利用设计模式构造高效 Web 应用开发的模型

彭 雷，李伟生
(北京交通大学 计算机与信息技术学院，北京 100044)

摘 要：网络的普及和办公自动化的需求推动了 Web 应用开发，从应用软件的设计角度，开发高效、易用、易维护和可复用的 Web 应用成为关键。结合设计模式和 Java Web 开发技术，讨论如何构建高效 Web 应用开发的模型，并从数据流的角度进行了详细分析。

关键词：设计模式；开 - 闭原则；策略模式；Web 应用；J2EE
中图法分类号：TP311.52 **文献标识码**：A **文章编号**：1001-3695(2005)01-0174-03

Constructing Effective Model for Web Application Development with Design Patterns

PENG Lei, LI Wei-sheng
(School of Computer & Information Technology, Beijing Jiaotong University, Beijing 100044, China)

Abstract: The popularization of network and office automation has been promoting the development of Web application. According to the principle of software design, the key to Web application is to develop efficient software which is easy to use, maintain and reuse. Joined with design patterns and Java technology, how to construct an effective model for Web application development is discussed and analyzed from the view of data stream.

Key words: Design Patterns; OCP; Strategy Pattern; Web Application; J2EE

网络的普及和办公自动化的需求推动了 Web 应用开发，Web 应用程序有三个特点：①普遍使用三层体系结构(或多层结构，本质类似三层结构)，即浏览器(表示层)、Web 应用程序(逻辑层)、数据库(数据层)等(图 1)；②多数应用涉及的技术领域知识较多，如网络、数据库、分布计算、通信等，但复杂程度不高；③更重视业务逻辑，而不是应用逻辑，有意义的数据是应用的重点。

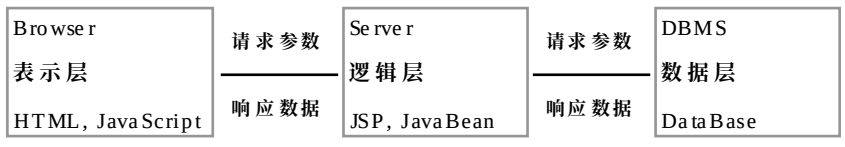

图 1 Web 应用的层次结构

这样，提高 Web 应用开发的效率并获得可靠性和易用性成为关键。设计模式是模式的一种[1]。模式由建筑领域中 Christopher Alexander 提出："每个模式描述一个在我们的环境中不断重复出现的问题，然后描述这个问题的方案核心，使你可以多次使用同一方案，而不必进行重复工作。"以 GoF 著的《Design Patterns: Elements of Reusable Object-Oriented Software》为代表，模式被引入软件工程领域，成为近年来被逐渐重视和采用的软件设计技术。设计模式，就是情境中标准设计问题的重复性解决方案。简单说，就是针对特定应用要求的一些经过实践验证的工作经验。

J2EE 是分布式网络计算环境，包含一系列重要技术，为 Web 应用开发提供了丰富的组件，简略看，其中 JSP 和 Servlet 一方面可以产生表示逻辑，另一方面可以封装业务逻辑，与下层的 JavaBean 通信，独立或者共同利用 JDBC 访问各种数据库；而 EJB, JNDI, RMI 组件更是提供了企业级分布计算功能，结合面向对象数据库、应用程序服务器等软硬件技术、配置，可以构建强大的应用服务平台。本文讨论利用设计模式，结合 J2EE Web 应用开发，构造一个易维护、可复用的开发模型[2]，易维护、可复用也是软件工程的一个要求。本文不再列举 J2EE Web 开发的结构及其相关组件的配置，数据是 Web 应用的核心，本文就从数据的角度，结合 Java 技术展开讨论。

1 设计与分析

多层结构的 Web 应用，业务逻辑的核心仍然是对数据的增删改查，数据库访问通过网络进行。本文以这四种操作为基础先介绍数据组织，再扩展开来从多个角度分析如何提高开发效率。

1.1 数据组织包含三方面的内容

(1) 数据映射。它包括两层含义：一是把业务逻辑中的对象映射到 Java 类，即把业务逻辑对象的特征映射到类的数据成员，把业务逻辑对象的操作映射到类的方法成员；二是把 Java 类映射到数据库存储逻辑，即把业务逻辑的特征、操作或类的数据、方法组织成数据库对象如库表、应用程序等。有关映射的顺序或者方向问题，将在最后进行讨论。

数据库设计的内容中，有许多重要的规则保证数据库设计的高效以及数据完整性，一致性等要求(如 1NF, 2NF 等)，这些内容不是本文的重点，本文主要讨论类与数据映射的几个组织原则：

①每个类映射到一个表，其中的数据成员对应表中的字

收稿日期：2004-02-19；修返日期：2004-04-27

段, 而方法成员映射抽象的处理逻辑, 类和成员的名称与数据库中的各种元素名称应当对应。根据“依赖抽象而非实现”的原则, 业务逻辑应合理地抽象成为类或接口方法。这样, 类的实例就代表了一个表中的一行或多行记录及相应的数据处理。

②当应用带有统计等多表或跨表数据操作功能时, 应当产生一个容器类, 它包含某个(些) 映射单独数据表的类的引用及相应操作, 从而封装了涉及多表或跨表的数据及其操作。这种数据组织结构如图 2、图 3 所示。

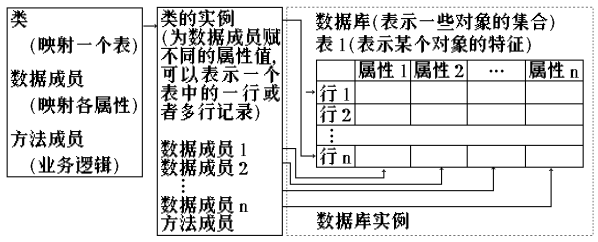

图 2　类(对象的抽象) 与数据库元素之间的映射

③类的业务逻辑操作代表对存储数据的操作, 可以分为两类: 一类包含对数据库操作的业务逻辑, 例如存取某种业务数据, 这类操作的实现独立在一个或几个数据库接口类中, 二者之间的关系是松耦合的; 另一类是在利用数据库中数据创建出类的实例后, 对实例保存的数据进行操作的业务逻辑, 例如统计分析等, 这类操作是耦合在类中的。以下对数据接口的分析可以使这种结构更清晰。

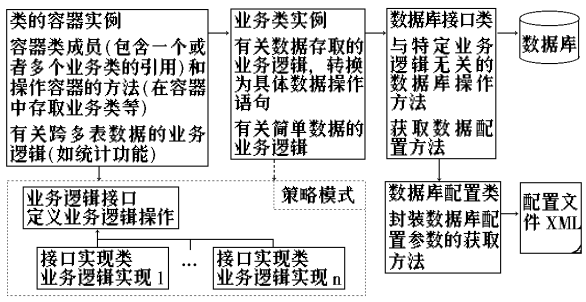

图 3　类或对象的结构

(2) 数据接口。它在业务逻辑类和数据库之间一般还有两层, 即数据库接口/配置类。业务逻辑类理解业务的特性, 把与业务数据有关的操作转换为数据库可以理解的操作语句; 数据库接口类是一个通用模块, 它处理数据库的连接、执行 SQL 操作、返回结果和关闭连接等操作, 与业务逻辑无关; 而为了提高 Web 应用的可扩展性, 使其可以容易地在数据库系统之间移植而不影响上层的应用, 需要一个灵活的配置管理方法, XML 配置因为具有良好的语义表达能力和跨平台特性成为最好的选择, 数据库配置类即被用来动态地解析 XML 文件以获取和配置数据库参数。这样的数据接口设计如图 3 所示。

数据接口类的设计原则其实和设计模式对软件设计的一般性要求相似, 主要就是开 - 闭原则[3], 即一个应用软件应该总是对扩展开放, 而对修改封闭(Open-Closed Principle: Software Entities Should be Open for Extension but Closed for Modification)。满足开 - 闭原则, 抽象化是关键, 策略模式常用来指导这类设计: 如果有一组算法, 那么就将每一个算法封装起来, 使其可以互换(图 4) 。

Web 应用中的业务逻辑是可能有变化的部分, 我们不希望有了变动就去修改所有相关类的代码, 而是希望在增删功能时可以方便地插入和拔出功能模块。在以上的设计中, 对数据库操作的业务逻辑已经独立封装在一个数据库接口类中, 经过良好的设计而基本不会修改, 而且对于扩展很容易; 映射数据的类对数据的业务逻辑操作都是抽象的, 它们的具体操作在实现了统一接口的类中实现。这样当业务逻辑改变或者新的业务逻辑出现、原有业务逻辑放弃时, 就可以修改, 或者产生、放弃一个实现统一接口的类。

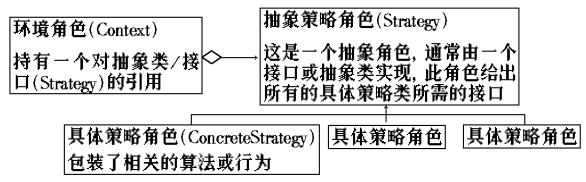

图 4　策略模式

(3) 数据流向。这个模型中的数据主要有两种, 即业务数据和控制信息, 可以用图 5 清楚地说明。

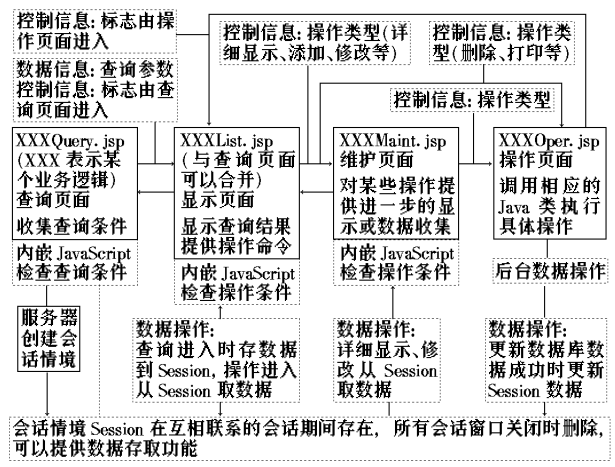

图 5　Web 应用模型中数据流图

模型的重点是尽可能把表示逻辑和业务逻辑相独立, 表示逻辑反映数据、业务逻辑处理数据、业务流程中数据的不同处理方式被监听和分析, 根据条件被导向相对独立的模块, 这使用了一些优秀设计模式的思想, 比如观察者模式和 MVC 模式(模型 - 视图 - 控制器模式) 。在 J2EE 体系中, JSP 一方面与 HTML 结合来表示数据, 一方面与 JavaBean, JDBC 等结合来操作数据, 而以何种条件, 何种方式操作数据常常是由 Servlet 控制的。在简单的 Web 应用中, JavaScript 也常常用来作为分析客户端操作类型, 向各个独立模块转发请求的工具, 并且具有加快响应速度, 减少与服务器通信, 降低服务器端负载的优点。这样设计的优点是可以用灵活的方式表示数据处理结果和在流程中加入取出业务逻辑。具体来看, 模型中主要有这样几个角色: 信息收集(查询页面和维护页面) 、数据表示(显示页面) 、检查(分析导向作用) 和数据处理(操作页面, 包含了组件对数据的存取) 。以下针对数据进行三方面的讨论。

1.2　数据共享与通信

当用户打开一个浏览器窗口连接到 Web 应用的时候, 就可以认为在客户机和服务器之间开始了一次会话 (Session) , 会话以此为开始, 从这个连接打开的其他所有连接窗口都处在共同的会话情境中 (Context) , 这些连接共享情境中的信息 (主要是环境参数) , 这是这个模型数据共享的一个方法

(图 5)。当用户进行一次查询时,访问一次数据库,所有得到的符合条件的数据构造成一个个业务对象先被存放在数据容器里(如 Vector),再放在会话情境中(Session Context)。在用户后继的操作中,除非开始一次新查询,显示都是从这个 Vector 中取出数据对象。当用户进行操作(增、删、改)后,返回显示操作后的数据时,是从 Vector 中取而不是再以某种条件进行一次数据库查询,这样可以加快响应速度,减少连接数据库等一些操作的开销。但同时就会自然涉及到一个问题,如何保持 Session 中的数据与数据库中的数据的前后台同步。

1.3 数据同步

上面涉及到的前后台同步的问题只是此类 Web 应用中的一类问题,可以在操作数据库后返回一个标志用来说明操作是否成功,如果成功,马上更新 Session 中的数据,否则 Session 中的数据也不更新,返回用户相应的操作结果信息。这样,Session 中的数据和数据库中的数据实际上是一致的,在操作成功时利用 Session 中的数据响应用户速度快,而且在按照新的条件获取数据前不必反复访问数据库。

1.4 数据安全

对于 Web 应用,必然涉及到安全管理的问题,除了用户的权限控制之外,还要进行数据的校验以及保护重要数据不被恶意获取,如校验和保护账号密码等关键信息。对于保存在数据库中的重要数据,可以利用其中某些数据产生一个校验码用于校验,也可以对其中某些数据进行加密加以保护,通常可以使用 SHA-1 算法(安全散列算法,由美国国家标准和技术协会提出,要求输入报文最大长度不超过 264bits,产生的输出是一个 160bits 的报文摘要)[4]。在一般的 Web 应用中,重要数据往往不是很大,而 SHA-1 算法具有速度快,简洁易用的优点,因此值得广泛采用。

2 讨论

2.1 性能提高

访问数据库,建立连接、操作、断开连接,这是在 Web 应用的运行过程中频繁发生的过程,而这个过程也是花费时间最多的。针对连接的高效利用,有连接池技术可以参考。另外,我们这里利用会话情境保存数据进行共享,已经加快了对客户端的响应速度。我们针对这个模型提出了一个方案,但该方案仍在讨论试验阶段中,该方案就是利用批量数据操作方法进一步减少对数据库的存取次数。这个方法假设对数据库的操作多数都能够成功,利用多线程技术,利用 Session 中的数据响应用户,给用户的感觉是这个操作已经完成,可以进行下一步的操作,同时对数据库的操作保存在一个任务队列里,由另外的并发线程操作数据库,在合适的时机使数据库中的数据与自身保持一致或者在操作无法完成时通知用户。

这个设想虽然可以大大减少对数据库的频繁连接操作,提高效率,但是它同样存在一些问题,主要就是如何尽可能地保证数据一致性,有两方面的一致性问题:①批量操作如果不能成功或者不能全部成功,那么用户之前看到的操作结果实际上和数据库不一致了,这虽然不会造成大的问题,但是会引起 Web 应用易用和可靠方面的问题;②Web 应用都是多用户的,属于多个用户的任务队列对同类数据的访问必须要受到限制,否则会造成大的问题。这需要数据库对批量操作中事务的支持以及对数据库连接或者数据库元素等资源进行互斥或者锁定的操作。

2.2 设计中的映射顺序和方向

前面已经提到,数据是 Web 应用的核心,以往的设计过程,往往也是以数据为中心的。首先看到的是一个业务逻辑包含多种分类数据,从而保存到不同的数据库表,然后用一组类来表示这些表,类的数据成员对应表属性保存属性值,类的方法成员对应对数据的操作,如图 6 所示的过程,这在传统的设计过程中被广泛应用。实际上,利用数据分析 Web 应用的构架,应该是在对 Web 所涉及的对象有了清楚的分析之后,因为任何一个有意义的数据,都是从属于特定对象的。

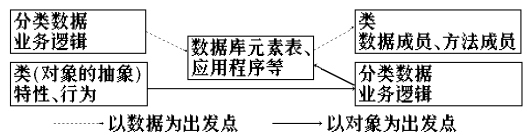

图 6 Web 应用的两种设计思路

对于现实问题,对象才是设计的核心。面向对象的思想能够更真切地反映事物的特征、行为以及事物之间联系等的本质,也符合人们的认知规律。从面向对象的角度来看,任何一个业务逻辑,都是多个不同的角色对象交互作用的过程,每个对象拥有一组反映自身特性的数据,一组控制特性的方法,业务逻辑的实现就是对象自身或多个对象之间利用数据通信的过程。利用面向对象的思想来指导 Web 应用设计,顺序与前者不同(图 6)。这不只是一个设计的问题。一个良好的设计观念能从根本上决定一个软件的开发效率、易维护性和可复用程度。面向对象的思想经过实践的检验,被证明是良好有效的软件工程理念,应该也必须提倡。

3 结束语

Web 应用的广泛开发,很大程度上提高了人们的办公自动化程度和工作效率,Web 应用开发本身,需要提高效率。基于此,本文讨论如何构件进行高效 Web 应用开发的模型,该模型已经被应用在笔者参加的“人工电话网络智能支撑平台”的设计中,取得了不错的效果,同时也有许多值得进一步思考的问题,并将继续完善模型的设计。

参考文献:

[1] rich Gamma, *et al*. Design Patterns: Elements of Reusable Object-Oriented Software[M]. Addison-Wesley, 1994.

[2] 阎宏. Java 与模式[M]. 北京:电子工业出版社, 2002.

[3] 张小勤, 滕至阳. 基于 MVC 设计模式的 J2EE 分布式应用系统模型 JMVC[J]. 计算机应用研究, 2003, 20(9): 63-64.

[4] [美] William Stallings. 密码编码学与网络安全: 原理与实践(第 2 版)[M]. 杨明, 胥光辉, 等. 北京: 电子工业出版社, 2001.

作者简介:

彭雷(1980-),男,硕士,研究方向为网络与数据库、分布式计算; 李伟生(1945-),男,副院长,教授,研究方向为网络与数据库、算法设计与分析。