基于反射和数据绑定机制的 WEB 动态查询研究①

Research of Web Dynamic Query Based on Reflection and Data Binding

金 弟　周晓慧　宋媛媛　（杭州电子科技大学 自动化学院 310018）

摘要：本文在分析了利用基于 Weblogic 平台的 JAVA 反射机制和 Workshop 数据绑定机制开发 WEB 动态查询的优越性基础上，提出了一种解决方案。结合印染商务信息系统的实践开发为例，介绍了这种思想、方法的具体实现。

关键词：反射机制 数据绑定 动态查询

1 引言

基于 B/S 架构 WEB 应用正在成为商业应用的主流，为了满足不断变化的业务需求，对 WEB 应用的可维护性、可扩展性、通用性提出了更高的要求。查询功能是开发 WEB 应用不可缺少基本公用模块，当原有的查询需求变化、增加新的查询对象和查询结果需要修改时，就必须要修改代码或重新设计。为了解决这些问题，本文提出了一种基于 Weblogic 平台的 JAVA 反射和 workshop 数据绑定机制设计方法来指导 WEB 查询设计，该方法通过利用 Workshop 数据绑定和 JAVA 反射机制的特点、优点，进行两者结合来实现查询对象的动态绑定、动态生成查询条件和动态绑定查询结果集的显示。

2 JAVA 反射与 Workshop 数据绑定机制

2.1 JAVA 反射机制

在计算机领域内，反射是指通过采用某种机制来实现对自己行为描述，检测，并能根据自身行为状态和结果，调整或修改应用所描述行为的状态和相关的语义。反射在语言上体现为动态性，Java 反射机制在一定程度上实现了动态功能，主要提供如下功能：

（1）运行时识别对象及其类型；

（2）运行时创建对象，根据类名，动态创建类实例；

（3）运行时提取对象元信息包括类型、是否抽象、构造器、方法、属性、超类、接口、内部类等各方面信息。为建立通用性和灵活性更强的程序提供了很好的支持。

java. lang. reflect 包和 java. lang. class 类对反射机制的具体的实现提供了编程接口。java. lang. reflect 包 3 个类 Field、Method 和 Constructor 相应地描述了一个类的字段、方法和构造器。这 3 个类都有一个称作 getName 方法，它返回相应的名字；Field 类有一个 getType 方法，返回描述字段的 Class 类型的对象；而 Method 和 Constructor 类都有返回类型和方法参数类型方法。这 3 个类都有 getModifiers 方法，它返回一个整数，通过设置不同的位来描述使用的修饰符。Class 类的 getFields、getMethods 和 getConstructor 方法分别返回类支持的公开字段、方法和构造器的数组；Class 类的 getDeclaredFields、getDeclaredMethods 和 getDeclaredConstructors 方法返回所有的字段、方法和构造器的数组，其中包括私有和受保护成员，但不包括超类成员。JAVA 反射机制的以上优点和提供的丰富 API 为实现动态查询提供了优越性条件。

2.2 Workshop 数据绑定机制

Workshop 数据绑定机制是将用户界面中的 UI 标签绑定到 Web 的应用数据，由此创建易于生成和维护的动态 Web 应用程序，为构建基于 Web 的用户界面

① 基金项目：浙江省科技厅重大攻关项目：2003C11043

提供了一个开发模型。Workshop 提供的数据绑定标签和向导使得开发人员可以非常方便地使用 Java 页面流创建丰富的动态页面。这些功能使得 UI 标签可以被绑定到各种数据上下文(不局限于表单 bean),而且数据可以来自任何地方包括数据库、web 服务、EJB、自定义应用等等。借助数据绑定标签,开发人员方便地绑定各种数据,其中包括复杂的类型,比如重复数据、列表、树、表格等等。各种 UI 标签绑定来自各种数据绑定上下文的数据。要使用 UI 标签进行数据绑定,要引用数据绑定上下文名称,例如

```
<netui:form action="ChangeZipCode">
<netui:textboxdataSource="{actionForm.postalCode}"/><netui:button>Submit</netui:button>
<netui:form>
```

表 1　数据绑定上下文

上下文名称	上下文引用的对象	描述
pageFlow	当前页面流	引用当前页面流的控制器类定义的属性
globalApp	GlobalApp 全局对象	引用 Global.app 中定义的属性
actionForm	netui:form 标记引用操作表单 Bean	可以引用在表单 Bean 中定义的属性
session	会话上下文中的特性	session 引用代表单独的用户会话的 JSP 会话对象
container	复杂数据集中的项目	引用由许多复杂数据绑定标记(如 repeater 标记)引用的数据
pageContext	页面流控制器类中定义成员变量	使用 pageInput 上下文显示在页面流控制器类中定义的成员变量的值
bundle	消息资源文件中定义的属性	bundle 上下文引用在消息资源文件中定义的属性
pageInput	页面流控制器类中定义成员变量	使用 pageInput 上下文显示在页面流控制器类中定义的成员变量的值
application	应用程序上下文	application 引用代表 JSP 所属的应用程序 JSP 应用程序对象
request	触发 JSP 的加载请求	引用操作方法内定义为请求的一部分特性
url	当前 JSP 的 URL 的查询参数	引用作为 URL 的一部分的查询参数的

在此示例中,netui:textBox 标记上的 dataSource 特性引用表单 Bean 中的变量 postalCode。表单 Bean 是使用上下文名称 actionForm 进行数据绑定关联的。所有数据绑定上下文元素都可以使用"点"符号进行访问,而且整个表达式必须始终用括号括起来。表 1 给出了各种数据绑定上下文的概述。

数据绑定机制提供丰富的数据绑定上下文以及 Web 界面和 Web 数据的动态有机结合和分离为创建易于生成和维护的动态 Web 查询模块提供了很好的支撑。

2.3　反射与数据绑定机制的结合思想

通过不同的类来封装不同查询对象和查询结果集,利用 Workshop 数据绑定机制把 Web 应用程序的查询界面的数据动态的与查询对象绑定,使用 JAVA 反射机制在 Web 应用程序运行时动态获取查询对象中的信息进行匹配,动态生成查询 SQL 语句。从而实现了查询对象的实时绑定和查询条件的动态生成,最后再通过 Workshop 数据绑定机制进行查询结果对象的绑定,以便动态的进行显示。下面以印染商务信息系统的一个查询印花 A4 小样为例进行具体阐述。

3　应用实例

(1) 查询界面

图 1 为查询界面,数据绑定主要代码如下:

```
<th valign="middle">小样编号:</th>
<netui:textBox dataSource="{actionForm.sampleID}"/>
<th valign="middle">规格:</th>
<netui:textBox dataSource="{actionForm.specification}"/>
<th valign="middle">品名:</th>
<netui:textBox dataSource="{actionForm.productNameName}"/>
<th valign="middle">审核状态:</th>
<netui:select dataSource="{actionForm.checkStatus}"
optionsDataSource="{pageFlow.checkStatus}"/>
<th valign="middle">送样开始时间:</th>
<netui:textBox dataSource="{actionForm.start_incomeDate}" onDblClick="calendar()"/>
<th valign="middle">至:</th><netui:textBox dataSource="{actionForm.end_incomeDate}" onDblClick="calendar()"/>
```

程序中使用上下文名称 actionForm 进行数据绑定关联的。所有数据绑定上下文元素都可以使用“点”符号进行访问，而且整个表达式必须始终用括号括起来。actionForm 对应于要绑定的查询对象，这个对象使用一个从 FormData 继承的 Bean 类进行具体封装。数据绑定机制保证界面的数据和查询对象之间能够始终保持数据同步和一致，而且 ActionForm 自动调用这个 Bean 类的 getXXX 函数进行初始化。pageFlow 是对当前页面流的成员变量 checkStatus 进行绑定，从而完成页面流和页面之间动态的数据传递及交换。

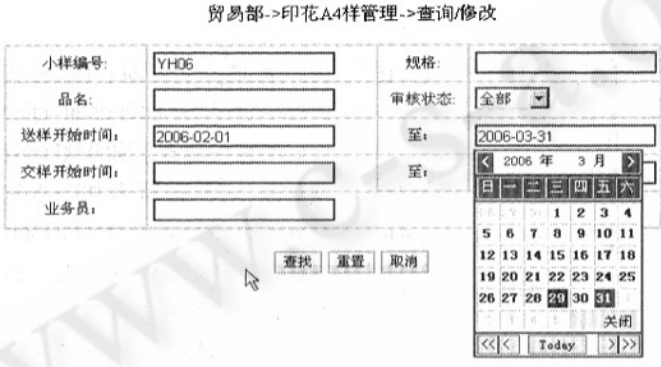

图 1　使用数据绑定的查询界面

（2）处理查询提交

输入相关查询条件，进行提交后，进入页面流的 FindResultAction 中，同时也绑定了相应的 A4 小样查询对象 A4Sample，它是 FormData 继承的 Bean 类，数据绑定机制对界面数据处理映射成对查询对象 A4Sample 处理。

```
protected Forward findResultAction(A4Sample form) {
//可以先对 form 中的数据进行处理再执行后继操作;
    findA4Sample = form; //用于保存查询条件
    findA4Sample.setSampleType(sampleType);
//把查询对象 form 作为参数传递以便动态生成语句
    allData = myControl.findA4Sample(form);
    for (int i = 0; i < allData.length; i++) { if
(allData[i].getCheckStatus() == 1) allData[i].
setStrCheckStatus("待审核"); else allData[i].
setStrCheckStatus("已审核"); } return new For-
ward("success"); }
//自定义控件 myControl 成员函数 findA4Sample 实现
public A4Sample[] findA4Sample(A4Sample sample)
{
String sqlClause = ComDBUtil.createQuery(sam-
ple,"");
//提交动态生成的 SQL 语句给数据库管理系统
    return a4SampleForm.find(whereClause);
}
```

（3）动态查询条件生成

包 ComDBUtil 的成员函数 createQuery 利用 JAVA 反射机制获取实时查询对象 form 中的信息，进行动态查询条件的生成。

```
Public static String createQuery(Object ob, String
prefix) {
//动态获取查询对象的类信息
    Class clazz = ob.getClass();
//动态获取查询表名
String tableName = getLastString(clazz.getName
(),".");
ArrayList strArray = new ArrayList();
//动态获取查询对象的查询字段
  Field[] fld = clazz.getDeclaredFields();
  for (int i = 0; i < fld.length; i++) {
 if (!fld[i].isAccessible())
fld[i].setAccessible(true);
  try {
    Object fldValue = fld[i].get(ob);
//根据数据绑定时的初始化确定查询条件是否空缺
    if (isInitValue(fldValue)) continue;
//获取组合字段的内容
    strArray.add(new String(getFieldSelSql(ob, f ld
[i], prefix)));
    } catch (IllegalArgumentException e) {
    e.printStackTrace();
    } catch (IllegalAccessException e) {
    e.printStackTrace();
    }
 } //for 循环结束
if ((strArray == null) || strArray.isEmpty() ||
(strArray.size() == 0)) return "";
StringBuffer result = new StringBuffer();
```

```
if ( prefix. equals( "" ) ) result. append( " where " ) ;
//根据查询字段及其内容拼接成最终查询条件
result. append ( StringUtils. join ( strArray. toArray ( ) , "
and " ) ) ;
return result. toString( ) ;
}
```

（4）查询结果集显示

如图 2 为一个查询结果集显示界面，数据绑定主要代码：

贸易部->印花A4样管理->查询/修改

查看	编辑	删除	打印预览	A4小样编号	送样时间	交样时间	业务员	品名	审核状态
		删除	打印预览	YH0603001	2006-03-07	2006-03-07	黄华林	14条T/C灯芯绒印花	已审核
		删除	打印预览	YH0602163	2006-02-27	2006-02-27	黄华林	14条T/C灯芯绒印花	已审核
		删除	打印预览	YH0602128	2006-02-27	2006-02-27	黄华林	21条灯芯绒	已审核
		删除	打印预览	YH0602123	2006-02-27	2006-02-27	高爱媛	11条灯芯绒	已审核
		删除	打印预览	YH0602118	2006-02-27	2006-02-27	沈祥华	T/C府绸印花	已审核

打印全部预览　返回

分页： 1
转到：　提交 第1页 共1页 共5条数据

图 2　使用数据绑定的查询结果集界面

```
//调用页面流的方法 getdata 把查询对象结果集数//
组 A4Sample[ ]中的数据进行分页处理。处理结果//
通过数据绑定机制自动存储于绑定对象//pageCon-
text 的属性 data 中
<netui - data: callPageFlow method = " getdata" re-
sultId = " data" >
//每组显示 10 页
<netui - data: methodParameter value = " 5" / >
//每页显示 5 条记录
<netui - data: methodParameter value = " 10" / >
//通过数据绑定上下文 pageContext 获取显示数据
<netui - data: repeater dataSource = " { pageContext.
data} " ignoreNulls = " true" >
//使用循环 UI 标签显示数据
<netui - data: repeaterItem >
//显示项通过数据绑定上下文 container 获取，其中//
container. item 的结构与 data 的一致
<netui: label value = " { container. item. sampleID} "
defaultValue = " " / >
<netui: label value = " { container. item. incomeDate} "
defaultValue = " " / >
<netui: label value = " { container. item. submitDate} "
defaultValue = " " / >
</netui - data: repeaterItem >
```

以上实现的方法具有很好的灵活性，解决了查询业务需求变化需要修改代码的问题，例如增加一个查询字段，查询逻辑处理模块 findResultAction 代码不需要修改。如果新增查询对象，只要把新增加的查询对象作为参数调用 createQuery 方法，也不需要修改查询逻辑处理代码。同样对数据查询结果的显示内容根据业务需求获取数据绑定上下文的数据项无需修改代码。

4　结束语

查询功能在基于 Web 信息管理系统中是必不可少的基本公用功能，在印染商务信息系统中的应用实践表明使用本文介绍方法实现的查询模块可扩展性强、维护性好，具有一定的通用性，具有工程应用价值。

参考文献

1　孙巍、徐学东，用 Java 反射机制在可重构 Web 框架中的应用，计算机工程与应用，2005 年第 3 期。

2　Cay S. Horstmann, Gary? Cornell, Core Java™ 2 Volume I – Fundamentals, Seventh Edition [M], Prentice Hall PTR, 2004。

3　戚欣、王静，在 Struts 架构中运用类反射机制，武汉理工大学学报，2005 年第 2 期。

4　http://e – docs. bea. com/workshop/docs81/doc/en/core/index. html