p"*/> |

|  |
| --- |
| <%@page contentType=*"text/html; charset=GBK"*%>  <jsp:useBean id=*"p"* class=*"com.wanczy.jjm.Person"* scope=*"page"*></jsp:useBean>  <%  request.setCharacterEncoding("GBK");  String name = "Wanczy";  %>  <jsp:setProperty property=*"name"* value="<%=name %>" name=*"p"*/>  <jsp:setProperty property=*"age"* value=*"10"* name=*"p"*/>  <h2><%=p.getName() %></h2>  <h2><%=p.getAge() %></h2> |

综合以上开看，使用“\*“的第一种方式是最好的。

javaBean的范围：

在java中，同样保存了四种范围，而且四种范围是不能修改的，即：一个javaBean如果定义成了page范围，则最好不要修改成其他范围。

以下我们以另外一个javaBean来说明问题：

Count.java

|  |
| --- |
| **package** com.wanczy.jjm;  **public** **class** Count {  **private** **int** count;  **public** **int** getCount() {  **return** count;  }  **public** **void** setCount(**int** count) {  **this**.count = count;  }  **public** Count(){  System.*out*.println("实例化对象");  }  } |

page范围：

只在当前页面保存：

|  |
| --- |
| <%@page contentType=*"text/html; charset=GBK"*%>  <jsp:useBean id=*"count"* class=*"com.wanczy.jjm.Count"* scope=*"page"*></jsp:useBean>  <%  System.out.println("-------------------------------------------------");  %>  <h2>访问了<font color=*"red"* size=*"10"*><%=count.getCount() %></font>此</h2> |

没刷新一次，Count类会重新实例化

request范围：

一次服务器跳转有效

|  |
| --- |
| <%@page contentType=*"text/html; charset=GBK"*%>  <jsp:useBean id=*"count"* class=*"com.wanczy.jjm.Count"* scope=*"request"*></jsp:useBean>  <%  System.out.println("-------------------------------------------------");  %>  <h2>访问了<font color=*"red"* size=*"10"*><%=count.getCount() %></font>此</h2>  <jsp:forward page=*"javaBean09.jsp"*></jsp:forward> |
| <%@page contentType=*"text/html; charset=GBK"*%>  <jsp:useBean id=*"count"* class=*"com.wanczy.jjm.Count"* scope=*"request"*></jsp:useBean>  <%  System.out.println("-------------------------------------------------");  %>  <h2>访问了<font color=*"red"* size=*"10"*><%=count.getCount() %></font>此</h2> |

session 范围：

|  |
| --- |
| <%@page contentType=*"text/html; charset=GBK"*%>  <jsp:useBean id=*"count"* class=*"com.wanczy.jjm.Count"* scope=*"session"*></jsp:useBean>  <%  System.out.println("-------------------------------------------------");  %>  <h2>访问了<font color=*"red"* size=*"10"*><%=count.getCount() %></font>此</h2>  <a href=*"javaBean11.jsp"*>session范围</a> |
| <%@page contentType=*"text/html; charset=GBK"*%>  <jsp:useBean id=*"count"* class=*"com.wanczy.jjm.Count"* scope=*"session"*></jsp:useBean>  <%  System.out.println("-------------------------------------------------");  %>  <h2>访问了<font color=*"red"* size=*"10"*><%=count.getCount() %></font>此</h2> |

application范围：

服务端设置javaBean

|  |
| --- |
| <%@page contentType=*"text/html; charset=GBK"*%>  <jsp:useBean id=*"count"* class=*"com.wanczy.jjm.Count"* scope=*"application"*></jsp:useBean>  <%  System.out.println("-------------------------------------------------");  %>  <h2>访问了<font color=*"red"* size=*"10"*><%=count.getCount() %></font>此</h2>  <a href=*"javaBean13.jsp"*>session范围</a> |
| <%@page contentType=*"text/html; charset=GBK"*%>  <jsp:useBean id=*"count"* class=*"com.wanczy.jjm.Count"* scope=*"application"*></jsp:useBean>  <%  System.out.println("-------------------------------------------------");  %>  <h2>访问了<font color=*"red"* size=*"10"*><%=count.getCount() %></font>此</h2> |

在javabean中进行数据库连接操作：

一般情况下把重复的代码写在javaBean中。

|  |
| --- |
| **package** com.wanczy.jjm;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.SQLException;  **public** **class** DateBaseConnection {  **public** **static** **final** String *DRIVER*="oracle.jdbc.driver.OracleDriver";  **public** **static** **final** String *URL* = "jdbc:oracle:thin:@192.168.10.250:1521:wance";  **public** **static** **final** String *DATEBASENAME*="system";  **public** **static** **final** String *DATEBASEPASSWORD*="wance";  **private** Connection conn;  **public** DateBaseConnection(){  **try**{  Class.*forName*(*DRIVER*);  conn = DriverManager.*getConnection*(*URL*,*DATEBASENAME*,*DATEBASEPASSWORD*);  }**catch**(Exception e){    }    }  **public** Connection getConn() {  **return** conn;  }  **public** **void** setConn(Connection conn) {  **this**.conn = conn;  }  **public** **void** close(){  **try** {  **this**.conn.close();  } **catch** (SQLException e) {  }  }  } |

总结：

·介绍了javabean的基本使用，应该将各个重复的代码写在一个小的javaBean中，以方便调用

·javabean的形成是靠开发过程中反复的总结，提炼经常使用到的一些代码

·了解javabean的语法，并且了解属性赋值的方式，里面是通过反射机制给类中的参数赋值

·一个标准的javabean应该有无参构造和getter以及setter方法

·javaBean的接收参数，是可以接收任意类型的参数，并且可以自动转换，但是在使用的时候要注意一点，类型转换的异常