

EJB 实现机制的软件设计模式研究

Researching the Implement Mechanism of EJB with Software Design Patterns

李晓华 (河北省城乡规划设计研究院 石家庄 050021)

摘要:从软件设计模式的角度研究了 EJB 的设计与实现机制。其中,用代理设计模式、适配器设计模式、工厂设计模式对 EJB 的基础部分—RMI 进行了研究,用组合设计模式对 EJB 的另一基础部分—JNDI 进行了研究,用观察者设计模式对消息驱动 EJB 的实现机制进行了研究。还论述了接口在 EJB 中的重要作用。

关键词:EJB 软件设计模式 接口 构件

1 引言

设计模式是人们在软件开发过程中积累的软件设计经验,一种软件设计经验是有关某一类软件设计问题的通用解决方案,此解决方案可能是类、类间关系的具体定义,也可能是一种软件设计思想的抽象描述。

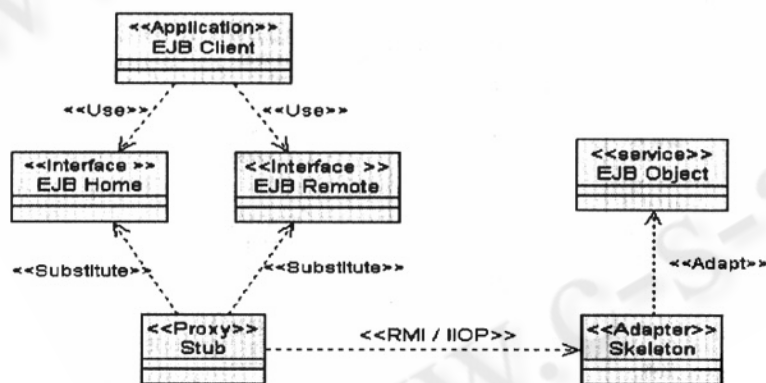


图 1 EJB 的 RMI 的 UML 类分析图

设计模式的最大特点是支持复用,设计模式经常被用于构件的设计与实现,即用设计模式所体现的软件设计思想来定义构件接口、构件所封装的相互关联的类、类间关系,以支持构件的可复用性、访问的安全性、可分布性等。

RMI 和 JNDI 是 EJB 的基础,RMI 和 JNDI 的设计与实现用到了设计模式,设计模式中又大量地用到了接口。

2 RMI 的软件设计模式研究

RMI 是 EJB 的基础,而 EJB 是 J2EE 的核心部分。RMI 过程如图 1 所示。图 1 中 Stub 的作用有三点:

(1) 作为远程被调用的 EJB 对象的代理,原因是 Stub 置换了 EJBHome 接口对象和 EJBRemote 接口对象,而 EJBHome 接口对象和 EJBRemote 接口对象是远程被调用的 EJB 对象的本地代理。

(2) 建立与远程 EJB 对象的握手连接。

(3) 序列化要传送到远程 EJB 服务器的客户端数据和反序列化从远程 EJB 服务器传送到客户端的数据。序列化的含义就是把要传送的数据编码成适于在网上传输的网络字节流,如果要传送的数据是对象,原则是只传对象属性,不传对象方法。如果对象

属性还是对象,按此原则分解下去,直到属性均为简单的 Java 数据类型,如,字符串、整型、布尔型、实数型。已分解和编码的数据通过网络传送到接收方后,接收方对所接收到的编码数据进行解码,即把网络字节流变换成接收方系统的数据表示格式,然后,在接收方重构发送方想通过网络要发送的对象,这样,接收方重构的对象就是发送方要发送对象的一个远程副本。接收方可以使用这个远程副本对象与发送方协作实现分布

式计算。

图 1 中 Skeleton 的作用有三点:

(1) 根据所收到的 EJB 客户端的请求信息,定位响应 EJB 客户端请求的 EJB 服务对象。

(2) 序列化要发送到 EJB 客户端的请求响应结果,反序列化来自 EJB 客户端的请求信息。

(3) 适配器,因为 EJB 客户端欲远程调用的方法可能与远程 EJB 服务对象所提供的方法不一致,这时 Skeleton 就负责把 EJB 客户端的调用适配成 EJB 服务对象所提供的具体服务。

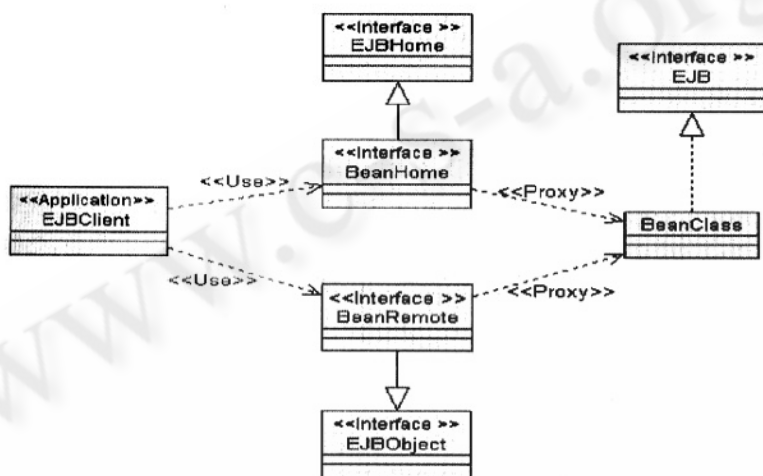


图 2 EJB 的代理模式的 UML 类分析图

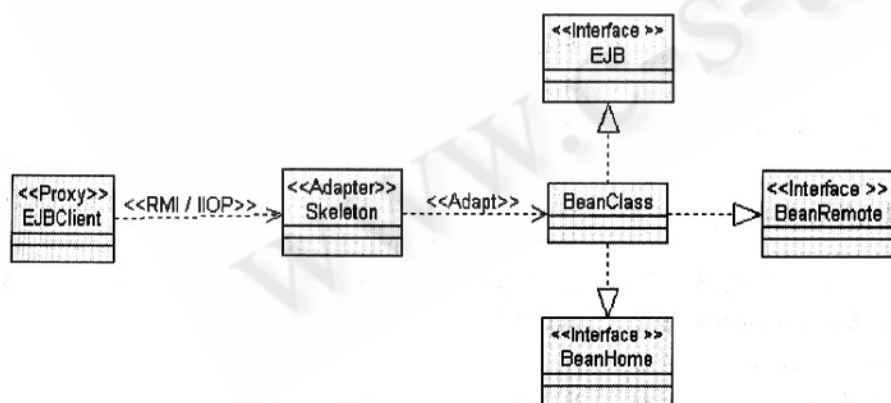


图 3 EJB 的适配器模式的 UML 类分析图

在实体 EJB 和会话 EJB 的设计和实现中用到了“代理”、“适配器”、“工厂”设计模式,用于分析的类示

意图如图 2 至图 4 所示。

在图 2 中, EJB 的 BeanHome 接口和 EJB 的 BeanRemote 接口是 EJB 的 BeanClass 实现的远程代理, EJB 客户端在本地通过 BeanHome 接口和 BeanRemote 接口访问 EJB 的远程的 BeanClass 实现。

在图 3 中, Skeleton 负责把 EJB 客户端的调用适配成 EJB 服务对象所提供的服务——实体 EJB 和会话 EJB 实例的创建逻辑、实体 EJB 实例的查询逻辑和 EJB 所封装的企业逻辑(企业的业务逻辑和企业的组织逻辑)。

在图 4 中, EJB 的 BeanHome 接口是生产 EJB 的 BeanRemote 接口对象和 EJB 的 BeanClass 实现对象的工厂。EJB 客户端通过 JNDI 机制获得 EJB 的 BeanHome 接口对象, EJB 客户端使用 BeanHome 接口对象的生产逻辑(实体 EJB 实例的创建和查询逻辑,会话 EJB 实例的创建逻辑)在远程 EJB 服务器上生成有效的 EJB 的 BeanClass 服务实现对象和对应的 BeanRemote 接口对象,并把 BeanRemote 接口对象通过网络传送到 EJB 客户端。EJB 客户端在本地通过调用 BeanRemote 接口对象中的企业逻辑代理来调用远程 EJB 服务器上 BeanRemote 接口对象所对应的 EJB 的 BeanClass 实现对象中的真正的企业逻辑,从而实现分布式计算。

3 JNDI 的软件设计模式研究

JNDI 与 RMI 一起构成了 EJB 的基础, JNDI 是一个公共资源库, 可以供 J2EE 域中的 Web 构件和 EJB 构件使用。向 JNDI 注册资源有四种方式:

(1) 缺省注册,即在 Application Server 启动过

程中默认创建一些资源并向 JNDI 注册。

(2) 使用 Application Server 控制台创建资源并向

JNDI 注册资源。

(3) 部署 EJB 的过程中,向 JNDI 注册 EJB。

(4) J2EE 程序在运行过程中动态创建资源并向 JNDI 注册资源。(1)、(2)、(3) 中创建、注册的资源是静态资源,即资源被创建并注册后,只要 Application Server 启动了,资源就有效;(4) 中创建和注册的资源是动态资源,即只有在创建并注册资源的应用程序运行时资源才有效。

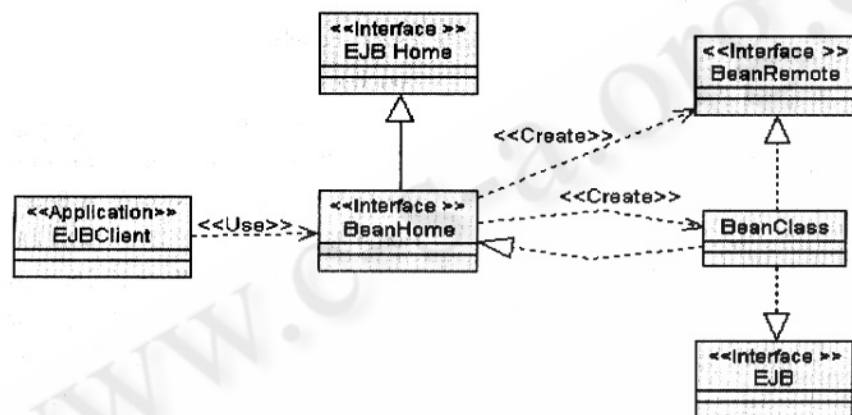


图 4 EJB 的工厂模式的 UML 类分析图

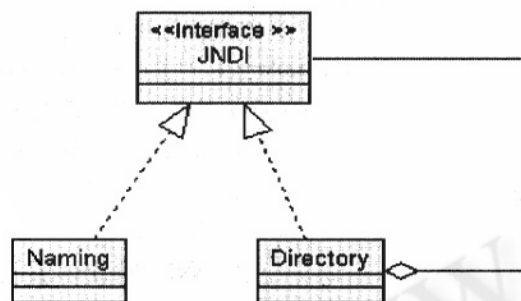


图 5 JNDI 的组合设计模式的 UML 类分析图

JNDI 负责公共资源的注册、查找和管理工作。可向 JNDI 注册的内容很丰富,如: EJB 组件, JMS 连接工厂、数据库连接池、事务管理对象等。J2EE 程序可用资源名称从 JNDI 中查找所要使用的资源对象,找到后就可使用资源,待程序不再使用资源时,就把资源对象归还给 JNDI,以供其他 J2EE 程序使用 JNDI 的共享资源, JNDI 的有效作用范围是整个 J2EE 域。

JNDI 的设计与实现应用到了组合设计模式, JNDI

用组合设计模式的思想来组织和管理 J2EE 公共资源,用于分析的 UML 类示意图如图 5 所示。

JNDI 的组合设计模式的对象示意图更直观一些, JNDI 以树型结构来组织和管理向它注册的资源对象,由于篇幅所限,就不画出 JNDI 的组合设计模式的对象示意图了。

4 消息驱动 EJB 的软件设计模式研究

消息驱动 EJB 的工作原理

用到了观察者设计模式,其中, JMS 服务器充当了主题中心的角色,是一个消息中转站,用于从消息生产者接收消息,储存消息,适当时候再把消息发送给对消息感兴趣的消息消费者。消息驱动 EJB 充当了观察者(消息消费者)的角色,他向 JMS 注册自己所感兴趣的消息,并异步接收和处理来自 JMS 的消息。消息生产者产生并向 JMS 发送消息。消息生产者和消息驱动 EJB 的关系是很松散的,

他们两者彼此不可见,即消息生产者产生的消息由谁来处理,消息生产者是不知道的;消息消费者——消息驱动 EJB 所处理的消息是由谁产生的,消息驱动 EJB 也不知道。消息的产生、发送、存储、接收、处理完全是按异步方式进行的,而且,消息还可以在 JMS Server 端永久存储,如果因为某种原因, JMS Server 中的消息还没来得及发送出去,系统就关闭或停止了,那么当系统重新恢复正常运行后, JMS Server 可以把以前还没来得及发送出去的消息从外存中提取出来,发送给消息消费者,这在某种程度上提高了 J2EE 程序的健壮性。消息驱动 EJB 用到的观察者设计模式的 UML 类示意图如图 6 所示。

5 接口在 EJB 中的作用探讨

(1) J2EE 程序在访问 EJB 构件时,直接访问的是 EJB 构件的 Home 接口或 Remote 接口,而不是 EJB 构件的具体实现, EJB 构件的 Home 接口或 Remote 接口充当了 EJB 构件实现的代理角色。只要 EJB 构件

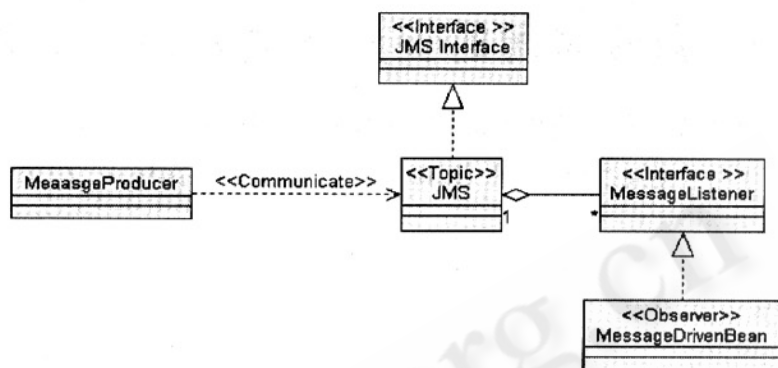


图 6 消息驱动 EJB 的观察者设计模式的 UML 类分析图

的 Home 接口和 Remote 接口保持不变,不论 EJB 构件的具体实现如何变化,访问 EJB 构件的 J2EE 应用程序不用修改就可以访问 EJB 构件的新的实现功能了,这使得 J2EE 应用程序的扩展、维护变得很灵活和高效。

(2) EJB 构件的接口部分和实现部分可以分布于不同节点的 JVM 中,J2EE 客户端程序直接访问的是 EJB 构件的接口,借助于 RMI/IIOP 机制而间接地访问远程的 EJB 构件实现,从而实现了分布式计算。

(3) EJB 构件的接口是 EJB 构件与客户端(EJB 构件的调用者)之间的一份合同或契约,开发 J2EE 应用程序时,在源代码级只做到访问 EJB 构件的接口就可以了,而不用考虑构件内部具体是如何实现的,J2EE 应用程序只是在运行时才绑定到构件的具体实现,这样就使开发时的绑定推迟到了运行时的绑定,从而使得 J2EE 应用程序与 EJB 构件的开发工作可以分开独立完成,或并行进行开发,这在一定程度上缓解了程序的开发难度,提高了开发效率。

(4) 一个 J2EE 系统中的构件可以应用于其它需要相同构件接口的 J2EE 系统中,或者一个 J2EE 系统中的构件可以被实现了相同接口的其它构件置换。

6 结束语

接口是设计模式的基础,设计模式是 RMI 和 JNDI 的基础,RMI 和 JNDI 是 EJB 的基础,EJB 是 J2EE 的核心部分。借助于软件设计模式的思想 and 理念,来解剖、研究 EJB 的设计和实现机制,可以对 EJB 的软件架构有更清晰、更透彻的理解,从而可以用 EJB 技术构建出能更好地支持复用性、安全性、分布性的系统。

参考文献

- 1 Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. DesignPatterns [M], 北京 机械工业出版社,2002 年.139 页-141 页。
- 2 Markus Volter, Alexander Schmid, Eberhard Wolff. Server Component Patterns - Component Infrastructures Illustrated with EJB [M], 北京 机械工业出版社,2004 年.134-150 页。
- 3 刘特,J2EE EJB 应用编程实例 [M], 北京 清华大学出版社,2003 年.130 页-137 页。
- 4 Jim Arlow, Ila Neustadt. UML and the Unified Process—Practical Object Oriented Analysis & Design [M], 北京 机械工业出版社,2003 年,185 页-188 页。