文章编号:1001 - 9081(2006)04 - 0853 - 04

# 一种业务逻辑可重构的三层应用服务器设计与实现

张慧翔,张新家
(西北工业大学 自动化学院,陕西 西安 710072)
(zhanghuixiang@gmail.com)

**摘 要**:将网络应用系统的逻辑抽象为表单流、事件流和数据流,提出了一种业务逻辑可重构的三层应用服务器框架,由此开发了具有可重构能力的企业信息化综合业务处理系统(ICETIP)应用服务器。ICETIP 应用服务器实现了客户连接池和数据库连接池的协调工作,给出了一种客户端和服务器之间通信机制。ICETIP 系统的主要特点是提出了一种全新的网络应用开发与维护模式,在不需要编程的条件下可方便地依照应用逻辑构造不同行业需求的网络应用系统。

**关键词**:业务逻辑可重构;应用服务器;连接池;通信机制
**中图分类号**: TP311.5 **文献标识码**:A

# Design and implementation of three-tiered application server with business logic reconstruction

ZHANG Hui-xiang, ZHANG Xin-jia
(*College of Automation, Northwestern Polytechnical University, Xi'an Shaanxi* 710072, *China*)

**Abstract:** Networking applications can be described as a set of form flows, event flows and data flows. Based on these three kinds of flows, a business logic-reconfigured framework of three-tiered application servers was proposed, and an ICETIP (Information-Centered Enterprise Transaction-Integrated Processing System) application server was developed, which is a transaction processing system with the ability of reconstructing and reconfiguring networking applications. Then the structure and implementation of the ICETIP application server were presented, including a client connecting pool and a database connecting pool and communicating mechanisms between clients and the application server. ICETIP is a new mode of developing and deploying networking applications, which can be used to build different kinds of applications without any further programming.

**Key words:** business logic reconstruction; application server; pool; communication mechanism

## 0 引言

企业应用处理系统的发展趋势是在新的模块化概念下变得越来越规模可变,且做到可配置、可重构。因此,实现企业应用处理系统的柔性化和可重构能力一直是理论界研究的热点[1]。为了达到这一目标,当前的主要方法是基于构件的软件开发。这种方法强调使用可复用的软件构件来设计和构造软件系统[2],其思想是创建可重用的构件并将其组合,用多个业务构件动态地组成一个新的应用系统,其核心是构建即插即用型的业务构件[3]。这种方法在一定程度上实现了应用系统的可重构。但是不同应用领域的软件系统,其构架与构件具有不同的特征,很难用统一的模式加以描述或定义,无法实现跨领域、跨行业的构件的重用。

基于上述目标,提出了一种软件逻辑可重构和可配置的方法,对此进行了可行性研究和设计,并开发出企业信息化综合业务处理系统(Information-Centered Enterprise Transaction-Integrated Processing System, ICETIP)。它突破了传统的实现应用系统可重构的方式,以业务逻辑可配置为主导思想,设计最核心的可重组单元,实现信息化软件系统的可配置、可重构。由 ICETIP 平台来部署企业应用系统,不仅周期快,系统稳定,而且能根据业务逻辑变化随时重构应用系统,达到了企业应用跨行业的可重构和可配置要求。

## 1 ICETIP 系统设计思想

在企业应用系统中,把所有的业务逻辑抽象成表单流、数据流和事件流。表单流是业务逻辑处理的执行界面,一系列执行界面的切换就构成了业务流程;事件流是用户在某个执行界面中操作而触发的一系列业务数据的传递和处理;数据流是伴随事件触发产生的业务数据的存储和更新工作。

应用系统的可重构、可配置体现在两个方面:1)业务逻辑处理的执行界面可配置,用户执行界面的产生是可视化的配置过程,主要是界面控件的布局;2)业务逻辑可配置,这里提出了活动事务的概念。执行界面中触发的事件流就是一个个活动事务的集合。比如,用户点击了执行界面上的一个按钮,触发单击事件,从而执行了一系列的操作,就把这些操作的整体归为一个活动事务。一个活动事务又由一个个具体的动作组成。如上面所说的点击按钮,可能是管理员要删除某个员工的记录信息,这其中就包括了两个具体的动作,首先是删除数据库里有关该员工的相关信息,其次是把处理后的结果重新刷新并表现在界面上。把这两个具体的操作步骤叫做

**收稿日期**:2005 - 10 - 10;**修订日期**:2005 - 12 - 22

**作者简介**:张慧翔(1981 - ),男,湖南益阳人,硕士研究生,主要研究方向:网络信息安全、软件可重构; 张新家(1961 - ),男,湖北洪湖人,副教授,博士,主要研究方向:网络信息系统、软件可重构.

动作。数据的改变融合在活动事务的处理当中。图1是业务逻辑和活动事务关系图。动作的组合构成活动事务,活动事务的组合形成执行界面上不同的业务处理逻辑。通过活动的不同组合来适应企业业务逻辑的改变。

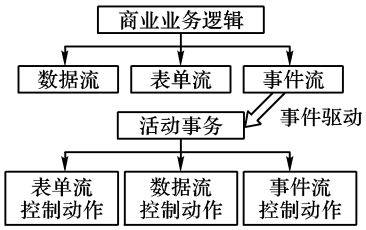


图1 业务逻辑和活动事务关系

活动事务的执行是由事件来触发的,基本的事件类型有:单击(Click),双击(DoubleClick),选择改变(SelectChanged),选择完成(SelectOK),初始装载(OnLoad),窗口关闭(OnClose)等。这些事件在执行界面中由用户操作产生。

针对抽象出来的表单流、数据流和事件流,定义了表单流处理动作、数据流处理动作和事件流处理动作。表单流处理动作主要处理表单之间的切换;数据流处理动作主要处理业务数据与业务数据库的交互;事件流处理动作主要是业务流程执行界面中控件和数据变量的控制。下面是一些基本的动作,动作的种类可以根据系统需要增加。

1) 表单流

OPEN:打开一个业务执行界面。

CLOSE:关闭一个业务执行界面。

2) 数据流

PUT:把操作数据写入后台数据库。

SET:把后台数据库数据导入到执行界面。

UPDATE:更新后台数据库相关数据。

DELETE:删除后台数据库相关数据。

TESTDATA:测试后台数据库中数据的唯一性。

3) 事件流

MOVE:执行界面中控件数据相互传递。

DELETE:清除执行界面中控件显示的数据。

REFRESH:根据后台数据库来更新执行界面中控件显示的数据。

NEWVAR:生成新的数据变量,类型包括字符型、整型、浮点型和逻辑型等。

用户点击按钮触发Click事件,执行管理员要删除某个员工的记录信息的活动事务,包括一个数据流动作DELETE,删除数据库该员工信息;然后是一个事件流动作REFRESH,重新刷新控件显示数据。

ICETIP系统实现应用的可重构、可配置,其核心思想是基于企业应用系统所包含的业务逻辑都能被抽象成数据流、表单流和事件流,据此设计并实现了可重构应用核。可重构应用核是一个可重构应用配置管理系统,负责业务流程到数据流、表单流和事件流的抽象。该系统产生的流配置信息被存储到后台数据库中,由客户端应用自动处理系统来读取并重现整个业务流程。客户端应用自动处理系统实现了客户端应用随配置信息变化而产生不同的业务逻辑处理流程。该系统通过应用服务器来获取流配置信息,并在应用服务器中完成业务逻辑数据的处理。后台数据库存储了企业应用抽象出来的流配置信息和业务逻辑处理产生的数据。客户端应用自动处理系统与后台数据库的交互通过应用服务器完成。

## 2 ICETIP系统结构

ICETIP系统基于网络应用的三层架构应用,它的系统构架设计及实现如图2所示。

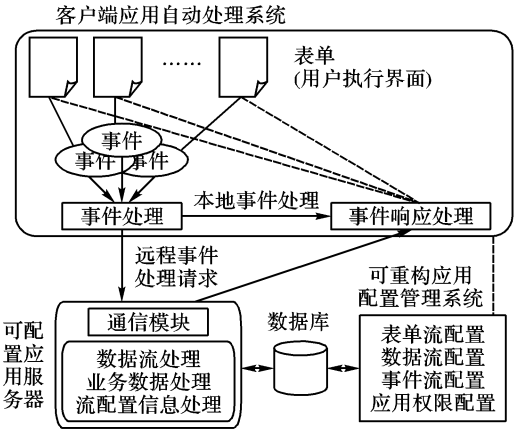


图2 系统构架设计及其实现

1)可重构应用配置管理系统:该系统根据不同的业务逻辑需求,以可视化配置来生成用以描述具体企业应用系统业务逻辑的表单流、数据流和事件流,并将这些信息以特定的组织形式存入后台数据库。在ICETIP开发平台中,引入了基于角色的用户访问控制配置管理,将访问权限与角色相联系[4]。对业务逻辑参与用户按角色分类,每个角色关联对应的流配置信息。

2)客户端应用自动处理系统:系统接受用户登录,并据此向应用服务器提交请求信息,通过应用服务器返回的与用户角色相关的表单流自动生成适合于各种不同业务需求的用户操作界面;响应用户操作触发的事件,完成本地事件处理,提交数据库处理请求给应用服务器。应用服务器处理完相关操作,把结果返回客户端。客户端向用户显示具体的执行结果。

3)可配置应用服务器:该模块是整个ICETIP系统运行的重要支柱。应用服务器处理远程客户端的请求,主要解决业务逻辑处理,客户端连接管理,客户端资源访问控制,数据库连接管理等。

## 3 ICETIP系统设计与实现

使用ICETIP系统在开发具体的企业应用,分为两个阶段。第一个阶段,搜集企业业务流程,然后使用可重构应用配置管理系统抽象到数据流、表单流和事件流,产生流配置信息,储存到后台数据库中。同时搜集参与业务逻辑的用户权限信息,分为多个角色,每个角色关联一定的流配置信息。第二阶段,系统运行。应用服务器和客户端应用自动处理系统配合维持整个企业业务逻辑的运行。运行阶段产生的需求变更可以在可重构应用配置管理系统的参与下及时更改。

客户端应用自动处理系统解释和接受用户的应用请求逻辑,将请求发给后台应用服务器进行处理,并接受和处理后台应用服务器返回的结果;后台应用服务器主要负责接收客户端应用自动处理系统的事件处理请求,完成请求消息的解释和执行,并对后台数据库进行连接管理。应用服务器可以分为通信模块,业务处理单元和数据库操作模块三个基本部分来支持ICETIP系统。

### 3.1 客户端应用自动处理系统

客户端系统对应用逻辑是自适应的。客户端的处理依赖于可重构应用配置管理系统在企业应用开发阶段产生的配置信息。当企业应用业务逻辑更改时,可重构应用配置管理系

统完成相应流配置信息的更新,对客户端系统不产生影响。

用户登录系统,提交用户名和密码到应用服务器。应用服务器验证用户信息,从后台数据库根据用户权限读取相应的流配置信息,并传输给客户端。客户端应用自动处理系统根据流配置信息中的用户执行界面信息生成用户初始界面,等待用户进一步操作。

用户操作控件,触发事件,系统在流配置信息中根据事件类型读取活动事务,按照流配置的顺序依次把活动事务中每个活动装载到事件处理器中处理。定义好的事件都可以在事件处理器中处理。一个事件处理器可以根据不同的输入,完成不同的动作。在事件处理器中产生的需要访问后台数据库的请求,由客户端封装成数据请求发送到应用服务器端,由服务器端完成进一步的处理。客户端接收服务器的返回结果,显示到执行界面,等待用户进一步处理。

客户端应用自动处理系统自身在获得流配置信息后可以完成表单流和与数据库无关的事件流的处理,对数据流的处理依赖于应用服务器。客户端对应用服务器发送两种请求:1)流配置信息请求,即用户登录时,客户端向服务器请求其角色相关的流配置信息;2)数据流处理请求,这类处理无法在客户端独自完成,需要后台数据库的数据。

### 3.2　通信模块

通信模块作为应用服务器和客户端自适应程序的信息交换中介,两者之间定义了通信协议。通信模块负责协议的解析和用户连接管理,维护了一个消息处理池来提高执行效率[5]。

协议格式定义如下:

| Type | MssgLength | Command/Code | Identifier |
|---|---|---|---|
| Trans | Data | | |

各字段定义如下:

Type:1Byte,规定消息的类型,1 为请求消息,2 为响应消息,3 为错误或状态报告。

MssgLength:2Byte,消息的长度。

Command:1Byte,特定命令编码,如流配置信息的请求代码为 1000。

Code:1Byte,特定命令的响应消息编码,如流配置信息的响应代码为 1001。

Identifier:2Byte,用于请求消息和响应消息的匹配。

Trans:1Byte,标识活动事务的起始,0 为开始,1 为结束,2 为中间活动。

Data:与特定命令相关的数据,具体格式可以根据相应命令进行设计。对于数据较大的消息可以将长消息分段传输。对流配置信息的请求消息,该部分包含用户名和密码(UID = xxx; PWD = xxxxxx);对数据流处理请求消息,该部分包含了要执行的 SQL 语句。

假设管理员要删除员工 ID 号为 1024 的记录信息,该活动事务中的两个活动 DELETE 和 REFRESH 在客户端是无法独立处理的,客户端应用自动处理系统将向服务器发送两个请求数据协议包,假设数据库中员工表名为 Empl。格式如下:

| 1 | 41 | 4 | 1001 | 0 |
|---|---|---|---|---|
| Sql = Delete from Empl where Empl_ID = 1024 | | | | |

| 1 | 24 | 6 | 1002 | 1 |
|---|---|---|---|---|
| Sql = Select * from Empl | | | | |

通信模块接到客户端的连接请求,由连接分配器将该客户连接分配到消息处理池中空闲的消息处理器。消息处理器接收请求数据协议包,分析请求数据协议包头部封装信息,形成一个活动对象。每个消息处理器包含一个业务处理单元,活动对象排队等待业务处理单元处理,处理结果由消息处理器封装为响应数据协议包返回给客户端。通信模块的处理流程如图 3 所示。

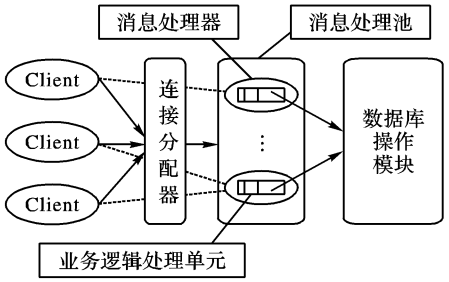


图 3　通信模块处理流程

消息处理池内消息处理器数量可以根据用户连接请求数来增减。每个消息处理器维护了一个用户请求队列,可以服务多个用户请求,且管理的最大用户请求数量可配置。

### 3.3　业务处理单元

消息处理器生成命令对象交给业务处理单元处理。活动对象仅包含业务处理单元处理所需的必要信息:

```
{ int ComType;                                    //命令类型
  string data;                                    //命令数据
  long id;                                        //对象 ID
  longTransID;                                    //活动事务 ID
  struct Resultresult;                            //处理结果
  intTransStatus;                                 //活动起始标志;
                          //0: 起始活动; 1: 结束活动; 2: 中间活动
  intstatus;    //命令处理状态; 0: 未处理; 1: 处理成功; -1: 错误
}
```

消息处理器会根据请求数据协议包中活动事务的起始标志分配一个新的活动事务 ID,同一个活动事务中的活动对象具有相同的活动事务 ID。业务处理单元对每一个新的活动事务请求一个新的数据库连接对象,即一个数据库连接对象服务于一个完整的活动事务。业务处理单元维护了一个当前服务活动事务的数组,记录了当前服务活动事务的事务 ID。

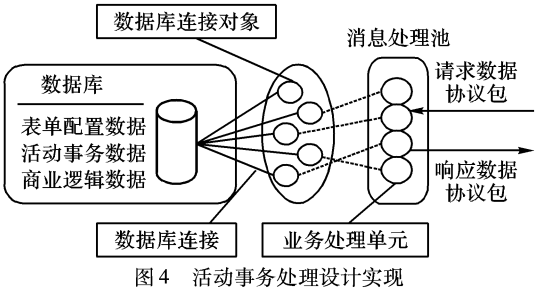


图 4　活动事务处理设计实现

业务处理单元根据命令类型形成不同的数据库请求,提交相应数据库连接对象执行。命令类型为数据流请求时,data 变量是要执行的 SQL 语句。命令类型为流配置请求时,该字段保存的是用户名和密码,此时需要将 data 字段重新赋值生成获取流配置信息的 SQL 语句。处理结果保存在命令对象中,由业务处理单元所在消息处理器打包成响应消息发给对应的客户端。图 4 描述了活动事务的处理过程。

业务处理单元根据活动事务 ID 来保证活动事务的完整性。活动对象处理过程中出现数据库操作语句错误或者客户端发送了活动事务执行错误消息时,数据库需要进行回滚。

### 3.4 数据库操作模块

数据库操作模块采用数据库连接对象池技术[6],对象池中每一个对象代表一个数据库连接。业务处理单元向数据库操作模块申请数据库连接对象。对象池具有伸缩能力,可以配置池的规模。每个数据库连接对象可为所有用户服务,但用户获得一个数据库连接对象必须按一个完整的事务进行调度。

用户从连接池中获得一个数据库连接并使用这个连接,而不需要为每一个连接请求重新建立一个连接。一旦一个新的连接被创建并且放置在连接池中,用户就可以重复使用这个连接而不必实施整个数据库连接创建过程。当用户使用完连接后,该连接被归还给连接池而不是直接释放。数据库连接池中可能存在着多个没有被使用的连接一直连接着数据库。

数据库连接池都与一个独立的连接字符串及其事务上下文关联。每次打开一个新的连接,数据提供者会尝试将指定的连接字符串与连接池的字符串进行匹配。如果匹配失败,数据提供者创建一个新的连接并将它加入连接池。连接池被创建之后,除非进程结束,否则不会被拆除。

数据库连接池创建之后,系统会创建一些连接对象并将它们加入连接池,直至达到额定的最小连接对象数量。以后,系统会根据需要新建和加入连接对象,一直到达最大连接对象数量限额为止。如果程序请求一个连接对象时没有空闲的连接对象可用,且连接池里面的对象数量已达到上限,则请求被放入队列,一旦有连接被释放回缓冲池就立即取出使用。

## 4 系统运行测试

对开发出来的 ICETIP 系统进行了测试。测试数据库分别用了Oracle8.17和SQLServer7.0,由此配置出来的客户端系统能顺利运行,而且能根据配置端的修改配置,重新以新的功能执行。在不改动任何程序代码的情况下,系统达到了可重构的目标。应用服务器提供的消息处理池、数据库连接池运行稳定。

由 ICETIP 系统开发出来的某信息化处理系统已投入运行。该系统在整个开发过程中,没有再进行任何的程序开发。从当前运行情况来看,系统功能稳定。

## 5 结语

把企业应用系统所包含的业务逻辑抽象成数据流、表单流和事件流,通过活动事务的不同组合实现应用的可重构、可配置,据此开发出来的 ICETIP 系统平台很好地解决了同一行业内和不同行业间网络应用系统的业务逻辑可重构,同时又能够实现快速部署,能够根据业务逻辑变化随时重构应用系统。

**参考文献:**


[1] 韦东方,薛恒新,游专. 任务/目标驱动的 ERP 系统构建的分析与应用[J]. 计算机应用,2004, 24(11):56-59.

[2] 张屿,李彤. 一个基于构件的软件过程控制模型[J]. 计算机应用研究,2005, 22(3):74-75.

[3] 石双元,吴新明,刘琦. 构件化信息系统体系结构及其业务构件模型研究[J]. 计算机工程与科学,2005, 27(5).

[4] 汪厚祥,李卉. 基于角色的访问控制研究[J]. 计算机应用研究,2005, 22(4):125-127.

[5] 段雪峰,张新家,戴冠中. 中间件服务性能建模与分析[J]. 计算机应用,2004, 24(1):105-106.

[6] HALL M. Core Servlets and JavaServer Pages[M/CD]. USA: Prentice Hall,2001.


(上接第 819 页)

综上所述,我们可以获得以下结论:MMAC 协议能够有效地实现对多源组播协议的安全访问控制。

### 2.2 安全性分析

下面将依次对协议中的 4 个消息进行分析。

(1) 在消息 1 中,消息使用共享密钥 $K_{(gm,TR)}$ 加密传输,保证了该消息的机密性。同时,当 *TR* 中的路由器收到此消息后,能够确认其来自于 *gm*。而 *M-timer* 除了实现 *TR* 中路由器与 *gm* 的时间同步外,还有效地保证了 *SC'* 的新鲜性。

(2) 消息 2 实现了 *gm* 向合法主机分发授权证书。消息使用合法主机的公钥加密,保证了证书只能被此主机获得。同时证书使用 $K_{(gm,TR)}$ 加密,即保证了证书内容不被主机所知,又实现了 *gm* 对证书的签名。

(3) 当主机要参与 $c_i$ 组通信时,向路由器 *r* 发送消息3。此消息使用 *PK*(*TR*) 加密,保证了只有可信路由器 *r* 可收到此消息。而路由器 *r* 通过对授权证书的验证来判断证书是否有效,确认此主机是否是合法主机及是否有权参与 $c_i$ 组通信。

(4) 因为只有 *gm* 和 *TR* 中的可信路由器才会拥有共享密钥 $K_{(gm,TR)}$,所以上游路由器只需要对 *MAC*(*JOIN/REQS/LEAVE*, $K_{(gm,TR)}$) 进行验证,就可实现对不可信路由器的访问控制。

可见,MMAC 协议实现了对多源 IP 组播的安全访问控制。

**参考文献:**


[1] DIOT C, LEVINE B, LYLES B, *et al*. Deployment Issues for the IP Multicast Service and Architecture[J]. IEEE Network, 2000, 14(1): 10-20.

[2] PUNEET S, EDWARD P, RADHIKA M. IP Multicast Operational Network Management: Design, Challenges, and Experiences[J]. IEEE Network, 2003, 17(2):49-55.

[3] JUDGE P, AMMAR M. Security Issues and Solutions in Multicast Content Distribution: A Survey[J]. IEEE Network, 2003, 17(1):2-8.

[4] BALLARDIE A, CROWCROFT J. Multicast-Specific Security Threats and Counter-Measures[A]. The ISOC Symposium on Network and Distributed System Security[C]. San Diego, 1995. 2-16.

[5] HARDJONO T, CAIN B. Key Establishment for IGMP Authentication in IP Multicast[A]. IEEE European Conference on Universal Multiservice Networks(ECUMN)[C]. Colmar, France, 2000.

[6] HE H, HARDJONO T, CAIN B. Simple Multicast Receiver Access Control[Z]. Draft- irtf-gsec-smrac-00. txt, 2001.

[7] HE H, CAIN B, HARDJONO T. Upload Authentication Information Using IGMPv3[Z]. Draft-he-magma- igmpv3-auth -00. txt, 2001.

[8] JUDGE PQ, AMMAR MH. Gothic: Group Access Control Architecture for Secure Multicast and Anycast[A]. IEEE INFOCOM[C]. 2002.

[9] HAYASHI T, ANDOU D, HE H, *et al*. Internet Group Membership Authentication Protocol(IGAP)[Z]. Draft-igap-03. txt, 2003.

[10] BRYNJúLFSSON B, HELGASON ÓR, HJÁLMTýSSON G. Self-configuring Lightweight Internet Multicast[A]. IEEE SMC 2004[C]. Hague, Netherlands, 2004.