

基于 J2EE 多模式的 B2B 电子商务系统设计

毛 力, 英恒松, 须文波

(江南大学信息工程学院, 无锡 214036)

摘 要: J2EE 模式提供了一组基于 J2EE 平台的对常见问题的解决方案, 使用 J2EE 模式可以显著优化系统性能, 有效增强系统的可维护性、扩展性和重用性, 大大缩短系统研发周期。该文用一个纺织行业 B2B 网上订购系统作为实例, 说明如何以最佳方式将多个 J2EE 模式连结在一起设计出强壮、高效的 B2B 电子商务系统。

关键词: J2EE; J2EE 模式; B2B 电子商务系统; 网上订购系统

Design of B2B E-business System Based on J2EE Multi-pattern

MAO Li, JIA Hengsong, XU Wenbo

(School of Information Engineering, Southern Yangtze University, Wuxi 214036)

【Abstract】 The J2EE patterns are a collection of J2EE-based solutions to common problems. Using J2EE patterns can optimize system performance remarkably and enhance numerous aspects of the system, including maintainability, extensibility, and reusability, and also can abridge the developing time greatly. This paper chooses a B2B online ordering system for textile vocation to discuss how to link patterns together in best practices to design robust, efficient B2B E-business system.

【Key words】 J2EE(Java 2 platform enterprise edition); J2EE patterns; B2B E-business system; Online ordering system

B2B(企业 - 企业)电子商务是当今电子商务中最重要、最具发展潜力的一种模式, 它使用Internet或各种商务网络来完成企业之间的商务过程, 从而达到提高效率, 减少库存, 降低采购、销售及售后服务等方面成本的目的。J2EE是一种利用Java 2 平台简化企业解决方案的开发、部署和管理相关复杂问题的体系结构, 是目前大型B2B电子商务的主要开发平台。J2EE确实能够帮助团队开发出功能非常强大的系统, 但是, 目前的事实却是, 在J2EE提供的抽象和服务与团队必须构建的最终应用之间仍然存在巨大的差距, J2EE模式正是能够不断缩小这种差距的解决方案^[1]。纺织行业B2B网上订购系统是无锡纺织信息化服务平台的子系统, 系统面向纺织行业, 全力为广大会员企业提供快捷、高效、安全、具有纺织品特色的各种网上订购功能。本文以该系统为例, 讨论如何以最佳方式将多个J2EE模式有机地组合在一起设计出结构清晰、易于维护、运行高效的B2B电子商务系统。

1 J2EE 模式

模式的概念最早由建筑专家 Christopher Alexander 提出: “每一个模式描述了一个在我们周围不断重复发生的问题, 以及该问题解决方案的核心, 这样, 就能多次地使用该方案而不必做重复劳动”。J2EE 模式专用于解决与 J2EE 平台相关的设计问题, 它所提供的对这些问题解决方案, 已经在不同的时间、不同的项目中被反复用于解决相似的问题, 因此具有强大的可重用机制。同时, 由于遇到相同问题, 使用相同的解决方案, 这样就使编写的代码具有一致性并易于理解, 因此也就提高了系统的开发效率和质量, 有利于系统今后的维护和扩展。另外, J2EE 模式还为软件设计者提供了一套共同词汇, 有助于在开发者之间通过共同词汇交流各自经验。

2 系统功能需求

纺织行业 B2B 网上订购系统要求具有以下主要功能:

- (1) 提供会员服务和管理功能, 包括会员注册、会员验证、会员信息管理。
- (2) 对产品进行科学的分类、存储、管理, 以便于客户快速从海量数据中找出所需产品。
- (3) 提供强大、方便、快捷的产品查询功能。
- (4) 提供购物车管理功能, 包括浏览、添加或删除购物车中的产品、生成、提交订单等功能。
- (5) 提供各种订单管理的功能, 包括对客户订单的审核, 生成各类订购信息报表, 订单查询等功能。
- (6) 为客户提供安全、可靠、便捷的在线支付功能。
- (7) 系统应制定一系列的价格策略, 例如基于客户角色提供产品价格策略、基于订购批量提供产品价格的策略、询价策略等。

3 系统设计

基于系统的业务性质和功能需求, 本系统采用基于 J2EE 的五层 B/S 体系结构(即将系统分为 5 层: 客户端层, 表示层, 业务层, 集成层和资源层), 系统由在线购物、后台管理、询/报价、在线帮助子系统组成, 简要说明如下:

- (1) 在线购物子系统, 包括产品查询、购物车、在线支付等子模块。
- (2) 后台管理子系统, 包括订单管理和处理、产品管理、资料管理和统计分析等子模块。
- (3) 询/报价子系统, 包括询价单管理(如制作/查询/修改/删除/收

基金项目: 无锡市“十五”科技攻关基金资助项目“面向纺织行业的信息化服务平台”(CI03003)

作者简介: 毛 力(1967 -), 男, 副教授, 主研方向: 计算机网络应用及信息管理; 英恒松, 硕士生; 须文波, 教授、博导

收稿日期: 2005-10-28 **E-mail:** wxmaoli@163.net

发)、报价单管理、在线洽谈等子模块。

(4)在线帮助子系统,提供以上各子系统的在线式、上下文相关帮助。

限于篇幅,下面仅介绍在线购物子系统,该子系统包含以下5个主要功能子模块。

(1)会员注册登录子模块:包括会员在线注册、登录、修改个人资料等功能。

(2)产品查询子模块:提供给客户按类别、关键字、品牌等进行的高级或一般的产品查询功能。

(3)购物车子模块:包括管理客户的购物车、生成并提交订单等功能。

(4)在线支付子模块:用银行支付系统进行在线支付,提供与银行之间联系的接口。

(5)客户订单查询子模块:包括查询订单执行状态,查询历史订单等功能。

4 多模式集成在系统中的应用

系统在表示层采用了服务到工作者(Service to Worker)模式;业务层采用了业务代表(Business Delegate)模式、会话门面(Session Façade)模式、应用服务(Application Service)模式、业务对象(Business Object)模式、服务定位器(Service Locator)模式和传输对象(Transfer Object)模式;集成层采用了数据访问对象(Data Access Object)模式,系统多模式集成框架如图1所示。

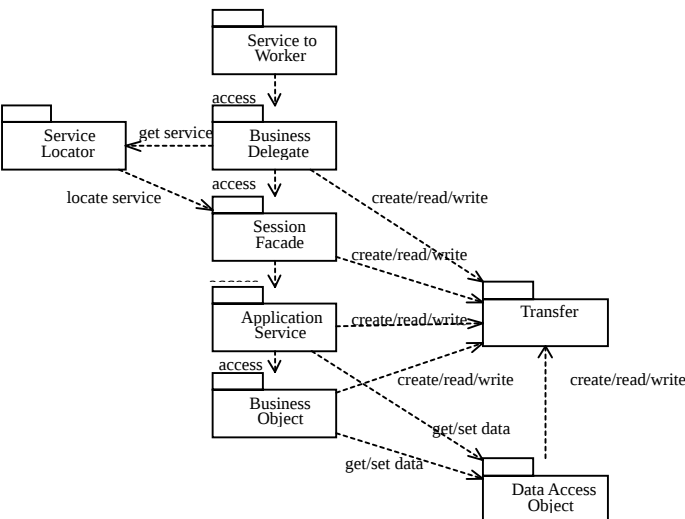


图1 系统多模式集成框架

以下仅介绍与业务层相关的几个主要模式及其在系统中的集成应用。

4.1 Session Façade 模式的引入

在多层 J2EE 应用中,一些服务器端的业务组件(包括 entity Bean、Session Bean 以及服务器端的各种对象)封装了各种业务逻辑和业务数据。如果让客户端直接访问它们,将出现一系列问题:客户端层与业务组件紧密耦合导致二者的直接依赖关系,如果业务组件需要改动,那么客户端的代码也要修改,扩展性很差;客户端与业务组件的频繁交互将导致网络性能的下降;客户端对业务组件的访问缺乏统一的访问策略,这样可能导致对服务器端组件的滥用。解决这个问题的方法是使用

Session Façade 模式,利用 Session Façade 封装业务组件,为客户端提供一个统一、简单的粗粒度访问接口,客户端通过这个接口去访问业务组件。例如,本系统在业务层中为产品查询子模块提供了一个高度集成的产品查询 Session Façade (ProductSearchFaçade), ProductSearchFaçade 由 Session Bean 实现,封装了该子模块所提供的多个业务方法(如按类别查询、按关键字查询等)。这样做的好处有:减少客户端层与业务组件之间的紧密耦合以及客户端层对业务组件的依赖性;集中了安全管理和事务控制;减少客户端与服务器的数据传递次数,减轻网络负载;对客户端隐藏了数据模型的复杂性。

4.2 Application Service 模式的引入

Business Object 是指包含业务逻辑和业务状态的对象(本系统中典型的 Business Object 有客户、订单、产品等),在 J2EE 应用系统中,Business Object 主要用 entity Bean 或 POJO(传统 Java 对象)来实现。为了完成某功能,应用系统可能会调用到多个 Business Object。与单个 Business Object 相关的独立业务逻辑和业务规则一般在 Business Object 内部实现,但还会有一些业务逻辑要涉及到多个 Business Object,或者在 Business Object 所实现的一些功能之上,可能还需要提供一些增值的业务功能。这时如果将这些业务逻辑放在 Business Object 内部实现,无疑会增加 Business Object 之间的耦合,降低它们之间的内聚性。同样如果将这些业务逻辑放在

Session Façade 中,则会使 Session Façade 和它的各个方法变得冗长、臃肿,且这些业务逻辑代码也很难被其它的 Session Façade 重用,这就降低了通用代码的可重用性和可维护性。解决这个问题的方法是使用 Application Service 模式,让 Application Service 来实现这些业务逻辑。这样 Session Façade 实现起来就更简单,更易于维护,因为它只需将业务处理委托给 Application Service,并且由于 Application Service 封装了涉及到多个 Business Object 的业务逻辑,也就提高了业务逻辑的可重用性。如本系统中的 ProductSearchFaçade 通过与 ProductSearchService 这个 Application Service 交互,完成查询产品的功能,其功能类图如图2所示。ProductSearchService 用 POJO 来实现,它通过与几个 Business Object 的交互,完成各种形式的产品查询。类图中的 Business Object 有产品(Product)、产品附加(ProductAddition)和产品目录(Category)。其中 Product 包含每种产品都必不可少的属性;ProductAddition 包含每种产品各自的特有属性;Category 包含产品类别信息。

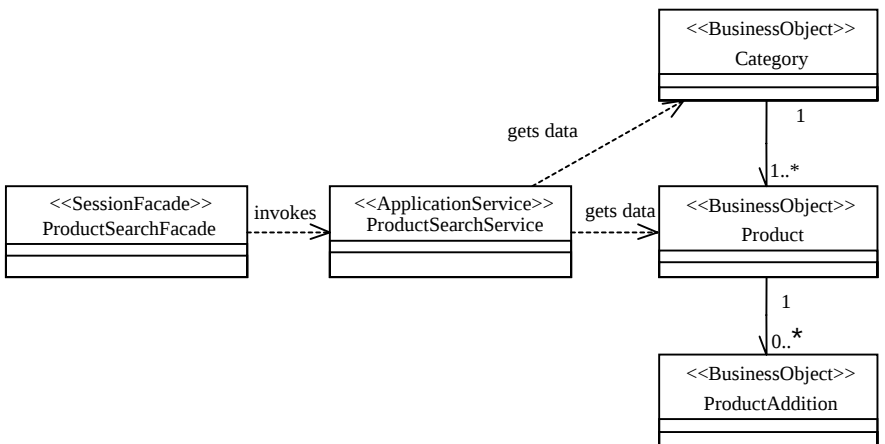


图2 查询产品功能类图

4.3 Business Delegate 模式的引入

虽然已使用了 Session Façade 模式来封装客户端对业务层业务组件的直接访问,但客户端直接与 Session Façade 的交互,还是造成了客户端与业务层的紧密耦合。解决这个问题的方法是使用 Business Delegate 模式,Business Delegate 以 POJO 的形式实现,利用 Business Delegate 作为 Session Façade 的门面,封装了客户端对业务服务的访问。如本系统在表示层中提供了一个产品查询 Business Delegate(Product SearchDelegate),和业务层中对应的 ProductSearchFaçade 保持一对一的关系,ProductSearchDelegate 中提供的方法可以一一映射到 ProductSearchFaçade,客户端通过 ProductSearch Delegate 访问 ProductSearchFaçade。这样做能有效地减低表示层和业务层的耦合度,有助于系统获得更好的可维护性和灵活性。同时,还能够为表示层组件缓存信息,从而可以提高常用服务请求的性能。

4.4 其它业务层模式的引入

系统业务层引入了 Service Locator 模式(系统的 Service Locator 使用 Singleton 模式实现),把对业务组件的查找用一个专门的 Java 类来实现,其他模块共享这个对象,这样就简化了查找,同时也提高了系统的可维护性。另外在实现过程中,还引入了缓冲机制,以保证这些对象只被查找一次,以后直接访问到缓冲区中就可以了,从而避免多次网络开销。

另外业务层还引入了 Transfer Object 模式,利用 Transfer Object 来封装业务数据。当客户端需要获取服务器中业务组件的相关信息时,业务组件先构造出封装所需相关业务数据的 Transfer Object,并将它一次调用传递给客户端,客户端再根据需求从中提取数据;业务组件也可以从用户中取得一个 Transfer Object 格式的数据,并应用这些数据执行一些更新。引入 Transfer Object 模式有利于减少客户端与远程业务组件

通信的次数,降低网络流量,提高应用程序的性能。

4.5 多模式集成应用

本系统在业务层引入并集成了以上多个 J2EE 模式后,其业务调用过程为:

- (1)客户端请求 Business Delegate 提供对底层业务服务的访问;
- (2)Business Delegate 通过 Service Locator 查找并实例化其所对应的 Session Façade,然后将客户端请求的方法调用传递给该 Session Façade;
- (3)Session Façade 调用 Application Service 的业务方法,业务方法执行一些必要的业务处理,包括初始化,调用 Business Object、调用其它 Application Service、调用 Data Access Object,以及执行业务逻辑,从而最终完成客户端的服务请求;
- (4)在上述业务调用过程中,表示层组件(如 Business Delegate)与业务层组件(如 Session Façade)之间、业务层组件(如 ApplicationService、Business Object)与集成层组件(如 Data Access Object)之间均使用 Transfer Object 进行跨层次数据传递。

5 结束语

将多种 J2EE 模式有机地融合在一起应用于 B2B 网上订购系统的设计,能很好地改善和优化系统性能,增强系统的可维护性,扩展性和重用性、提高系统开发效率,大大缩短系统研发周期。

参考文献

- 1 Alur D, Crupi J, Malks D. Core J2EE™ Patterns: Best Practices and Design Strategies(Second Edition)[M]. Prentice Hall, 2003.
- 2 Gamma, Helm R, Johnson R, et al. Design Patterns: Elements of Reusable Object-oriented Software[M]. Addison Wesley, 1994.
- 3 Marinescu F. EJB Design Patterns[M]. John Wiley & Sons, Inc., 2002.
- 4 须文波,黄 涿. EJB 中的 Façade 模式原理[J]. 计算机工程与设计, 2004, 25(12): 2185-2187.

(上接第 276 页)

裁,在找下一个审批人时,将所有的副总裁都列在下拉列表中,可能会造成单据错误提交的情况。为了最大限度地使计算机准确指导用户的行为,增加一个审批人权限的设置,每个人只能审批自己管辖范围内的单据。考虑到集团公司的情况,可以考虑跨部门、公司定义审批人权限。

(3)审批处理方法

对于审批流程的每个节点,目前让其对单据的处理都有一个默认的方法,该默认方法是审批通过或不通过并给出审批意见。在实际应用中,如果哪个节点对单据的审批处理有特殊的要求,可以扩充新的处理方法。这一点很好地体现了 SOFM 的可扩展性和开放性。

2.3 匹配审批流程

审批流程中的一个重要内容是触发条件集合。对象 E 必须满足某规则的触发条件才能按照这个规则流转,流转的过程记录在对象 P 中。任何一张被配置为需要审批的单据都必须有其审批的流程,换句话说,定义的审批流程应该覆盖所有单据可能出现的情况,也就是不能出现一张单据匹配不到它的审批流程,无法流转的情况。为了能达到这个效果,除了使用先精确匹配后模糊匹配的类似路由匹配的算法外,还为单据实体制定了一条触发条件全是“所有”的审批流程,只要是单据,最差的情况也可以匹配到这条审批流程。

2.4 单据的动态审批流转

单据的动态审批流转主要解决的问题就是匹配到合适的

审批流程后按照该审批流程的审批节点找到一个个的审批人,实现单据的审批流转。匹配到审批流程之后,就可以得到审批节点。如果审批节点的实施者为人,那么单据就流转到他手中。如果审批节点的实施者是角色,需要找到对应于该角色的有审批该单据权限的人,如果没有满足条件的人,那么无法进行审批流转。如果有多个满足条件的人,就需要用户来干预审批的流向,也就是让用户决定将单据提交给谁。

3 结束语

SOFM 是笔者在实践过程中总结出来的,具有很好的柔性、普适性和可扩展性。基于 SOFM 的审批应用方案实现了单据审批节点、审批流程的自定义,单据审批流程的匹配以及单据的动态审批流转,可以提高企事业单位审批工作的效率,在 ERP 系统中能得到很好的应用。

参考文献

- 1 Ferraiolo D F, Sandhu R S, Gavrila S, et al. Proposed NIST Standard for Role-base Access Control[J]. ACM Transactions on Information and Systems Security, 2001, 4(3): 224-274.
- 2 丁 柯,金蓓弘,冯玉琳. 事务工作流的建模和分析[J]. 计算机学报, 2003, 26(10): 1304-1311.
- 3 赵 文,胡文蕙,张世琨等. 工作流元模型的研究与应用[J]. 软件学报, 2003, 14(6): 1052-1059.
- 4 范玉顺,吴 澄. 一种提高系统柔性的工作流建模方法研究[J]. 软件学报, 2002, 13(4): 833-839.

