(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 112882689 B
(45)授权公告日 2022.05.20

(21)申请号 202110097920.X
(22)申请日 2021.01.25
(65)同一申请的已公布的文献号
申请公布号 CN 112882689 A
(43)申请公布日 2021.06.01
(73)专利权人 中原银行股份有限公司
地址 450018 河南省郑州市郑东新区商务外环路23号中科金座大厦
(72)发明人 宋松涛 宋秀庆
(74)专利代理机构 北京金信知识产权代理有限公司 11225
专利代理师 夏东栋
(51)Int.Cl.
G06F 8/20(2018.01)

审查员 刘升

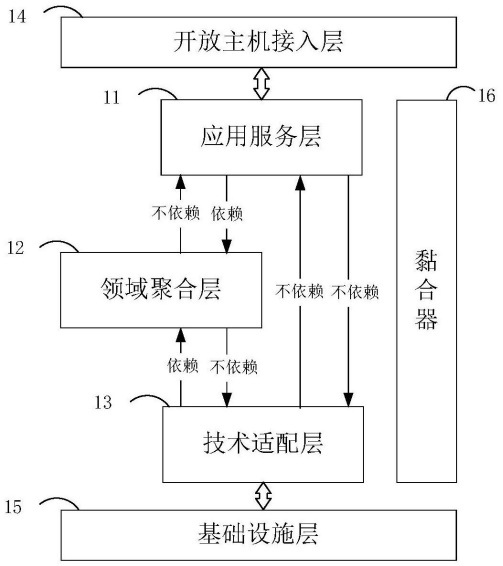
权利要求书1页 说明书6页 附图3页

(54)发明名称

一种基于领域驱动设计的工程架构方法及详细设计框架

(57)摘要

本公开涉及一种基于领域驱动设计的工程架构方法及详细设计框架，该工程架构方法包括：采用基于领域驱动的方法对工程架构进行分层，所述分层包括应用服务层、领域聚合层和技术适配层，其中，应用服务层用于应用服务的实现，包括服务的编排，参数校验，模型组装等，领域聚合层用于领域内的核心业务逻辑的实现，包括领域模型、领域服务等，技术适配层用于所述领域聚合层中的接口或抽象类的实现，以及数据访问相关的实现；对应用服务层、领域聚合层和技术适配层之间的依赖关系进行约束，其中，应用服务层依赖于领域聚合层，技术适配层依赖于领域聚合层，领域聚合层不依赖其他层，以及应用服务层和技术适配层之间相互不依赖。



CN 112882689 B

1.一种基于领域驱动设计的工程架构方法,其特征在于,所述工程架构方法包括:

采用基于领域驱动的方法对工程架构得到分层,所述分层包括应用服务层、领域聚合层和技术适配层,其中,所述应用服务层用于应用服务的实现,至少包括服务的编排、参数校验和模型组装,所述领域聚合层用于领域内的核心业务逻辑的实现,至少包括领域模型和领域服务,所述技术适配层用于所述领域聚合层中的接口或抽象类的实现,以及数据访问相关的实现;

定义层间依赖关系,所述层间依赖关系包括所述应用服务层依赖于所述领域聚合层,所述技术适配层依赖于所述领域聚合层,所述领域聚合层不依赖其他层,以及所述应用服务层和所述技术适配层之间相互不依赖。

2.根据权利要求1所述的工程架构方法,其特征还在于,所述应用服务层通过通信接口与开放主机接入层进行交互,共同提供面向用户的应用服务功能。

3.根据权利要求1所述的工程架构方法,其特征还在于,所述技术适配层通过通信接口与作为平台支撑的基础设施层进行交互,共同提供工程所需的技术支持与服务能力。

4.根据权利要求1所述的工程架构方法,其特征还在于,还具有黏合器,所述黏合器用于实现所述应用服务层、所述领域聚合层和所述技术适配层的层间的数据转换与传递。

5.根据权利要求1所述的工程架构方法,其特征还在于,所述层间依赖关系通过所述应用服务层、所述领域聚合层及所述技术适配层各层代码间的调用关系约束来保证。

6.一种基于领域驱动设计的工程架构的详细设计框架,其特征在于,所述详细设计框架包括:

应用服务层、领域聚合层和技术适配层,其中,所述应用服务层用于应用服务的实现,至少包括服务的编排、参数校验和模型组装,所述领域聚合层用于领域内的核心业务逻辑的实现,至少包括领域模型和领域服务,所述技术适配层用于所述领域聚合层中的接口或抽象类的实现,以及数据访问相关的实现;以及在所述详细设计框架中,

所述应用服务层的目录结构包含API访问器及应用服务中的至少一种;

所述领域聚合层的目录结构包含读模型、领域服务、领域模型、聚合工厂、适配器接口、仓库访问器中的至少一种;

所述技术适配层的目录结构包含数据仓库、读模型、适配器、领域服务中的至少一种。

7.根据权利要求6所述的详细设计框架,其特征还在于,所述详细设计框架中的所述目录结构约束,采用基于Eclipse插件开发的定制化工程目录结构实现。

一种基于领域驱动设计的工程架构方法及详细设计框架

技术领域

[0001] 本公开属于应用软件程序开发领域，涉及软件开发中的软件工程和软件 体系架构设计开发，更具体地，涉及一种基于领域驱动设计的工程架构方法 及详细设计框架。

背景技术

[0002] 领域驱动设计是一种处理高度复杂域的设计思想，试图分离技术实现的 复杂性，围绕业务概念构建领域模型来控制业务的复杂性，以解决软件难以 理解、难以演化等问题。现有的基于领域驱动设计的着重解决前端应用程序 开发中软件复杂性越来越高的问题，但是未能提出通用的、非前端的一般领 域驱动设计工程的分层方法及落地方式，尤其是没有在架构层面约束所有技 术及业务的分离性，因此不能避免一般开发人员的业务与技术混合开发的开 发模式。

发明内容

[0003] 提供了本公开以解决现有技术中存在的上述问题。

[0004] 需要一种基于领域驱动设计的工程架构方法及详细设计框架，能够为基 于领域驱动的方法进行工程设计与开发提供合理的分层方法以及详细的设计 框架，使技术与业务更好地分离，便于业务演进和代码重构，同时提供工程 设计开发效率。

[0005] 根据本公开的第一方案，提供一种基于领域驱动设计的工程架构方法，所述工程架构方法可以包括采用基于领域驱动的方法对工程架构进行分层，所述分层包括应用服务层、领域聚合层和技术适配层，其中，所述应用服务 层用于应用服务的实现，包括服务的编排，参数校验，模型组装等，所述领 域聚合层用于领域内的核心业务逻辑的实现，包括领域模型、领域服务等，所述技术适配层用于所述领域聚合层中的接口或抽象类的实现，以及数据访 问相关的实现。该工程架构方法还包括对所述应用服务层、所述领域聚合层 和所述技术适配层之间的依赖关系进行约束，其中，所述应用服务层依赖于 所述领域聚合层，所述技术适配层依赖于所述领域聚合层，所述领域聚合层 不依赖其他层，以及所述应用服务层和所述技术适配层之间相互不依赖。

[0006] 根据本公开的第二方案，提供一种基于领域驱动设计的工程架构的详细 设计框架，所述详细设计框架包括应用服务层、领域聚合层和技术适配层，所述应用服务层、所述领域聚合层及所述技术适配层按照目录结构约束进行 详细设计。

[0007] 利用根据本公开各个实施例的工程架构方法和工程架构的详细设计框架，可以在分层设计的基础上，通过对层与层之间的依赖关系的约束，有效实现 了应用、业务及技术的分离，更好地聚焦并守护领域模型，利用工程架构的 详细设计框架进行设计开发，可以显著降低开发人员学习领域驱动设计代码 结构的成本，并且可以对工程代码进行强约束，保证了工程代码规范性，提 高工程设计和开发效率，同时为业务演进和代码重构提供了便利。

附图说明

[0008] 在不一定按比例绘制的附图中，相同的附图标记可以在不同的视图中描 述相似的部件。附图大体上通过举例而不是限制的方式示出各种实施例，并 且与说明书以及权利要求书一起用于对所公开的实施例进行说明。在适当的 时候，在所有附图中使用相同的附图标记指代同一或相似的部分。这样的实 施例是例证性的，而并非旨在作为本装置或方法的穷尽或排他实施例。

[0009] 图1示出根据本公开实施例的工程架构方法的分层和各层之间依赖关系 的示意图；

[0010] 图2示出根据本公开实施例的详细设计框架中应用服务层的目录结构示 意图；

[0011] 图3示出根据本公开实施例的详细设计框架中领域聚合层的目录结构示 意图；

[0012] 图4示出根据本公开实施例的详细设计框架中技术适配层的目录结构示 意图。

具体实施方式

[0013] 为使本领域技术人员更好的理解本公开的技术方案，下面结合附图和具 体实施方式对本公开作详细说明。

[0014] 应理解的是，可以对本公开的实施例作出各种修改。因此，上述说明书 不应该视为限制，而仅是作为实施例的范例。本领域的技术人员将想到在本 公开的范围和精神内的其他修改。

[0015] 包含在说明书中并构成说明书的一部分的附图示出了本公开的实施例，并且与上面给出的对本公开的大致描述以及下面给出的对实施例的详细描述 一起用于解释本公开的原理。

[0016] 通过下面参照附图对给定为非限制性实例的实施例的优选形式的描述，本公开的这些和其它特性将会变得显而易见。

[0017] 还应当理解，尽管已经参照一些具体实例对本公开进行了描述，但本领 域技术人员能够确定地实现本公开的很多其它等效形式。

[0018] 当结合附图时，鉴于以下详细说明，本公开的上述和其他方面、特征和 优势将变得更为显而易见。

[0019] 此后参照附图描述本公开的具体实施例；然而，应当理解，所申请的实 施例仅仅是本公开的实例，其可采用多种方式实施。熟知和/或重复的功能和 结构并未详细描述以避免不必要或多余的细节使得本公开模糊不清。因此，本文所申请的具体的结构性和功能性细节并非意在限定，而是仅仅作为权利 要求的基础和代表性基础用于教导本领域技术人员以实质上任意合适的详细 结构多样地使用本公开。

[0020] 本说明书可使用词组“在一种实施例中”、“在另一个实施例中”、“在又 一实施例中”或“在其他实施例中”，其均可指代根据本公开的相同或不同实 施例中的一个或多个。

[0021] 图1示出根据本公开实施例的工程架构方法的分层和各层之间依赖关系 的示意图。在根据本公开实施例的工程架构方法中，采用基于领域驱动的方 法对工程架构进行分层，在分层结构中，如图1所示，可以包括应用服务层 11、领域聚合层12和技术适配层13。

[0022] 应用服务层11用于应用服务的实现，包括服务的编排，参数校验，模型 组装等。

[0023] 其中，服务的编排是对应用服务中单独的组件和应用层的工作进行组织 和协作

的完整的业务流程,涉及例如进程间通信方式、分布式事务等,通过 设计完整的编排框架来支撑。在一些实施例中,进程间通信方式可以采用RPC (Remote Procedure Call)框架,例如利用java.rmi包,基于Java远程方 法协议(Java Remote Method Protocol)来实现。在另一些实施例中,分布式 事务可以通过例如两阶段提交协议(2Phase Commitment Protocol)进行处 理。在其他实施例中,也可以采用面向可执行的流程、面向协作或API网关 等不同框架来实现服务的编排。

[0024] 应用服务中的参数校验,是指在服务的前端及服务后台,对用户的输入 进行验证,以此来保证系统数据的正确性的处理。在一些实施例中,例如可 以采用JSR303(一种JavaBean参数校验的标准)来进行参数校验。

[0025] 应用服务中的模型组装,是指对应用服务层中已创建的多个服务模型进 行装配,已形成新的服务模型的操作。

[0026] 领域聚合层12用于领域内的核心业务逻辑的实现,包括领域模型、领域 服务等。其中,领域聚合层12中的领域模型也称为业务对象模型,是对领域 内的概念类或现实世界中对象的可视化表示,是通过分析问题领域,发掘重 要的业务领域概念,从业务角色内部的观点定义业务用例,并建立业务领域 概念之间的关系之后建立的。在一些实施例中,领域模型为产生预期效果, 还定义了业务人员在业务中承担的角色及其当前职责,以及他们处理和使用 的对象之间具有的静态和动态关系。所有这些业务对象模型组合在一起,可以执行所有的业务用例。

[0027] 在一些实施例中,领域聚合层12中的领域服务表示实现某个领域的任务 的无状态操作,在领域服务中的操作,从领域的角度来看,它是一个整体。“向银行客户推荐金融产品”可以视为领域服务的一个具体示例。

[0028] 技术适配层13用于领域聚合层12中的接口或抽象类的实现,以及数据 访问相关的实现。

[0029] 在根据本实施例的工程架构方法中,以领域聚合层12为核心,定义各层 之间的依赖关系。首先,在一些实施例中,约束应用服务层11依赖于所述领 域聚合层12,而领域聚合层12不依赖于应用服务层11。即:在应用服务层 11具体实现的模块和代码中,可以直接调用领域聚合层12中的模块和代码, 而领域聚合层12中的模块和代码根据本公开实施例所定义的依赖关系,不允 许直接调用应用服务层11中的任何模块和代码。

[0030] 在另一些实施例中,还约束技术适配层13依赖于领域聚合层12,而领 域聚合层12不依赖于应用服务层11,以及应用服务层11与技术适配层13 彼此之间互不依赖。即:在技术适配层13的模块和代码中,允许直接调用领 域聚合层12中的模块和代码,而领域聚合层12中的模块和代码则不允许直 接调用技术适配层13的模块和代码,以及技术适配层13的模块和代码与应 用服务层11的模块和代码彼此不能直接调用。

[0031] 作为示例,当采用maven工程构建一个微服务时,该微服务根据本公开 实施例进行基于领域驱动设计的工程设计,一个父工程即为一个微服务,包 括多个功能模块,每个功能模块由一个子工程实现,各个功能模块分别属于 应用服务层11、领域聚合层12和技术适配层13中的一个。工程设计时,首 先根据业务模型进行业务核心概念的模型化,生成领域模型,依据领域模型 开发的功能模块属于领域聚合层12。可以理解,在根据本公开实施例中的基 于领域驱动设计的工程架构方法进行工程设计时,领域聚合层12是先于其他 层

进行设计开发的,因此,设计时不依赖于其他层,其模块和代码实现也不 允许调用其他应用服务层11或技术适配层13中的任何内容。在其他一些实 施例中,通过约束技术适配层13中用于实现领域聚合层12中的接口或抽象 类的实现,以及数据访问相关的实现的模块或代码的依赖关系,即:只允许 调用领域聚合层12中的模块和代码,而不允许直接调用应用服务层11中的 模块和代码,来实现其如图2所示的依赖关系。同样地,应用服务层11在实现包括服务的编排,参数校验,模型组装等功能时,也只允许调用领域聚合 层12中的模块和代码,而不允许直接调用技术适配层13中的任何内容。由 此,通过模块之间的调用关系所代表的依赖约束,实现本公开实施例中各层 之间的依赖约束。

[0032] 在一些实施例中,应用服务层11还可以通过适当的接口,与开放主机接 入层14进行交互,共同实现应用服务功能。

[0033] 在一些实施例中,技术适配层13还可以通过适当的接口,与作为平台支 撑的基础设施层15进行交互,共同提供工程所需的技术支持与服务能力。

[0034] 在另外一些实施例中,还具有黏合器16,用于实现层与层之间的不同格 式的数据转换及传递。

[0035] 图2至图4分别示出根据本公开实施例的详细设计框架的应用服务层、领域聚合层和技术适配层的目录结构示意图。在根据本公开第二方案的实施 例中,可以采用基于Eclipse插件开发的定制化工程目录结构实现对各层目 录结构的约束。在另外一些实施例中,也可以采用IntelliJ Idea插件实现 上述目录结构。

[0036] 如图2所示,应用服务层目录结构21中示例性地包括子目录:API访问 器(内部)211、API访问器(外部)212及应用服务213。其中,

[0037] API访问器(内部)211是微服务内部访问应用服务的入口目录,其配置 为进一步包括:

[0038] 访问器2111可以配置为:用于存放应用服务入口的实现的包;以及

[0039] 传输对象2112可以配置为:一种在不同设计模式之间传输数据的对象。

[0040] API访问器传输对象(外部)212是微服务外部访问应用服务的入口目录, 其配置为进一步包括:

[0041] 应用服务213可以配置为:不包含领域知识的服务的集合目录,该目录 进一步配置为包括:

[0042] 公共定义2131可以配置为:用于存放工具类及常量定义的包;

[0043] 服务接口2133可以配置为:用于存放服务接口的包;以及

[0044] 服务开发2134可以配置为:存放服务实现的包。

[0045] 图3示出根据本公开实施例的领域聚合层的目录结构示意图。在根据本 公开的基于领域驱动设计的工程架构方法进行工程设计时,领域聚合层12 不依赖于其他各层,先于其他层进行设计开发,是整个工程设计的核心,且 在随后的系统运行中,同时被应用服务层11和技术适配层13依赖,即:其 模块和代码将被应用服务层11和技术适配层13中的代码调用,因此,领域 聚合层的目录结构的规范化,不仅能够提高领域模型建模效率,而且能够使 依赖于该层的应用服务层11和技术适配层13的开发更高效和易于维护。

[0046] 如图3所示,领域聚合层的目录结构31中示例性地包括子目录:读模型 311、领域服务312、公共定义313、适配器314、领域聚合315及示例聚合 316。其中,

[0047] 读模型311用于解决基于领域驱动设计模式下跨多聚合多表关联复杂查 询的方案,其配置为进一步包括:

[0048] 读服务3111可以配置为:提供复杂查询类的服务;以及

[0049] 仓库访问器3112可以配置为:给予读服务3111使用的访问数据模型的 接口;

[0050] 领域服务312可以配置为:包含领域知识的服务,是充实了领域的专有 任务的一系列无状态操作,并执行有意义的业务处理,在一些实施例中,可 以是跨多聚合知识的服务;

[0051] 公共定义313可以配置为:用于存放工具类及常量定义的包;

[0052] 适配器314为领域内部需要访问的一些外部系统或技术实现的适配工具, 这里存放的是接口,具体实现在技术适配层可以配置为进一步包括:

[0053] 服务调度3141可以配置为:实现对另一个微服务的调用;

[0054] 流程调度3142可以配置为:实现对流程编排的调用;

[0055] 消息调度3143可以配置为:实现发送或接收消息;以及

[0056] 分布式调度3144可以配置为:实现定时任务。

[0057] 领域聚合层12可以包含至少一个领域聚合,图3中仅作为示例示出其中 两个,领域聚合A和领域聚合B。多个领域聚合中的每一个,其目录中所包 含的内容相类似,例如,领域聚合A可进一步包括:

[0058] 领域服务A1可以配置为:聚合内的领域服务,侧重于聚合内的领域知识;

[0059] 传输对象A2可以配置为:用于层与层之间的数据传递的对象;

[0060] 常量定义A3可以配置为:用于定义一些常量,或错误码的包;

[0061] 领域模型A4可以配置为:存放领域模型的包,其中,领域模型是一个工 程设计覆盖的业务区域,采用业内通用的术语和语言,根据业务需求解决领 域内的问题。

[0062] 聚合工厂A5可以配置为:实现聚合初始化创建的包;以及

[0063] 仓库访问器A6可以配置为:实现聚合持久化及获取的接口。

[0064] 领域聚合B,作为领域聚合层12中的另一个领域聚合示例,其可以具有 与领域聚合A相同或相近的配置,也可以具有不同的配置。

[0065] 图4示出根据本公开实施例的技术适配层的目录结构示意图。

[0066] 如图4所示,技术适配层的目录结构41中示例性地包括子目录:数据仓 库411、读模型412、适配器413、领域服务414、领域聚合415及示例聚合 416。其中,

[0067] 数据仓库411可以配置为存放访问数据库的相关文件的目录,其进一步 包括访问器(Mybatis)4111,是针对Mybatis访问数据库实现的目录,具有 作为Mybatis中的Mapper接口的Mapper 4111a、Mybatis中的数据模型Model 4111b及Mybatis中的xml文件XML 4111c。

[0068] 读模型412可以配置为:针对领域聚合模块中读模型的接口的具体实现 目录。

[0069] 适配器413可以配置为:针对领域聚合模块中的适配器接口的具体实现 目录,其进一步包括:

[0070] 服务调度4131可以配置为:针对领域聚合模块中的服务调度接口的具体 实现;

[0071] 流程调度4132可以配置为:针对领域聚合模块中的流程调度接口的具体 实现;

[0072] 消息调度4133可以配置为:针对领域聚合模块中的消息调度接口的具体 实现;

[0073] 分布式调度4134可以配置为:针对领域聚合模块中的分布式调度接口的 具体实现。

[0074] 领域服务414可以配置为:针对领域聚合模块中的领域服务中的抽象的 具体实现;

[0075] 领域聚合AA可以配置为:针对领域聚合层12中对应的领域聚合A的接 口实现的目录,可以进一步包括仓库访问器AA6,其配置为针对领域聚合层 12中的特定领域聚合A中的仓库访问器A6的接口的具体实现。

[0076] 领域聚合BB,其配置为领域聚合层12中,领域聚合B的接口实现的目 录。

[0077] 在本实施例中,详细设计框架与根据本公开第一方案的实施例中的分层 方法一致,且对各层的目录结构进行了强约束,这样不仅能够提高代码的规 范性,也可以显著降低开发人员学习领域驱动设计代码结构的成本,提高系 统设计开发的效率。

[0078] 以上描述旨在是说明性的而不是限制性的。例如,上述示例(或其一个 或更多方案)可以彼此组合使用。例如本领域普通技术人员在阅读上述描述 时可以使用其它实施例。另外,在上述具体实施方式中,各种特征可以被分 组在一起以简单化本公开。这不应解释为一种不要求保护的公开的特征对于 任一项权利要求是必要的意图。相反,本发明的主题可以少于特定的公开的 实施例的全部特征。从而,以下权利要求书作为示例或实施例在此并入具体 实施方式中,其中每个权利要求独立地作为单独的实施例。

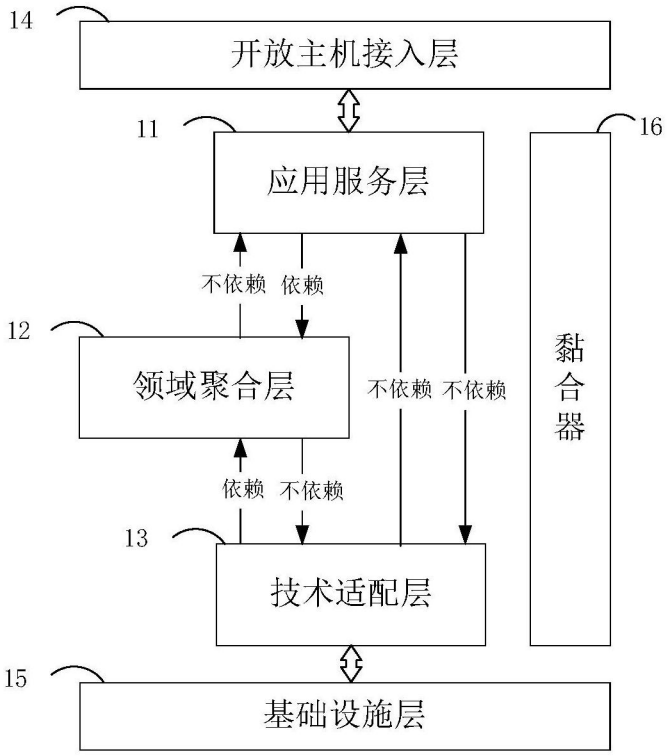

14
开放主机接入层
11
应用服务层
16
不依赖
依赖
12
领域聚合层
不依赖
不依赖
黏合器
依赖
不依赖
13
技术适配层
15
基础设施层

图1


应用服务层目录结构21
API访问器(内部)211
访问器2111
传输对象2112
API访问器(外部)212
访问器2121
传输对象2122
应用服务213
公共定义2131
传输对象2132
服务接口2133
服务开发2134

图2

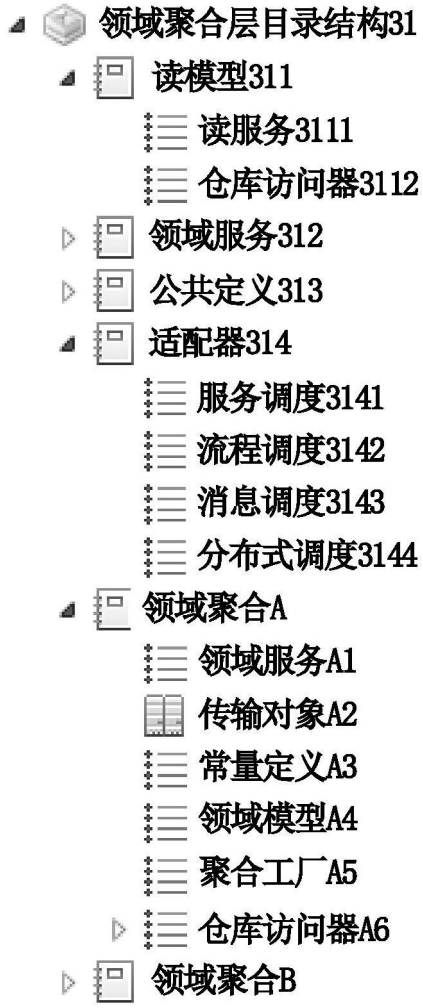

领域聚合层目录结构31
读模型311
读服务3111
仓库访问器3112
领域服务312
公共定义313
适配器314
服务调度3141
流程调度3142
消息调度3143
分布式调度3144
领域聚合A
领域服务A1
传输对象A2
常量定义A3
领域模型A4
聚合工厂A5
仓库访问器A6
领域聚合B

图3

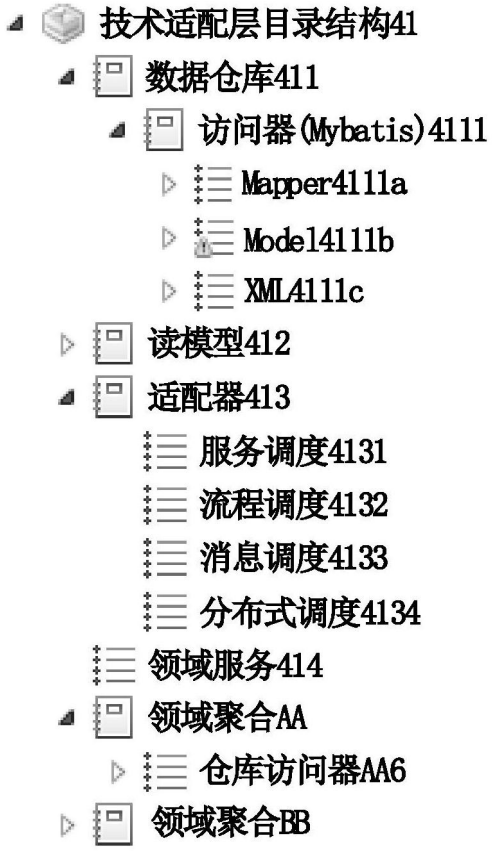

技术适配层目录结构41
数据仓库411
访问器(Mybatis)4111
Mapper4111a
Model4111b
XML4111c
读模型412
适配器413
服务调度4131
流程调度4132
消息调度4133
分布式调度4134
领域服务414
领域聚合AA
仓库访问器AA6
领域聚合BB

图4