• 信息化技术 •

基于智能客户端的企业应用开发框架

吴寒思， 邵 栋， 荣国平
(南京大学 软件学院，江苏 南京 210093)

摘 要：为了解决当前中小软件企业开发混乱的问题，帮助开发人员实现软件架构级别的重用，设计了一个基于智能客户端的企业应用开发框架，通过辅助工具支持代码一键生成和快速开发，并封装一系列的设计模式和业务逻辑，特别是面向特定领域业务逻辑的基础服务供开发人员复用，同时也提供了对第三方组件的支持。基于本框架开发管理信息系统可以简化开发流程，缩短开发周期，提高软件复用率和软件质量。

关键词：软件复用；应用开发框架；快速开发；智能客户端；.NET 平台
中图法分类号：TP311.52 **文献标识码**：A **文章编号**：1000-7024 (2009) 11-2707-04

Enterprise application development framework based on smart client

WU Han-si, SHAO Dong, RONG Guo-ping
(Software Institute, Nanjing University, Nanjing 210093, China)

Abstract: In order to tackle the chaos in current small ISV development and help the developer achieve architectural reuse, an enterprise application development framework based on smart client is designed, which supports one-touch code generation and rapid development with auxiliary tool, encapsulating a series of design patterns and business logic, especially domain-specific business logic and infrastructure for developers, along with support for third party component. Developing based on this framework will benefit from simpler development process, shorter development cycle, higher software reuse rate and higher quality.

Key words: software reusing; application development framework; rapid development; smart client; .NET platform

0 引 言

软件复用和软件构件化已是当前软件开发的大势所趋。而在目前国内中小软件企业里，由于开发人员技术水平较低，管理人员软件工程素养欠缺，导致开发过程混乱，软件复用率低，开发出的产品质量差，成本高。面对这种缺乏严格软件过程控制、需求变化频繁、低成本要求、工期紧张的开发环境，引入企业应用开发框架成为中小软件企业软件开发一种有效的解决方案。

框架是一个试图实例化说明的部分完整的软件(子)系统[1]。企业应用开发框架在以往应用开发的基础上，抽象出一系列的基础服务、业务逻辑，并组织成结构化的组件，把复用提升到了系统架构级别[2]。区别于一些基础开发框架，如开源ORM工具NHibernate，微软的企业库，应用开发框架预先设定了总体架构，通过模板化、定制开发等机制，使开发人员的工作从使用基础框架的服务变成填充、扩展框架(应用开发框架也提供、集成了大量基础服务)，体现了好莱坞原则。由于使用了大量的设计模式，开发人员有意识无意识地重用了许多优秀的软件设计方法。同时通过复用面向特定领域的业务逻辑，缩短了开发人员学习业务知识的时间。这大大提高了开发效率和开发质量，使快速开发变得可行。

本文设计的EAF(enterprise application framework)是基于以上的思想设计实现的一个基于智能客户端的企业应用开发框架。智能客户端(Smart Client)[3]是微软公司于2004年推出的一种基于.NET Framework的应用程序。该技术结合了胖客户端的强大功能和瘦客户端的高度可部署管理性的特点。在吸取了传统C/S架构的应用程序的优点如用户界面灵活、丰富，可用性好，支持离线工作，充分利用客户端的计算资源，节省网络带宽的同时，也具有B/S架构应用程序的易于部署、管理的特点，实现了无接触部署和自动更新。

1 EAF架构

EAF架构采用了服务层模式[4]，通过后台向前台暴露统一的接口提供服务。总体架构图如图1所示。框架分为前台和后台：其中前台运行在客户机，负责数据的展示、输入数据的简单验证、命令的包装；后台运行在服务器上，又可以细分为服务提供层、业务外观层、业务规则层、数据存储层和领域对象层。合理的权责划分保证了各个组件之间的松耦合。前台和后台之间使用Web Service进行通信。后台统一处理对资源的操作，如数据库操作，短信或邮件的发送。此外，还提供了代码生成器和一些基础服务。

EAF首先关注代码生成快速开发。通过读取数据库里符合命名规范的表和字段信息，生成各个后台层相应的代码，减

收稿日期：2008-06-21；修订日期：2008-08-22。
作者简介：吴寒思(1985－)，男，福建厦门人，硕士研究生，研究方向为软件工程；邵栋(1976－)，男，安徽淮南人，硕士，副教授，研究方向为软件工程；荣国平(1977－)，男，江苏昆山人，硕士，讲师，研究方向为软件工程。E-mail：mg0732008@software.nju.edu.cn

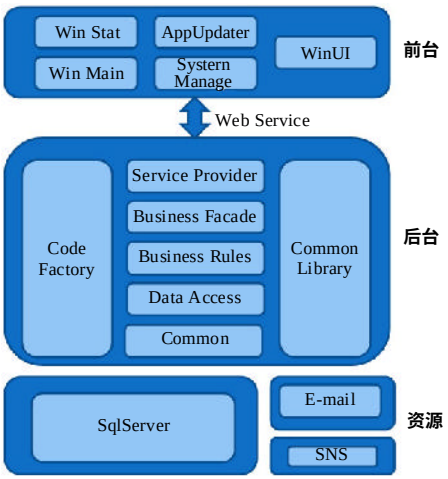

图 1 EAF 体系结构

轻了开发人员的负担，也在一定程度上降低了分层结构导致的代码冗余。EAF 也支持简单的工作单元[4]。这主要由代码生成器生成对象 CRUD 操作的 SQL 语句来实现。

和大部分框架类似，EAF 内置了基础服务和业务逻辑，包括用户—权限系统、基础数据管理、访问统计、异常处理等。这些内置组件使开发人员避免了大量的重复劳动，从而专注于业务逻辑。

前台的界面往往需求变化大，要求有良好的封装和一致大方的界面。EAF 采用了大量的第三方组件，实现了不同显示条件下(如分辨率、窗口风格)的一致显示，提高了程序的安全性和健壮性，同时美化用户界面，大大减少了开发的工作量。

通过引入以上的框架设计、工具和组件的支持，在数据库设计、代码生成、简单界面设计之后，目标系统已经完成了大量的基础工作，一个带有日志记录、访问统计等功能的原型已经可以运行。需求分析人员可以通过这个原型与客户进行交流，深入了解需求。

2 EAF 设计

2.1 层次分析

2.1.1 前台用户界面层

如图 2 所示，前台从后台提取数据进行展示，具有以下几个特点：

(1)采用智能客户端技术，部署升级通过 Internet Explorer，解决了 C/S 系统升级部署的问题。而又通过本地缓存支持离线工作。另外，由于充分利用了本地计算机资源进行数据验证和客户状态保存，降低了服务器端对计算和存储资源的消耗，取得了良好的系统性能。

(2)早期的 C/S 系统客户端直接与数据库进行连接，在安全和数据库稳定性上没有保证。EAF 框架不直接与系统资源如数据库发生交互，而是通过加密访问后台暴露的 Web Service 接口进行操作。而统一的接口使前台和后台解耦合，并有较好的扩展性。

(3)主菜单不直接绑定资源或事件，而是读取数据库里的菜单项以及语言配置，结合用户权限，实时生成菜单项。

(4) 使用了成熟的第三方组件。例如显示控件采用了 Infragistics 公司的 Net Advantage，Component One 公司的 C1Sizer 等商业组件，把开发人员从繁杂的界面设计美化工作中解放出来，也得到了更好的用户体验，报表制作和展示采用了 Data Dynamic 公司的 Active Report。

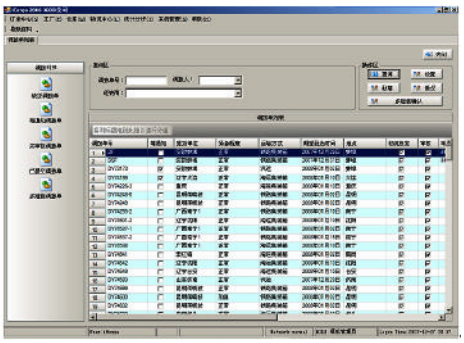

图 2 EAF 运行界面

2.1.2 服务提供层

服务提供层是服务器端对外界的接口，需要考虑互操作性、可扩展性和数据的安全性。EAF 里服务提供层封装了业务外观层提供的统一资源接口，并以 Web Service 的形式对外提供服务。前后台通讯使用 Web Service，使得数据传输可以穿过企业防火墙，并支持跨平台调用。同时，软件的客户端不仅仅可以是基于智能客户端的前台用户界面层，也可以是其它客户端如移动设备。如果客户需要把现有系统与其它系统进行集成，可以利用服务提供层提供的 Web Service 接口进行 SOA 整合。而通过使用 Web 服务增强 (web service enhancement，WSE) 对传输数据进行加密和压缩，保证了数据的安全，降低了网络流量。

2.1.3 业务外观层

业务外观层定义了后台服务的统一接口，对资源进行调度，隔离事务。和服务提供层类似，这一层对业务规则层进行了薄薄的封装。和文献[4]里描述的服务层模式不同的是，EAF 既没有使用领域外观，也不是纯粹的操作脚本，而是通过业务外观层分离了外观接口，而业务逻辑层包括了应用逻辑与领域逻辑。

2.1.4 业务规则层

业务规则层是 EAF 的核心，这一层包含了一些面向特定领域(主要是物流和外贸)和开发人员定制开发的业务逻辑，如数据计算、验证和工作流路由判定。除了调用数据访问层提供的 CRUD 方法，开发人员也可以直接编写 SQL 语句访问数据库资源，或者使用邮件或短信发送服务。

2.1.5 数据访问层

数据访问层封装了数据访问逻辑。这一层包含了大量的生成代码，结合 EAF 提供的通用方法，数据访问层为领域对象提供了 CRUD 方法，完成主要的持久化工作。在大部分情况下，开发人员只要提供领域对象，数据访问层生成对应的 SQL 语句或存储过程调用，并完成数据集的持久化。

2.1.6 领域模型层

领域模型层里包含了领域模型及用于界面显示和传递数据的值对象。领域模型由代码生成器生成。值得一提的是，除了生成领域模型的字段以外，还生成了字段名称和表名称

的字符串常量。当开发人员引用字段名称时，需要引用这些字符串常量。这个方法可以显著降低由于开发人员的拼写错误造成的程序缺陷，虽然增加了代码量，但结合IDE的智能提示功能，反而提高了效率和准确率。值对象简化了在复杂的用户界面下的开发，同时在客户端作为缓存数据的容器。

2.2 代码生成

把EAF划分成不同的层次，虽然带来了权责明确、松耦合的好处，但是也不可避免使得代码膨胀。例如，服务提供层和业务外观层并没有实际的业务逻辑，而仅仅是提供Web Service实现和隔离各个操作。对于数据访问层和领域模型层，代码生成提供了基本的持久化服务，生成领域对象和结果集，使CRUD操作自动化。而在业务规则层，代码生成器搭建了业务规则代码的基本框架，避免开发人员大量拷贝-粘贴原有代码造成的错误。

代码生成工具只依赖于数据库。通过读取数据库指定表的符合命名规范的字段，直接生成指定表对应的后台五层的代码和存储过程，开发人员只需要把生成结果导入解决方案和数据库。

2.3 基础组件

(1) 用户权限系统

用户权限系统是几乎所有企业应用系统的核心模块之一。EAF内置了基于RBAC（基于角色的访问控制）的用户权限系统，使得在项目构建启动之初，开发人员就拥有了一个具有高可定制性的用户权限模块。通过把对资源的操作合并成角色，再把角色赋予用户的方法进行访问控制。EAF封装了用户权限系统，包括数据库表、数据访问操作、接口列表和用户界面，不需要开发人员重新编写权限管理的代码。EAF管理的资源主要是菜单项，但也提供了界面控件级别的访问控制支持。

(2) 基础数据管理

在企业应用开发中，往往有大量的以读取为主要操作的基础数据，特别是一些选项数据。这些基础数据在输入、选取、统计时的行为比较固定，同时存在数据扩充的需求。EAF统一管理这些数据，并提供了绑定了数据库字段的用户控件，开发人员只要通过配置就可以直接添加绑定了基础数据字段的控件。

(3)访问限制和统计

访问统计功能通用性强，却往往是项目延期的牺牲品。EAF记录了客户端每次的访问信息，如访问时间、IP地址、用户名等，并提供了统计查询界面，满足了大部分客户对客户端登录的统计需求。另外，EAF集中管理客户端登录授权，所有访问系统的客户端需要中心统一授权之后才能登录，有效解决了客户端非法登录问题。

(4)异常处理

软件运行过程中，由于用户异常输入、运行环境变化或者软件自身的缺陷，出现异常是不可避免的。如何有效处理异常并得到缺陷反馈是EAF关注的方面之一。除了日志记录以外，EAF还提供了“错误报告”机制，把错误发生时的运行环境、出错代码行数、用户输入等信息提交远程服务器，有效收集大量错误信息，供开发和维护人员参考。同时也把系统与项目任务管理软件JIRA和项目知识共享平台Confluence进行对接，使用户的反馈在第一时间可以传达给开发人员。

2.4 开发流程

使用EAF开发流程如图3所示。

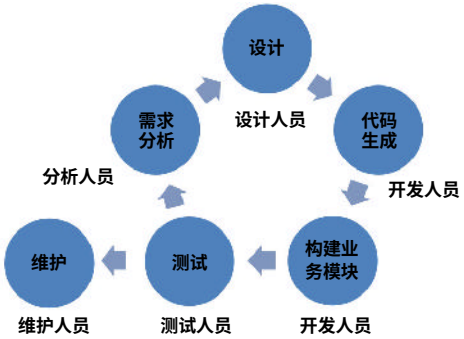

图3 使用EAF开发流程

使用EAF进行开发，推荐使用迭代式开发，把系统分成各个功能模块，每次集中于一到两个功能模块，在需求分析后设计数据库、业务外观接口，然后使用代码生成器生成相应的代码并导入项目，在相应业务代码完成后进入测试阶段，进入下一个迭代或者交付软件。

3 其它快速开发框架

3.1 AppFuse

AppFuse[5]是一个“一键启动”式的J2EE入门框架，由Matt Raible于2002年创建。AppFuse使用Java平台上的开源工具构建，帮助开发人员快速而高效地开发Web应用。AppFuse的核心在于通过一系列的启动脚本和配置，在新建项目之初就已经生成了项目的骨架，包括了大量的类、资源文件、配置文件和数据库建表语句。AppFuse无缝集成了主流的开源工具，如Spring，Hibernate，Struts，Acegi Security，XFire等，开发人员无需另外学习集成配置。AppFuse通过AppGen工具生成代码，支持从数据库生成领域模型、领域模型生成数据库建表语句及应用层代码。AppFuse还提供了很多应用程序需要的特性，如认证授权、模板化布局、Email和自动化测试[6]。

EAF与AppFuse类似，支持代码生成，集成了一些基础服务如用户系统。区别在于AppFuse是“开箱即用”框架，而EAF更注重客户端应用和业务逻辑的抽象。

3.2 SAF

SAF(simplified application framework)是Xin Chen在文献[7]里描述的一个基于.NET平台，面向B2B领域的应用框架。SAF由通用的跨领域框架组件和特定领组件框架组成，前者包括了诸如类工厂服务、授权服务、加密服务等通用组件，几乎没有特定领域知识；而后者则包含了面向B2B领域的通用业务逻辑，如文档交换服务(用多个处理层来处理文档对象)和工作流。这些组件基于.NET框架，而这些组件之上是客户化应用层。

SAF提供了丰富的类库，覆盖了大量的企业开发常用功能。但在客户端的开发还有所欠缺，开发人员需要做大量的工作设计和实现客户端。另外，SAF主要用于教育目的，而非实际生产用途。

3.3 Ruby on Rails

Ruby on Rails[8]是一个用Ruby语言编写的“全功能”Web

开发框架,严格按照 MVC 的结构开发。Ruby on Rails 的设计原则包括“不要重复自己”(Don't Repeat Yourself)和“约定胜于配置”(Convention over Configuration)[9]。Ruby on Rails 的特点在于通过独特的 MVC 架构、代码生成和大量的设计约定,很大程度上减少了开发问题和开发工作中的重复劳动。

Ruby on Rails 最近两年获得了大量应用,特别是新兴的 Web 2.0 网站,如微博客 Twitter[10]、在线项目管理 Basecamp[11]。由于 Ruby on Rails 的成功,许多程序设计语言也推出了类似 Ruby on Rails 的快速开发框架,如 PHP 的 CakePHP、Zend Framework 和 Python 的 Django。

相比 Ruby on Rails 应用于 Web 2.0 网站,EAF 定位于企业级市场,并提供了面向特定领域的逻辑抽象。基于.NET 使得 EAF 可以自由使用大量的类库资源,而 Ruby 的类库目前还处于一个增长完善期。

4 结束语

EAF 在设计和实现中体现了模块化和重用的思想。在 EAF 的基础上,已经开发出几种物流、外贸(报关)和货物代理管理软件,部署在华南和华东数家货物代理和物流公司。在这些项目中,应用 EAF 省去了很多重复性的工作。代码生成体现了快速开发和对变化的及时响应;内建的面向物流领域的业务逻辑大大缩短了开发周期,而开发一个系统积累下来的业务逻辑经过抽象、重构,又被添加到框架里,使得框架本身也完成了一次迭代过程。

在开发中 EAF 也暴露出一些问题。最突出的在于表模式使得领域逻辑冗长缺乏可读性,另外缺少启动脚本。缺乏对象生命周期的管理机制使得框架组件之间仍然存在一定的耦合。以上这些都是 EAF 在下几个迭代周期中亟待解决的问题。

参考文献:

[1] Bushchmann F,Meunier R,Rohnert H,et al.面向模式的软件体系结构,卷 1:模式系统[M].北京:机械工业出版社,2003:230.

[2] 温昱.软件架构设计[M].北京:电子工业出版社,2007:32.

[3] Hill D,Webster B,Jezierski E,et al.Smart client architecture and design guide[M].Microsoft Press, 2004:1-7.

[4] Fowler M.企业应用架构模式(影印版)[M].北京:中国电力出版社,2004:133-142,184-194.

[5] Raible M. AppFuse[OL/CP].http://appfuse.org.

[6] Raible M.Seven simple reasons to use AppFuse[OL/EB].http://www.ibm.com/developerworks/java/library/j-appfuse/index.html,2006.

[7] Chen Xin.应用框架的设计与实现——.NET 平台[M].北京:电子工业出版社,2005:39-50.

[8] Ruby on rails[OL/CP].http://www.rubyonrails.org.

[9] Wikipedia.Ruby on rails[OL/EB]. http://en.wikipedia.org/wiki/Ruby_on_Rails.

[10] Twitter[OL/CP].http://twitter.com.

[11] Basecamp[OL/CP].http://www.basecamphq.com.

(上接第 2683 页)

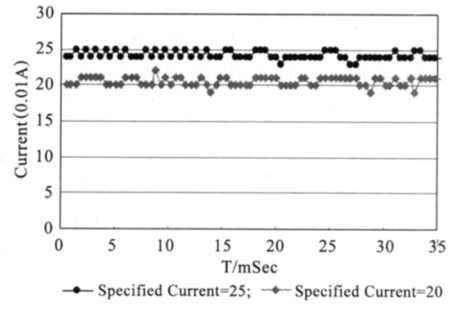

图 8 电流控制实验结果

一般情况下,当电机的转向和电流的控制方向相反时,会产生一个较大的反向电流,使得实际电流比指定的电流要大得多,并随着反向转动速度的增大而增大。在本文提出的方法中,一个控制周期对电流进行 8 次检测,每 50uSec 能对电机驱动器的 FET 进行一次操作。即使有较大的反向转动速度,也能实现实际电流和指定电流的基本一致。

6 结束语

本文提出的基于 PIC 单片机的直流电机伺服控制器有位置控制、速度控制和电流控制 3 种控制模式,并且 3 种控制模式可在任意时刻自由选择和切换。在电流控制算法中,提出了电机的转动方向和电流的控制方向相反情况下的控制方法。实验结果指出了开发的控制器具有令人满意的控制效果。该控制器适用于多关节服务机器人的柔性控制和嵌入式控制系统。

参考文献:

[1] Chia S H,Su K L,Chien T L.Develop an internet based home security robot[C].Honolulu,Hawaii,USA:The IASTED Int Conf on Robotics and Applications,2006:311-316.

[2] 熊光明,赵涛,龚建伟,等.服务机器人发展综述及若干问题探讨[J].机床与液压,2007,35(3):212-215.

[3] 姬振营,张杰.嵌入式运动控制器在网络化交流伺服系统中的应用[J].伺服控制,2007(3):40-42.

[4] 肖雄军,蔡自兴.服务机器人的发展 [J].自动化博览,2004(6):10-12.

[5] Takahashi Y,Kohda M.Simple humanoid biped robot with PIC microcomputer for university education[J]. Journal of Robotics and Mechatronics,2005,17(2):226-231.

[6] 李荣正,刘启中,陈学军. PIC 单片机原理及应用[M].北京:北京航空航天大学出版社,2006.

[7] 王建萍,费跃农.基于 PIC 单片机的新型电动卡丁车控制器研究[J].电气应用,2006,25(6):49-51.

[8] 郭黎滨,侯玉玲,孟庆鑫,等. 基于 PIC 单片机的机器人灵巧手的控制系统设计[J].现代电子技术,2004,(15):62-63.