基于UML的Web系统中间层组件的设计与实现

陈刚 陈志刚 杨路明 （长沙中南大学信息科学与工程学院 410083）

摘要：随着Web应用复杂性的日益增长，中间层组件的应用越来越广泛。本文介绍了一种使用UML为Web应用系统中中间层组件建模的方法，从而完成对组件的面向对象的设计与实现。

关键词：Web应用 中间层组件 统一建模语言 面向对象分析 面向对象设计

1 前言

一个系统如果拥有Web服务器(或通过Web服务器接受用户输入的应用服务器)，并可以通过Web游览器来改变服务器业务逻辑的状态，这样的系统就是Web应用。Web应用中起着重要作用的概念有：页、脚本、组件、表单、框架等。其中，中间层组件的应用越来越广泛，并逐渐成为Web应用设计的核心。在这种Web应用中，并非所有的业务逻辑都是通过Web页中的脚本来实现，而是利用位于用户界面与持久系统（如数据库服务器）之间的中间层组件。它通常由一组能在服务器端运行的已编译好的组件组成，封装了所有的业务逻辑。其优点是能共享整个应用中的业务功能，提高性能。其性能好坏将决定整个Web应用的成败。本文从这点出发，结合笔者在网站开发中的实践，介绍一种基于UML的中间层组件的设计与实现。

2 UML基础

统一建模语言UML（Unified Modeling Language）代表面向对象技术的最新发展，它统一了一些主要的面向对象方法，如Booch，OMT和OOSE等，融合了当今技术发展的成果以及未来的发展趋势，如组件技术、分布式计算等。UML为软件系统的可视化分析与构建以及文档的产生提供了一致的语言。UML能够满足如下要求：

(1) UML通过提供公共对象分析(OA)和对象设计(OD)元模型(metamodel)的正式定义来表达OA&D模型的语义，其中包括静态模型、动态模型和体系结构模型。

(2) 为OA&D工具（如Rational Rose）所产生的模型之间的交互机制提供接口定义语言（IDL）规范，支持用户模型的动态创建。

(3) 提供通俗易懂的标记来表达OA&D模型，UML标记能够对UML语义提供一致的完美图形表达。

UML通过提供不同形式的图形来表达从软件分析开始的软件开发全过程的描述。主要图形有以下几种：

① 静态结构图（Static Structure Diagram）。包括类图（Class Diagram）和对象图（Object Diagram）。类图是静态结构模型的图形化表示，是（静态）声明的模型元素集合。而对象图是类图的实例。着重描述系统中内部结构以及系统中存在的事物与事物之间的相互联系。

② 包（Package）。在各种图中用来对一组模型元素打包的元素。可包含从属包或普通的模型元素。比如可把某些类封装于一个包中。

③ 使用案例图（Use Case Diagram）。用与表现活动者（角色）与使用案例（Use Case）之间的关系。从用户的角度对系统的功能需求进行描述。而Use Case则是一个系统或一个类提供的紧凑的功能单元，它是由系统与一个或多个活动者之间交换的消息序列以及系统执行的活动共同体现的。

④ 顺序图（Sequence Diagram）。展示对象之间按时间顺序排列出来的交互。强调对象之间消息发送的顺序以及对象之间的交互。但不表达对象之间的关联关系。

⑤ 合作图（Collaboration Diagram）。表示在一些对象之间组织的操作和它们之间的链（对象关联的实例），描述对象之间的关系以及对象之间的联系。表明了在特定上下文中一组相关对象之间的协作，以及这组对象为产生所要求的操作或结果而进行协作时所交换的消息（分别表示调用、控制流和异步等3种不同的消息）。

⑥ 状态图（Statechart Diagram）。描述对象或交互在生命周期中受激励时的状态序列以及它们的反应与活动。

⑦ 活动图（Activity Diagram）。状态图的一种变形。

他的状态表示操作所执行的活动，其转换是由操作的完成而触发的。它表示了一个过程本身的状态机，过程是对类中一个操作的实现。

⑧ 实现图（Inplementation Diagram）。表现实现方面的问题。包括源代码结构和运行时的实现结构。实现图分为两种：一种是代表源代码自身结构的成分图(Componet Diagram)；另一种是表示运行时系统结构的展开图（Deployment Diagram）。

基于UML的开发模式是以使用案例驱动，以体系结构为中心，迭代式增量开发。在每一次迭代过程中沿着面向对象分析、面向对象设计、面向对象实现、软件测试与交付的路线进行。

3 基于UML的中间层组件的开发过程

中间层（商业服务层）组件其实就是很多类组成的集合。这些类可以通过创建Activex.Dll或Activex.EXE来实现。而这些正是UML所擅长解决的问题。利用UML提供的各种图形表达，可以对从需求分析开始的整个开发过程进行描述。

3.1 面向对象分析

首先是利用UML的使用案例图捕获投资者的用户需求。由于使用案例图使用图形表示法表达了系统的功能，使分析人员很容易就能与用户进行交流和商讨，从而就需求达成共识，共同完成一份用户、开发者双方均认可的需求规格说明。而且使用UML可视化建模系统便可生成需求规格说明文档。

在用户需求得到反复论证后，就可以分析Web应用业务中可能涉及的对象，找出系统中的对象类，以及各对象类之间的关系。笔者在实际的网站开发中曾使用一种称作“文法分析和E-P图”的方法完成这个过程。即采用类似语文中的文法分析的方法找出系统中会涉及到的对象以及对象的属性和它们之间的关系。大致实施方法如下：将业务过程的每一个步骤用1至2句话描述出来，在这1至2句话中找出中心名词，即为这步骤所涉及的对象，再从这些句子中找出描述中心名词的限定词作为对象属性。画出E-P（如图1）。对画好的若干E-P图进行相应的合并，调整优化处理后可以准确和充分地分析出对象、其属性以及对象间的关系。

接着使用UML类图确定系统所有对象类的基本框架，包括类及类之间的静态关系（如关联关系、聚集关系、依赖关系等）；使用顺序图和合作图、活动图和状态图（当对象行为复杂时结合这两种图）描述对象行为和对象间的合作关系，并不断地对类图中类的描述（属性和方法）加以修正和补充。利用这些图我们可以得到面向对象分析的三要素：对象模型、动态模型和功能模型。

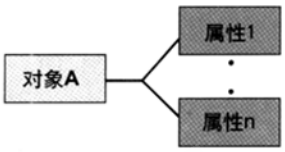

图 1 E-P图示例

在Web应用中间层组件的面向对象分析过程中有两点值得特别指出：

(1) UML对于复杂的使用案例表达能力不强。比如当一个使用案例与多个活动者产生交互时，如果在相应的顺序图中直接画出所有活动者与中间层组件中对象类的交互关系，不仅会造成顺序图中链线纵横交错，失去清晰度，从而使对象类的属性和方法以及它们之间的关系难以确定，而且还会在ASP代码中出现很多创建对象的语句，从而引发诸如何时创建哪个对象、何时销毁哪个对象、何时使用哪个对象等难以理清的问题，增加编码和维护的难度。这个问题可以通过增加一个Presentation类（表现层类）来解决。这个类位于中间层组件其他对象类与活动者之间，专门负责活动者与组件的交互。访问者通过游览器访问时，只与Presentation打交道，由Presentation类统一控制对象的生命期和动态网页的呈现。实际应用中，此类的增加，不仅在很大程度上解决了上述问题，简化了顺序图，而且此类的状态图可以成为编写ASP代码的依据。

(2) 由于中间层组件多由Activex.DLL或Activex.EXE实现，特别是Activex.DLL属于进程内调用，如果将所有对象类对数据库的访问封装于对象类自身当中，当访问量集中增加时，可能会造成后台数据库服务器连接数大量增加，浪费资源，降低系统运作速度。可以通过增加一个类——DBAccess类（数据访问类），专门负责对后台数据库的访问。这个类必须囊括组件所有对象类对数据库的访问（但只需要一个连接）作为其自身的方法。这样，一个访问者登录只会产生一个连接。并且当系统扩展增添新类时，只需相应的扩展DBAccess类便可。

3.2 面向对象设计

设计阶段将考虑系统中的所有细节，综合利用UML提供的各种图形，对中间层组件进行设计，对分析阶段结果进行扩展、细化和修正。

3.2.1 利用UML的包对组件进行结构设计

将分析阶段定义的类根据其功能放进不同的功能包

中，并定义包与包之间的关系。包有利于描述组件的逻辑组成，以及各部分的依赖关系。遵照高内聚低耦合的原则，各个包之间的关系应简单明了，减少包之间的依赖关系，尤其是双向依赖关系。可定义如下几个包，如图2所示：

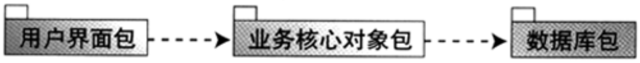

图 2　各包之间的关系

（1）用户界面包。该包主要提供用户使用中间层组件，访问系统数据的界面。将Presentation类封装于此包中，以动态创建网页和中间层组件中的对象。

（2）数据库包。将DBAccess类封装于此包中，实现对数据库的访问。

（3）业务核心对象包。包含中间层组件对象类中除上述2个类之外的其他对象类。

3.2.2 详细设计

从实现的角度对系统进行更详细的描述，可以利用UML提供的各种图形，如类图、状态图、合作图、活动图顺序图等，对分析阶段产生的对象类进行扩展和细化，形成足够详细的有关类属性、接口和操作的规格说明。

3.3 面向对象的实现

3.3.1 Activex.Dll或Activex.exe代码编写

在详细设计后，可以利用UML生成Activex.Dll或Activex.exe的源代码框架（Rational Rose可以生成多种语言的源代码框架），充分利用开发工具的面向对象特性，进行编码工作。在这个过程中可以对设计阶段所建的模型进行适当的修正，关键是一定要保持模型和编码的一致性，以便于维护。

3.3.2 面向对象数据库的实现

现在的数据库服务器多是关系型数据库系统（RDBMS）。它的一个主要缺陷是不能支持复杂对象，也不能直接支持对象的存取。而对象模型最终是由二维表及表间的关系来描述的，所以，对象模型与关系模型的转换成为面向对象实现的必不可少的环节。

这可以采用扩充关系型数据库的技术。扩充关系数据库以支持对象有两种方式：

（1）直接在关系数据库上添加面向对象接口。在实现上存在一个对象至关系的转换机制，将上层的面向对象模式转换为关系存储模式，存放到关系数据库中。其结构如图3。

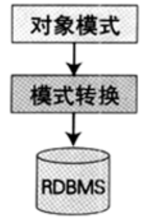

图 3　增加面向对象接口

（2）扩充允许关系表中的字段值包含有对象指针，这样对象就构成关系表中一种新的数据类型，对象数据的操作将是在关系库外进行。这种方式可避免模式转换的开销，但使对象查询功能受到一定的限制。

笔者曾在具体的中间层组件开发中使用第一种方法。主要利用UML类图，使用图4所示的转换模型。

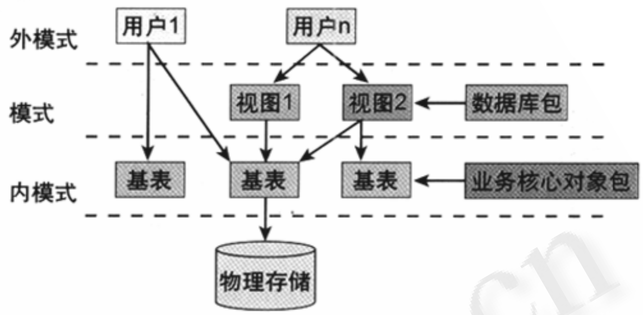

图 4　对象模型到关系模型的转换

数据接口包中的类主要是实现对数据库的访问，映射为关系数据库系统的外模式相对简单。关键是业务核心对象包到模式的映射，实际上是类图中每个类到基表的映射。这需要十分清楚类之间的关系，然后进行具体映射。大致原则是：对于类图中每个类，根据不同的具体情况，映射成一个或多个表，类中属性与表中字段相同；对于不同类之间的关系，如泛化、聚集、关联等，应采取不同的方法进行映射。

4　软件测试与交付

测试包括单元测试、集成测试、系统测试。不同的测试使用不同的UML模型作为测试依据：（1）单元测试使用类图和类的规格说明。（2）集成测试使用组件图和合作图。（3）系统测试使用案例图来验证系统的行为。

测试阶段不可避免地会发现错误，对生成的代码进行修改后，可能造成模型和代码的不一致，此时可使用UML逆向转换系统将修改结果映射到模型，使得组件的

（下转第41页）

(上接第 38 页)

扩充、增删和维护得以顺利进行。从而可以进行再次的分析和修改，进入新一轮的迭代。

5 结束语

复杂Web应用中中间层组件由于封装业务规则，其正确性与健壮性决定了整个应用是否能正常运作。而UML是一种功能强大的面向对象的可视化建模语言，采用一整套成熟的建模技术，为中间层组件的正确设计与实现提供了强有力的支持。■

参考文献

1 *Rational software Corp。UML Summary V1.1, September 1997*

2 *Rational software Corp。UML Notation Guide V1.1, September 1997*

3 *Jacobson I。Object—Oriented Software Engineering：A use case Driven Approach。New York：Addison—Wesley Publishing Company, 1992*

4 *冯玉琳，黄涛，倪彬　对象技术导论　北京：科学出版社，1998*