Java 组件技术的应用研究
Application of Java Component Techniques

吕立坚,李静滨,赵 明,文 静
Lü Lijian, Li Jingbin, Zhao Ming, Wen Jing
(广西大学计算机与信息工程学院,广西南宁 530004)
(College of Computer and Information Engineering, Guangxi University, Nanning, Guangxi, 530004, China)

摘要: 从组件体系结构和组件模型理论上阐述 Java 组件在软件开发体系结构下实现的基本原理和方法。在服务器端编写 Java 组件,用 1 个实例实现在客户端上自由调用服务器端的 Java 组件进行文件加解密的操作。

关键词:Java,组件,JavaBeans,Java Bean
中图法分类号:TP316.2

Abstract: The basic knowledges of component architecture systems and component models, as well as implement of Java Beans in JavaBeans are explained. The operation of encrypting and decrypting files in client of Java Beans of sever freely is released.

Key words: Java, components, JavaBeans, Java Bean

近年来,随着互联网应用的普及,涌现了大量的互联网新技术,如 EJB、CORBA、Servlet、JSP、网格计算及 Web services,尤以 Web services 技术成为当今互联网新技术的焦点。在 Web services 中,应用了大量面向对象的组件技术,组件成为互联网上的最小应用单元。它的出现,改变了传统的网络应用程序开发模式,并得到了迅速的发展。同时,软件开发人员希望能使用一种网络式跨平台的软件开发体系结构。在这样的体系结构下开发企业级应用软件,软件人员可以不受时间和空间的限制,方便地进行阶段性的软件开发,即构造组件,然后在互联网上通过组件的组合和重用快速地实现各种不同的企业事务逻辑需求。使得编程有如垒积木一样快意,程序员终于从繁重的体力劳动中解脱出来,变成愉快的艺术大师。基于组件化的程序设计是继面向对象理论之后的又一大进步,这种编程思想具有很大的现实意义和发展前景。

本文主要从组件体系结构和组件模型理论上阐明 Java 组件在软件开发体系结构下实现的基本原理。然后在服务器端编写实现加解密功能的 Java 组件,实现在客户端上自由调用服务器端的 Java 组件进行文件加解密的操作的 1 个实例,以此来体现组件的重用性和用户使用组件功能的随意性。

1 Java 组件应用的基本原理

1.1 J2EE 平台和组件模型

1.1.1 J2EE 平台

J2EE(Java 2 Platform Enterprise Edition)提供了 1 个企业级的计算模型和运行环境[1],其用于开发和部署多层体系结构的应用模型[2],该模型如图 1 所示。

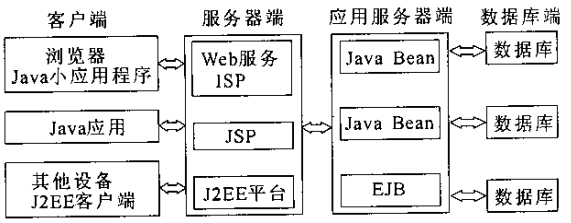

图 1 多层分布式应用模型

J2EE 树立了 1 个广泛而通用的标准,大大简化应用软件的开发和移植过程。由于 J2EE 产品可以满足当前不断变化、日趋复杂的商业需求,所以很快就成为企业构建新系统的首选产品。本文研究的是 JavaBeans 组件体系、编写的是 Java Bean 组件,因此选择 J2EE 平台。

J2EE 平台在互联网上的应用流程如图 2 所示。

2003-11-17 收稿。

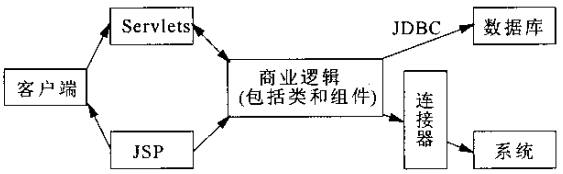

图 2 J2EE-Web 应用

1.1.2 组件模型

采用组件技术可以为企业级软件开发带来很多方便。首先,为建立在面向对象软件和开发概念之上的组件提供了一个更高级的开发模型,并且能加速企业应用程序的建立。此外,还会促使商业组件的生产商集中提高组件的质量和提供具备良好的文档接口,这样就可以加强系统质量并且还不要求内部人员具备特定底层实现技术的知识和实现服务的相关知识。

于是,可以方便快捷地使用组件模型,在组件模型的基础上构建各种企业应用。组件模型是源代码的实际单元,也就是组成应用程序的可执行单元。一般组件模型如图 3 所示,它由应用程序代码、组件和容器组成。

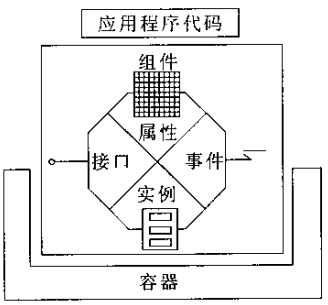

图 3 一般组件模型

如图 4 所示,各种类将被分配到组件中,以提供可重复使用的应用程序结构部件。这些组件将为即插即用的应用程序结构奠定基础。

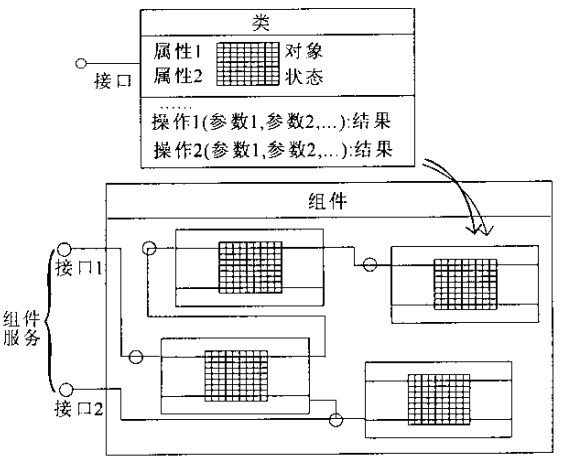

图 4 类和组件关系图

1.2 JavaBeans 组件体系和 Java Bean 组件

JavaBeans 提供一个与平台无关的组件体系结构,包括 Java Bean 和企业 JavaBean 。前者说明开发工具中应用程序组装的问题,而后者则侧重于部署组件的服务框架的细节。两者的区别是当使用企业 JavaBean 时框架是现成的,只需遵守它的 APIs 即可,而使用 Java Bean 创建服务器应用时,还得设计整个的服务框架。在本文的研究中选用的是 Java Bean 组件模型。

1.2.1 JavaBeans 组件体系

JavaBeans 组件体系是用来编写高度可重用软件单元的规则的集合,将软件单元在“即插即用”的模式下连接到一起,构造出新的应用程序[3]。JavaBeans 体系结构如图 5 所示,它一般由 Java 组件容器、组件内省信息、组件方法、组件事件、接口等组成。编写 JavaBeans 的对象规范意味着必须编写极少的定制代码,并把对象连接到一起。

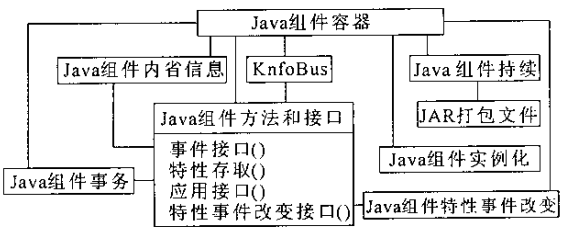

图 5 JavaBeans 体系结构

1.2.2 Java Bean 组件的重用性和扩展性

Java Bean 是一个可在编程工具中被可视化处理的可复用的软件组织[4]。

Java Bean 的应用范围是很广泛的,传统的应用是在可视化的领域,如 AWT 下的应用。但是自从 JSP 诞生后,Java Bean 更多的应用在了非可视化领域。它可以实现整个应用程序的任何一部分,包括完成计算、存储和通信等不可见的部分。

此外,Java Bean 还可以很好地工作在分布式的万维网环境。设计分布式系统的 1 个关键部分就是设计好本地和远程进程的分离。Java Bean 可以通过 JDBC (Java Database Connectivity,Java 数据库连接)与数据库相连,进行数据库查询等操作;还可以通过 IIOP 与 CORBA 相连,从而解决异构网络、不同系统间的集成等困难;Java Bean 还可以通过 RMI 与 Java 服务器相连,方便地调用部署在 Java 服务器中的任何组件以实现不同的事务逻辑功能。Java Bean 的远程应用如图 6 所示。

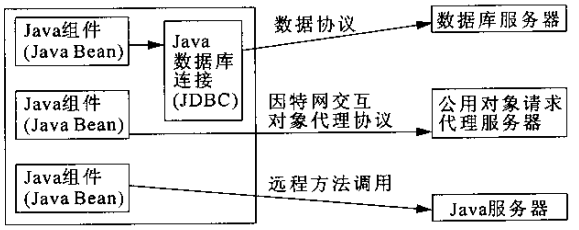

图 6 Java Bean 的远程应用

1.2.3 JavaBeans 和 Java Bean 的区别

JavaBeans 是指某种组件体系，而 Java Bean 则指使用这种体系的组件。也就是说 JavaBeans 是定义了一套规则的集合，而 Java Bean 是指按照这些规则运行的通常的 Java 对象，是遵守 JavaBeans API 和设计模式的 Java 对象，这些对象可以在可视的应用程序下进行设计和操作[3]。

2 Java 组件应用的实现方法

2.1 Java Bean 的设计步骤

一般设计 Java Bean 都遵循以下步骤[9]：

(1)指定 Bean 的属性；

(2)指定 Bean 所产生或响应的事件；

(3)定义 Bean 应公开给其它 Bean 或容器的属性、方法及事件；

(4)确定其余的复杂问题，如 Bean 是否有其自己的定制对话框，或者是否需要一些典型的信息。

2.2 定制 Java Bean 的关键技术

2.2.1 Java Bean 容器

Java Bean 容器是 Java Bean 组件的操作环境。在创建 Java Bean 时，往往不需要创建 Java Bean 容器，因为 IDE 或另一个 Java Bean 容器能“插入”标准的 Java Bean 组件。

2.2.2 InfoBus

它为 Java Bean 组件提供了在它们之间传递信息的标准机制。

2.2.3 Java Bean 组件规则

Java Bean 组件定制的最基本规则是：把 Java Bean 定义成公有类，具有缺省的公共无参数构造函数，并实现 Java.Io.Serializable 接口。如果 Java Bean 组件是可视化的 bean，那就应该扩展 javax.swing.Jcomponent 类或是扩展 javax.awt.Component 类。

2.2.4 Java Bean 属性

Java Bean 属性必须有公共的 getter 和 setter 方法，定义应按照特定的标准模式进行，这样 Java Bean 容器环境才能识别 Java Bean 属性。

此外，所有的访问方法都应该声明为 synchronized。因为一般我们假设 Bean 总是运行在一个多线程环境下的。使用同步的(synchronized)的方法可以避免形成竞争的局面。

例如，我们在定制加密组件的时候需要给出加密的类型，因此我们可以令加密类型作为一个单独的属性(jiamitype)。

则设置加密类型的方法为：

```
public synchronized void setjiamitype(){
jiamitype="DESede";
}
```

读取返回加密类型的方法为：

```
public synchronized String getjiamitype(){
    return jiamitype;
}
```

2.3 运行环境的配置

(1)操作系统：选用 Microsoft Windows 2000 高级服务器版本。

(2)Java2 平台：选用 JDK1.4.1。运行 Java Bean 最小的需求是 JDK1.1 或者以上的版本。

(3)应用服务端软件：选用 Apache Tomcat 4.0。

3 Java Bean 组件应用的实例

首先在服务器端编写实现加密和解密功能的 Java 组件，并部署在应用服务器端，然后在 Web 端编写能自由调用服务器端的 Java 组件进行文件加解密操作的 html 文件和 JSP 文件。在客户端调用服务器端的 Java 组件实现对本地文件的加解密，从而实现组件的重用性和用户使用组件功能的随意性[4~6]。

3.1 服务端应用服务器的启动

首先，在服务器端启动应用服务器。

3.2 客户端调用服务端的 Java 组件实现本地文件加密

(1)打开 IE 浏览器，在地址栏输入服务器端的 html 文件的地址，然后在加密提交页面上输入本地要加密的文件名字和加密后保存的文件名字(如图 7)。

要加密文件是本地硬盘下的 yuanwen.txt 文件，其显示如图 8 所示。

(2)按加密提交页面的“提交”按钮后自动调用 JSP 文件，从而调用服务端的 Java 组件实现本地文件加密，生成本地硬盘下的 miwen.txt 文件。密文文件的显示如图 9 所示。

图 7 加密提交页面

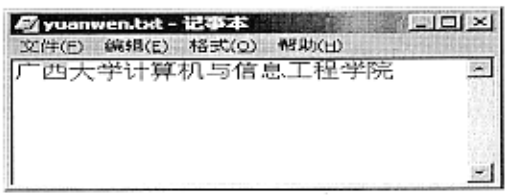

图 8 原文件显示

图 9 密文文件显示

在定制加密程序时，默认将加密生成的密钥文件 key. key 存储在服务器端，以方便对密钥的管理。因为密文和密钥的生成是由随机数做参数的，因此每次运行加密的结果都会不一样。任取一对为例子。例如，生成的密文文件如图 9 所示，那么生成的密钥文件如图 10 所示。

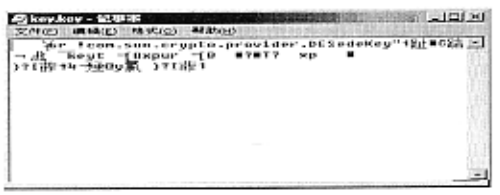

图 10 密钥文件显示

3.3 客户端调用 Java 组件实现本地文件解密

(1)打开 IE 浏览器，在地址栏输入服务器端的 html 文件的地址，然后输入本地要解密的文件名字和解密后保存的文件名字，其解密提交如图 11 所示。

图 11 解密提交页面

(2)按解密提交页面的“提交”按钮后，自动调用 JSP 文件，其显示如图 12 所示。从而调用服务端的 Java 组件实现对本地文件解密，生成本地硬盘下的 newyuanwen. txt 文件。

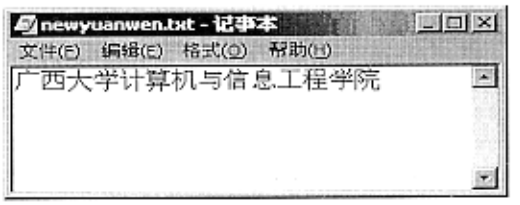

图 12 还原文件显示

4 结束语

Java Bean 的应用范围是很广的，装配和重用它的技术也很多。从社会效益和前景来看，对企业级的程序设计而言，采用组件技术就能够协调不同的设计模式和实现策略，可以根据企业计算的需求重新进行裁剪和拼装，使其能迅速反应市场的变化和技术的发展趋势。

参考文献：

1 Paul J，Perrone，et al. J2EE 构建企业系统-专家级解决方案. 北京：清华大学出版社，2001. 6.

2 飞思科技产品研发中心编著. EJB 应用开发详解. 北京：电子工业出版社，2002. 1.

3 Patrick Niemeyer，Jonatban Knudsen. JavaTM 语言入门. 北京：中国电力出版社，2001.

4 Graham Hamilton. Sun Microsystems-JavaBeans，1997.

5 Jess Garms，Deniel Somerfield. Java 安全性编程指南. 北京：电子工业出版社，2002. 1.

6 Http://java. sun. com. 2003-07-10.

（责任编辑：黎贞崇）