第 33 卷 第 9 期 计 算 机 工 程 2007 年 5 月
Vol.33 No.9 Computer Engineering May 2007
·软件技术与数据库· 文章编号：1000—3428(2007)09—0072—02 文献标识码：A 中图分类号：TP311

基于 JavaBeans 的软件构件复用技术研究

许 峰，丁 珂，王志坚
(河海大学计算机及信息工程学院，南京 210098)

摘 要：软件构件技术作为一种软件复用技术，旨在解决软件系统开发所面临的困境。大多数构件组装工具都是基于某种特定的构件模型，选取 JavaBeans 构件模型作为研究对象，分析了其与软件复用相关的特征，探讨了其对软件复用的支持程度，并对其中的一些特征进行了扩展。

关键词：构件；构件模型；构件组装工具；自描述

Research of Software Component Reuse Technology Based on JavaBeans

XU Feng, DING Ke, WANG Zhijian
(College of Computer & Information Engineering, Hohai University, Nanjing 210098)

【Abstract】 As a technology for software reuse, component technology aims at these challenges. Component-based composition tool is used for software reuse, and makes user to develop application more easily. Composition tool is based on some kinds of component model. Now, there are two kinds of commerce component model. One is COM, which belongs to Microsoft and based on OLE. Another is Ejb/JavaBeans, which belongs to Sun. Composition tool is used for software reuse, so when designing a composition tool, the first thing which should be thought about is how to make software reuse more easily. This paper analyzes the JavaBeans, gets the features related to software reuse, extends the model, and makes it more easily for reuse. The ability of self-description is extended.

【Key words】 Component; Component model; Component-based composition tool; Self-description

1 概述

基于构件的应用系统开发技术是软件复用思想的具体体现，是软件产业化的迫切需要。构件复用[1]可分为两部分：可复用构件的开发(Development for Reuse)和基于可复用构件的系统构造(Development with Reuse)。前者促成构件生产的专业化分工，后者有利于软件的批量化生产，而这些正是软件产业化的基本特征。

可复用构件的开发和相应应用系统的构造一般都是基于某个特定构件模型，如 JavaBeans、COM 等。当前对于 JavaBeans 的构件复用技术虽然已经具有商业化标准，但还存在一些不足，如构件自描述(Self-description)不够充分等。随着构件技术的发展，出现了很多基于可复用软件的应用系统构造工具，包括实验性和商业性的产品，如基于 JavaBeans 的 Bean Builder。

1.1 基于构件的开发方法

软件构件的出现改变了传统的软件开发方法，基于构件的开发方法(Component-based Development, CBD)建立在过去长期的结构化方法中的模块系统，以及近 10 年来的面向对象系统的基础上的。通过标准化运行级构件的规约，依靠构件运行平台(中间件平台)提供的基础设施，CBSD提供了一种自底向上的、使用标准软件构件构造系统的有效途径，并得到了广泛的应用[2]。

1.2 构件模型与组装

构件易复用的特点要求构件在组装过程中扮演积极的角色，能够半自动或自动地完成组装任务。而构件大粒度复用的特点又使构件的主动组装成为可能，一方面构件有足够的空间封装用于组装的功能和数据，另一方面只有大规模的复用才能使因组装而引起的效率损失变得可以接受，这也是传统的小粒度复用单位(unit)无法面向组装的原因。

构件面向组装的本质是从传统复用技术所注重的静态组装向动态组装的转变。静态组装以组装工具的使用为主要特征。当前基于控件的 GUI 构造和 4GL 工具可以称得上是组装工具的成功典范，确实降低了应用系统构造的复杂度。然而，这些工具还远未达到普通用户能够轻松掌握和熟练使用的程度。动态组装以构件模型、构件构架(architecture)的建立和标准化以及开放系统技术的运用为主要特征。与静态组装相比，动态组装虽然牺牲了应用系统的部分运行效率，但动态组装具有很高的灵活性，并能实现构件组装的自动化。

2 JavaBeans 的构件模型

Sun对JavaBeans的定义是[2]：JavaBeans是一个可重复使用的软件部件，该部件可以用来生成其它可视化的处理。一个构件即Java应用程序或Applet的可重复使用的部件。

应用程序构件模型是一个部件结构和执行结构的 APIs。JavaBeans 应用程序构件模型十分简单：它由模式的基本元件、这些元件的属性及支持这些属性的服务组成。

2.1 JavaBeans 模型定义的两个元件

构件：构件是应用程序可重复使用的部分。Bean 是能用

基金项目：国家“973”计划基金资助项目(2002CB312002)；江苏省高技术项目(BG2005036)
作者简介：许 峰(1975－)，男，讲师、博士生，主研方向：分布式计算，软件体系结构；丁 珂，硕士生；王志坚，教授、博导
收稿日期：2006-05-08 **E-mail：**njxufeng@163.com

可视应用生成器修改的一个构件。Bean 可能是轻型(如按钮)、中型(数学公式编辑、程序)和重型(复合文件系统)。通过提供的 JavaBeans APIs，构件设计变得很容易。

容器：在构件上下文中，容器是相关元件的组装和集合。容器可以是常规应用程序、组合文件、可视生成器、网页等。容器可以相互嵌入：一个容器可以拥有若干容器，而这些容器又可以具有 Bean。Bean 也可以是容器：一个 Bean 能包含具有更简单特性的 Bean。在应用程序中一个容器内包含另一个容器的例子是嵌在字处理器中的绘图程序。

2.2 JavaBeans 的特性、服务和目标

JavaBeans有 6 个特性[3]，其中前 3 个(属性、事件、方法)是面向对象的原则特性，JavaBeans是面向对象的，可以使用属性和方法，出现的事件表明构件之间如何工作和构件如何与容器工作。后 3 个特性(一致性/串行化、自我检查/反射、定制)使JavaBeans成为构件。

服务是允许 Beans 在彼此之间以及在容器之间交互的底层支持，JavaBeans 所提供的服务有：构件属性接口，构件行为接口，事件管理，构件配置，构件组装，可视生成器接口。

JavaBeans 的目标包括便携性、轻量、易于创建和能嵌入其它构件模型。

2.3 JavaBeans 的自描述能力

标准 JavaBeans 模型中，构件(Bean)的自我描述能力主要通过两种方法实现：Java 语言的发射机制(reflect)和 JavaBeans 模型中的自省机制(introspection)。

Java 语言是标准 JavaBeans 构件模型存在的基础。在 Java 语言中，提供了一种机制，可以在运行时获得类和接口的信息。每个 Java 类都必须继承 java.lang.Object，其中的方法 getClass 可以返回运行中的类型对象(Class Object)，通过该返回值可以获取更多的类型信息。

自省机制(Introspection)是 JavaBeans 模型提供的另外一种自描述方法，通过它 Bean 可以实现自描述。这种自描述能力是通过“ 命名和类符号规定 ”和定义类 BeanInfo 来实现的。

2.4 JavaBeans 的组装环境

在面向对象的开发方法中，出现了一种新的角色，即应用程序组装者。传统的应用程序开发员进行组件的开发，而应用程序组装者则通过这些组件来构建新的应用[4]。这种开发方式的优点就是应用程序组装者可以更多注意与领域相关的问题，而较少关注技术的实现细节。对于应用程序组装者来说，一种构件组装工具是必要的，通过这个工具就不再需要大量编码。

3 基于 Bean Box 的构件模型

3.1 Bean Box 的结构

Bean Box 是 Sun 公司提供的 JavaBeans 构件组装工具，Bean Box 主要由 3 部分组成：BeanBoxFrame，ToolBox 和 PropertySheet。BeanBoxFrame 包含了 BeanBox 的实例，用于显示当前 bean 实例的结构，用户可以在其中进行添加、修改和删除 bean 实例操作，并且可以关联 bean 实例的属性、事件，甚至可以保存当前组装好的应用程序，并在需要的时候加载。Bean Box 提供了一些标准 bean，这些都显示在 ToolBox 上，用户可把它们拖放至 BeanBoxFrame 中，方便程序开发。

Bean Box 由 47 个类构成(不包括内隐类)，这些类可以被分成以下几个部分：核心类，GUI 相关类，文件输入输出相关类，与 Infobus 通信的类，小型工具类和接口。

BeanBoxFrame 类实现了 Bean Box 的主窗口，但只能保持一个实例存在；它主要处理初始化和构件焦点的问题。类 BeanBox 和类 Wrapper 是 Bean Box 中最重要的类，包含了与组装相关的大多数功能。BeanBox 是 bean 实例的容器，控制了它所处 frame 的菜单，并处理添加新 bean 实例的操作、剪贴板操作以及对 bean 实例的图形界面进行放大缩小等操作；它本身是一个 AWT 构件。Wrapper 也是一个 AWT 构件，定义了 bean 实例在设计期的行为；它也可以包含一个 BeanBox 实例。主窗体(BeanBoxFrame)包含了一个 BeanBox 类的实例，该实例被 Wraper 类包装。BeanBox 能包含 Wrappers，每个 Wrapper 包含了一个 bean 实例。

Bean Box 核心的功能是由 Adapter 类完成的，用于管理 bean 实例之间的关联。对于每个事件关联，存在一个 Adapter 类的对象，同时扮演事件侦听和事件源的角色。通过这种方法，Bean Box 可以处理任意多个事件和方法之间的关联。属性间的关联关系也使用相似的方法，相互之间的通信通过事件实现；方法和方法之间的关联以及其它类型间的关联在 Bean Box 中不能可视化地被创建。

3.2 Bean Box 的视图

Bean Box 的标准视图是面向对象实例的，在它里面对象实例以可视化的方式展现，用户可以直接连接这些实例生成应用程序。在这个视图里，每个 Bean 都可以以不同的实例多次出现，而且每个实例都有自己的属性，可以被独立修改。

这种结构视图对组装静态视图是非常有用的，但一些动态结构(即在运行时生成的对象)不能被显示出来。它也存在其它的一些不足，如不能显示组装成的应用程序的全部结构，不能看出哪些构件是基本的构件，构件之间的依赖关系不能被直观地展示出来，应用程序的组装者不能知道哪些构件可以不采用。

Bean Box 中存在的不足可以通过对标准 JavaBeans 模型进行扩展解决，每个 Bean 文件包含更多的自描述信息，包含更多描述构件间关系的信息。

4 基于 Bean Box 的构件模型的拓展

如果一个构件在一个应用程序中经常被使用到，那么实例视图将变得非常复杂，几乎不可能看出那些构件实例所用到的场景，也不可能有方法去约束构件实例。基于这些原因，可以采用结构视图扩展 Bean Box。它是基于类型的，每个 bean 实例在其中最多只能显示一次；并且显示出 beans 之间的依赖关系。其目的是为组装应用程序提供帮助，使得组装者能够浏览应用程序的结构；构件组装者能够看出构件间的依赖关系，以及构件之间是否存在交互。在组装完应用程序后，应该可以通过结构视图运行该程序。

结构视图能通过 BeanBoxFrame 的下拉菜单“ Extra/Architecture ”来打开。首先，结构视图将显示所有 Bean Box 加载的扩展 beans，即实现了接口 ExtendedBeanInfo 的 beans(一般是在“ jars ”目录下的 bean)。那些标准 beans 不能被显示在结构视图中。每个 bean 都以一个图标和名字显示，如果没有显示图标，那么将以默认的图标显示出来。

在 beans 加载后，结构视图会自动生成接口间的联系，这些联系可以通过工具进行修改。这种接口间的关系使得用户可以很清楚地看出 bean 在应用程序中的可移性，如果有其它 bean 在使用它，那么它就不能随便移动。

在结构视图的下拉菜单中还存在着“ 文件 ”、“ 编辑 ”、“ 应用 ”3 个子菜单。“ 文件 ”菜单中包含“ 加载 ”和“ 保存 ”子

（下转第 105 页）