MDA技术在Web服务开发与集成中的应用

李长英,曹宝香,杨成伟
(曲阜师范大学 计算机科学学院,山东 日照 276826)

摘 要:模型驱动架构(Model Driven Architecture,MDA)是对象管理集团(OMG)提出的一个新的软件开发框架,它把应用开发的重点由编码转移到更高的抽象层次——模型(元数据)。针对现今Web服务开发模式中存在的问题,讨论了模型驱动架构技术在Web服务开发与集成中的应用,引出开发流程并且提出一个二者结合的模型集成架构,最后使用支持MDA的工具OptimalJ开发与集成了一个Web服务,从实例中分析它们结合的可行性与优势。

关键词:模型驱动架构;Web服务;OptimalJ
中图分类号:TP311.52 **文献标识码**:A **文章编号**:1673-629X(2008)05-0212-04

Application of MDA in Developing and Integrating Web Services

LI Chang-ying, CAO Bao-xiang, YANG Cheng-wei
(Department of Computer Science, Qufu Normal University, Rizhao 276826, China)

Abstract: MDA(model driven architecture) is presented by OMG(Object Management Group). It shifts the emphases of the application development to a higher abstract level-model (metadata) according to code. In view of the problems in Web services development patterns now, discusses how the model driven architecture can be used to develop Web services. Then the development flow based on MDA is induced and a model integration architecture coming from the combination of the two is given. Finally it uses OptimalJ based on MDA to develop and integrate a Web services and analyzes the feasibility and superiority of their combination from the example.

Key words: model driven architecture; Web services; OptimalJ

0 引 言

随着应用软件与网络的结合日益深入,人们都希望通过网络来获取自己所需要的服务和软件,Web服务就是在这样的背景下应运而生的。Web服务是自包含、自描述、模块化的新型Web应用程序分支,它提供了一种标准方法将访问机制从实现中分离出来以实现远程调用的业务功能,即可以在Web中被描述、发布、查找以及通过Web来调用。

根据Web服务开发工具行业提供的不同机制,现今的Web服务开发可总结为三种开发模式[1]:"自底向上"模式、"自顶向下"模式和"往返"模式。"自底向上"模式是从编写Java代码开始生成WSDL文档,即将创建的POJO(Plain Old Java Object)对象或无状态会话EJB组件实现为Web服务。然而,提供程序端的数据类型应当包含除数据传输对象外更多的业务逻辑,而这样的业务逻辑无法在请求程序端重新建立,所以必须实际设计Web服务。"自顶向下"模式从开发与Web服务领域相关的XML模式开始,然后为该Web服务创建一个WSDL文档。它可以使用XML模式定义(XSD)来定义可互操作的数据标准,由业务需求驱动设计Web服务。但开发者需详细了解WSDL和XSD的结构,因为必须自主开发或操作这两者。往返开发是综合上述两种开发模式,但会使开发流程变得复杂。

新一代的Web服务开发应该由业务需求驱动设计Web服务,通过细粒度调用来减少网络流量。现有的Web服务开发除了存在上述开发过程中的各种缺陷外,另一方面,开发完成后,因为Web服务技术本身以及开发工具平台的不断变化导致原有Web服务迅速废弃,以及极大的更新与集成旧系统的工作量。

收稿日期:2007-08-28
基金项目:山东省科技攻关项目(2006CC2301001)
作者简介:李长英(1983-),女,山东青州人,硕士研究生,研究方向为高级数据库技术与系统集成;曹宝香,教授,研究方向为中间件、网络数据库、CAD。

1 基于MDA的Web服务开发与集成架构

1.1 模型驱动架构

MDA(Model Driven Architecture,模型驱动架构)是OMG提出的新的软件开发理念,它的目标是以模

型驱动的方式在一个比技术的实现更抽象的层面来设计应用,核心思想是抽象出与实现技术无关的、完整描述业务逻辑的平台无关模型(Platform Independent Model,PIM),并针对不同实现技术平台制定多个映射规则,然后通过映射规则及辅助工具将 PIM 转换成一个或多个与具体实现技术相关的平台相关模型(Platform Specific Model,PSM),最后,将经过充实的 PSM 转换成代码。由此可见,建模和模型映射技术是 MDA 的核心,而它们主要是在 OMG 定义的各种标准的支持下实现,主要有:公共仓库元模型 CWM,元对象设施 MOF,统一建模语言 UML 和 XML 元数据交换机制 XMI 等。这些支撑标准成就了 MDA 管理和集成不同元数据的统一框架[2]。

文中将利用支持 MDA 的工具 OptimalJ,相对于原有设计站在更高的抽象层来设计 Web 服务。针对 Web 服务天生多平台的特性,MDA 方法使得规范 Web 服务功能的同一个 PIM 可以通过点变换或辅助变换标准变换到特定平台[3],从而在多平台上实现,减少和避免了面临技术变迁时系统的混乱。同时,它可以使不同的 Web 服务应用程序通过清晰的模型联系而集成在一起,从而促进了集成性和互操作性。

1.2　基于 MDA 的 Web 服务开发流程

基于 MDA 方式的 Web 服务开发,同样遵循 MDA 的开发流程[4]。

总体来说,开发中需要重点解决以下 4 个问题:

(1)在得到了使用计算无关模型 CIM 描述的业务系统需求后,如何实现由 CIM 到 PIM 的转换;

(2)由 PIM 转换为 PSM 时,需要定制一个适合业务特征的开发与集成体系架构;

(3)基于一定的映射规则,如何完成从 PIM 到 PSM 的自动转换,这是应用 MDA 开发 Web 服务最为关键的一步;

(4)如何完成从 PSM 到代码的自动生成,并保证 PSM 与代码表现一致。

上述问题的解决:

具体的业务模型到 PIM 的转换,就是以用例(Use Case)图做驱动,使用建模工具为业务应用信息、领域相关信息和数据信息创建一个 UML 表示的 PIM,其中包含不同业务组件、组件间的结构与接口以及依赖关系的模型。

传统的分布式体系结构都是将整个应用系统分为三层[5]:一个侧重表示的前端,一个提供业务功能的中间层,以及一个侧重数据和传统系统的后端。这里,将 Web 服务开发与集成架构演化为四层(如图 1 所示)。与根据传统的三层系统架构相比,表示层与业务服务和数据层的交互逻辑被分离出来放在工作空间层(Workspace Tier),工作空间层用来隔离系统功能的提供者和使用者,作为系统不同模块之间的调用接口。它可以使同一个业务逻辑能够处理不同的客户端请求,也就是说,通过设计该层的 Facade,同一个业务服务可以被暴露为 Web 服务、Web 页面、应用程序界面等等。可以在业务层(Business Tier)中定义业务服务的核心逻辑的 PIM,因为这个核心逻辑的 PIM 与具体的暴露服务的实现技术(WSDL、HTML、WAP 等)无关,从而可以长久地维持其存在价值,仅在业务环境发生变化时才需要变更。并且可在不同环境下重用它们。

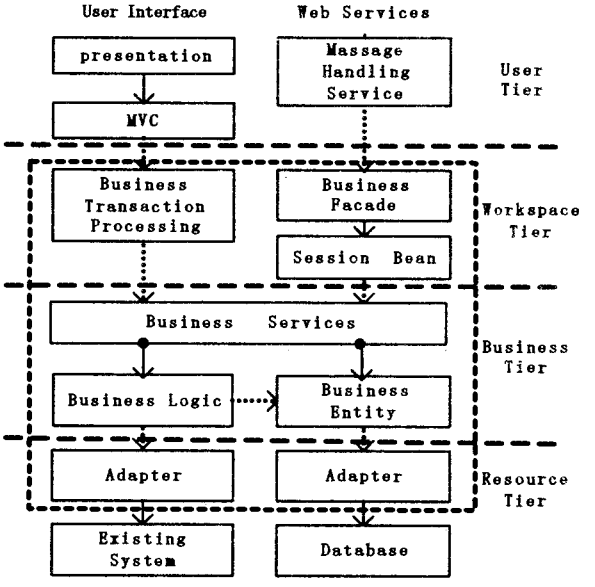

图 1　Web 服务的集成架构

根据一定变换规则完成从 PIM 到 PSM 和代码的转换取决于所选 MDA 工具的模型转换技术。Compuware 公司的 OptimalJ[6]完全实现了模型驱动架构的范式,并扩展了转换模式,它把模型又分为三大块:域模型(Domain Model)、应用模型(Application Model)和代码模型(Code Model),分别对应 MDA 的 PIM、PSM 和代码。OptimalJ 通过技术转换模式、执行转换模式分别支持模型-模型(即从域模型到应用模型)转换和模型-代码(即从应用模型到代码)转化,同时,通过功能模式提高可重用性,从而提高生产力。开发者可以采取 OptimalJ 默认定制的转换模式实现转换,也可以通过使用 OptimalJ 提供的模板模式语言(为开发执行模式定制的语言集)来定义元数据,从而创建和修改技术模式、执行模式和元模型,全面控制架构设计和代码生成。从这个意义上讲,OptimalJ 并不是一个黑匣子式的代码生成器。基于 MDA 方式的 Web 服务开发与集成的具体的实现技术,将结合下面的实例详细说明。

2 使用 OptimalJ 开发与集成一个 Web 服务

OptimalJ 对通常的应用抽象出了一个 CRM(客户关系管理系统)模型,它包括了构建一般系统的通用架构,这里用它作为文中实例的基础(限于篇幅没有列出),其 PIM 的核心类图包括:Customer 类,Call 类和 ServiceAgreement 类。其中 Customer 类是根类,与 Call 类有聚合关系,与 ServiceAgreement 类有关联关系。

要创建的 Web 服务包含一个名为 getServiceLevel 的操作,这里需要在通用架构上增加一个 Web 服务组件,输入参数是 customerId,返回值是客户 ID 的服务等级(BRONZE、SILVER 或者 GOLD)。最终产生一个 WSDL 文件,提供 Web 服务的描述。步骤如下:

(1)定义 Web 服务域模型。根据上述核心类图定制的域模型如图 2 所示。该域模型包括两个重要的模型 class 模型和 service 模型,通过 class 模型可以添加 Web 服务所需的一些元素,service 模型可以添加一些服务。将给名为 CustomerSvc 的 service 模型增加操作 getServiceLevel 来对它进行扩展。在下面的几个步骤中这个 Domain Service 将被转化为一个 Web 服务。CustomerSvc 的根类是 Customer,类 Call 和类 ServiceAgreement 作为根类的成员的方式包含在根类中。

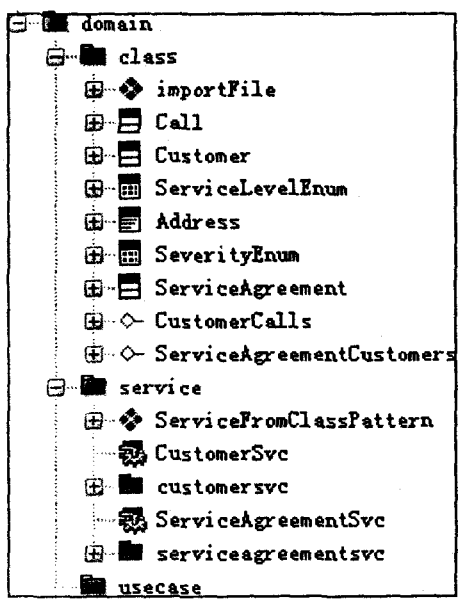

图 2 Web 服务域模型

(2)生成各种应用模型。选择执行 OptimalJ 菜单栏中 Model→Update All Models,这将会根据先前定制的 Web 服务集成架构产生应用模型。其中 class 模型转换为 EJBEntityComponents,service 模型转换为 business. logic. ejb 模型中的 EJBSessionComponent,此会话组件包含域服务操作转换成的 BusinessMethods。除此之外,应用模型包中还主要包括 Web 模型(包含 WebModule 和 WebOperation)、Common 模型(包含 class 模型定义的数据类型)和 DBMS 模型(包含表示关系数据的框架),接着选择 Model→Generate Model→Generate Web Service from EJB 生成 Web 服务集成模型,该集成模型最终将产生 WSDL、XML Schema 文件和框架代码。

(3)产生服务器端代码、WSDL 和 XML Schema。选择执行工具菜单栏中 Model→General All→Code,将生成可执行的 J2EE 代码以及 WSDL 文件和 XML Schema。此时,开发者可以定制应用程序的用户接口或实现附加的业务规则。下面将添加 getServiceLever 操作的服务逻辑,在[Code Model]视图中,选择 crm. application. business. logic. CustomerSvc 组件,双击 getServiceLevel 方法。在源代码编辑窗口中的 free block 代码段,输入下述代码并保存:

```
returnValue="No customer found";
try{
    CustomerDataObjectCollection c = retrieveByProfileOnKey(custid); /* 返回所有 key 和 profile 相匹配的 CustomerDataObjects */
  Iterator it=c.iterator();
  while (it.hasNext()){
   crm. application. business. common. customersvc. CustomerDataObject
  cdao = (crm. application. business. common. customersvc. CustomerDataObject)
  it.next();
   crm. application. business. common. customersvc. ServiceAgreementDataObject
   sadao = cdao. getServiceAgreementServiceAgreementDataObject(); /* 返回所有跟 Customer 相关的 ServiceAgreement */
  returnValue=sadao.getServiceLevel().toString(); /*方法 getServiceLevel 返回一个枚举类型,并转换为 String 类型 */
        }
}catch(Exception e){};
```

(4)测试 Web 服务。在编译生成的代码后,开启 OptimalJ 自带的 Solid 数据库,从 DBMS 模型生成的 SQL 文本中建立数据库表,选择"Test→Start EJB Server"启动应用服务器,从生成的主界面(如图 3 所

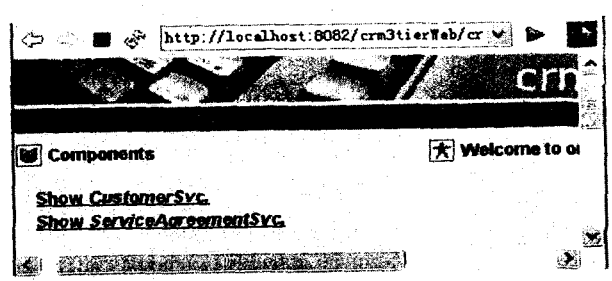

图 3 主界面

示)中,选择 Show CustomerSvc,将打开输入参数界面(如图4所示),要求输入客户 id,点击 Invoke_getServicesLevel 按钮来获得客户服务等级,返回结果如图5所示。

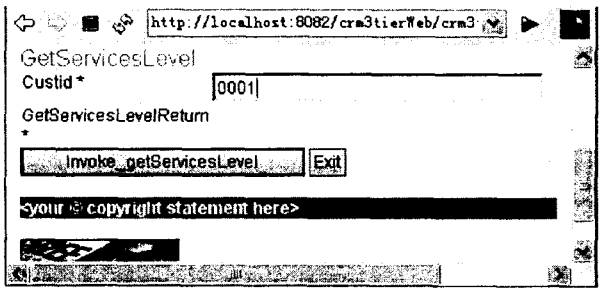

图4　输入参数

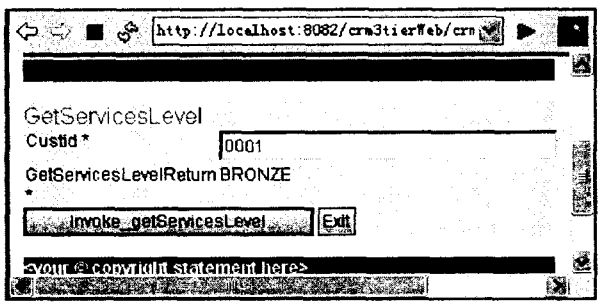

图5　返回结果

(5)集成 Web 服务。在测试完成之后,这个 Web 服务就可以集成了。借助 OptimalJ 导入第三步产生的 WSDL 文件(即包含这个 Web 服务的 Web 应用程序的 URL 地址)并创建一个客户端组件 WSClientComponent,用来关联服务组件和 Web 服务端口,接着生成 Web 服务集成模型中的域模型和应用模型,生成和编译代码,最后需要启动应用程序服务器和 Web 服务器测试并调用这个 Web 服务。这样就完成了 Web 服务的实现与集成。

从以上示例可以看出,采用 MDA 的开发模式,以 PIM 和 PSM 的变换为驱动,能够在一定程度上便捷和有效地开发 Web 服务。

3　结束语

应用 MDA 理论为指导,提出了 Web 服务的一种模型集成架构,并基于这个架构使用 OptimalJ 开发平台实现 Web 服务的开发与集成,从而验证了 MDA 理论应用于 Web 服务开发的可行性和便捷性。

总之,MDA 为开发与集成 Web 服务提供了全面的、结构化的解决方案。面对技术和平台的不断变化,基于 MDA 标准的应用使设计的 Web 服务不依赖于技术细节,能够提高 Web 服务的开发效率,减少应用的移植及维护成本。随着 MDA 的不断发展与成熟,它将成为新一代的软件开发的发展趋势。

参考文献:

[1] Flurry G, Modh M. Web services development patterns[EB/OL]. 2005. http://www.ibm.com/developerworks/Websphere/library/techarticles/.

[2] 范玉顺,李建强.企业集成与集成平台技术[M].北京:机械工业出版社,2004:78-82.

[3] 张德芬,李师贤,古思山.MDA 中的模型转换技术综述[J].计算机科学,2006,33:228-290.

[4] Frankel D, Parodi J. Using Model Driven Architecture to Develop Web Service[EB/OL]. 2002. http://www.omg.org/.

[5] 徐有威.模型驱动架构技术在分布式多层系统中的应用[D].湖北:武汉大学计算机学院,2004.

[6] OptimalJ White Paper: OptimalJ and Model Driven Architecture (MDA) [EB/OL]. 2005. http://www.compuware.com/products/optimalj.

(上接第211页)

输和视频流处理等技术。针对边海防系统的透雾问题,本系统中采用软件增强的方法,自动跟踪在上位机中通过软件采用相关跟踪算法实现,这样可以降低成本。如果要提高图像增强和自动跟踪的速度可以采用硬件加以实现,但会增加系统成本。今后可以利用当前一些先进的图像探测和处理技术,对现有系统进行改进提高,更好地为边海防建设服务。

参考文献:

[1] 李立仁.浅谈海防监控中的雷达与光电互动和透雾功能[J].激光与红外,2006,36(12):1155-1156.

[2] Stark J A, Fitzgeralid W J. An Alternative Algorithm for Adaptive Histogram Equalization[J]. Graphical Models and Image Processing,1996,58(2):180-185.

[3] Land E H. The Retinex theory of Color Vision[J]. Scientific American, 1977,237(6):108-128.

[4] Aggarwal J K, Nandhakumar N. On the computation of motion from sequences of images-A review[J]. Proceedings of the IEEE, 1988,76(8):917-935.

[5] Wang L, Hu W M, Tan T N. A survey of visual analysis of human motion[J]. Chinese Journal of Computers, 2002, 25(3):225-237.

[6] Bradski G R. Computer Vision Face Tracking For Use in a Perceptual User Interface[R]. Santa Clara, CA: Microcomputer Research Lab, Intel Corporation, 1998.

[7] 朱永松,国澄明.基于相关系数的相关匹配算法的研究[J].信号处理,2003.12(4):531-534.