J2EE

应用框架设计与项目开发

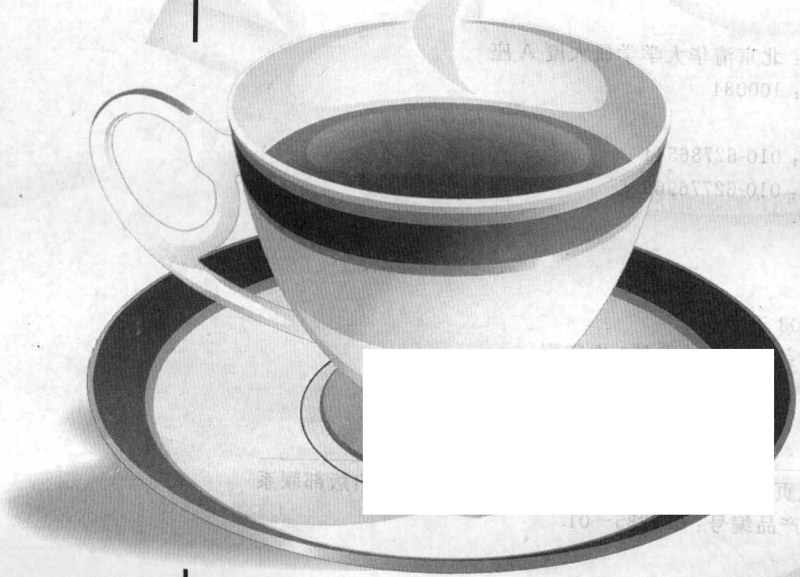
余浩东 著

- 打造属于国人自己的应用框架
- 一本不可多得的J2EE开发经验指南
- 深入揭示J2EE项目开发的全过程
- 凝聚作者在多个行业的数年开发经验

清华大学出版社

J2EE
应用框架设计与项目开发

余浩东 著



- 打造属于国人自己的应用框架
- 一本不可多得的J2EE开发经验指南
- 深入揭示J2EE项目开发的全过程
- 凝聚作者在多个行业的数年开发经验

清华大学出版社
北京

内 容 简 介

本书详细讲解了 J2EE 应用开发框架的实现过程，揭示了框架开发的原理和方法，并结合实际应用分析了相关的技术疑难。同时，本书还讲解了基于框架建构具体的 J2EE 软件项目的方法。本书内容全部基于作者开发的一个应用框架，作者在其网站提供该应用框架的全部源代码。

本书分两大部分。第一部分为前 6 章，详解一个 J2EE 应用开发框架的实现过程。作者将带领读者层层揭开框架设计技术的神秘面纱，深入剖析 J2EE 框架设计核心原理，以及涉及的技术难点。第二部分为第 7 章至第 10 章，基于第一部分实现的框架，讨论具体的 J2EE 软件项目的构建过程，分析各个开发阶段的设计方法、思路及面临的问题，向读者提供了一个真正的、可参考的实战项目案例。

本书凝聚了作者在多个行业的数年开发经验，对技术问题有独到的分析和见解，十分适合于对 J2EE 框架设计核心原理及技术感兴趣的读者，同时也是从事 J2EE 软件项目开发的人员必读的一部不可多得的编程参考指南。

图书在版编目（CIP）数据

J2EE 应用框架设计与项目开发 / 余浩东著. —北京：清华大学出版社，2008.2
ISBN 978-7-302-16863-8

Ⅰ. J…　Ⅱ. 余…　Ⅲ. JAVA 语言－程序设计　Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2008）第 006805 号

责任编辑：战晓雪
责任校对：张　剑
责任印制：王秀菊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　**邮　　编**：100084
c－service@tup.tsinghua.edu.cn
社 总 机：010-62770175　**邮购热线**：010-62786544
投稿咨询：010-62772015　**客户服务**：010-62776969
印 刷 者：北京鑫丰华彩印有限公司
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：203×260　**印　张**：27　**字　数**：703 千字
版　　次：2008 年 2 月第 1 版　　**印　　次**：2008 年 2 月第1 次印刷
印　　数：1～4000
定　　价：49.00 元

前言

计算机技术发展日新月异，在当今可用于企业级软件开发的平台中，J2EE 是首选。而从技术的成熟度和可跨平台特性来看，笔者甚至认为 J2EE 是目前开发企业软件系统的最佳平台。

然而，现实中很多应用 J2EE 技术开发的项目效果却不尽人意，其投资与回报往往令人失望。不是系统运行速度太慢，可靠性差，就是系统结构过于复杂，难于维护和扩展；面对业务需求的变更，代码无所适从，开发周期越拖越长等。

问题出在什么地方呢？当然，决定一个 J2EE 项目失败的原因是多方面的。抛开项目管理不谈，与其把失败的原因归咎于 J2EE，认为 J2EE 技术不成熟或存在技术缺陷，倒不如说是因为 J2EE 没有被正确地理解和使用。不正确地理解往往表现为对 J2EE 技术规范过于痴迷，陷入了一个个纯技术的陷阱；不正确地使用往往表现为忽视各种现实问题的特点、体系结构和解决方法而盲目地去实现。

现在市面上有很多关于 J2EE 技术的书籍，大多数都是强调 J2EE 的各种技术规范，而围绕和关注人们使用这些技术规范来解决各种实际问题的图书则甚少。如何在实践中有效地使用 J2EE，快速开发出高效、可靠、可维护、可扩展而又可灵活部署的 J2EE 软件系统呢？这正是本书所要探讨的，也是本书所要实现的目标。

本书分两大部分。第一部分实现一个 J2EE 应用开发框架，它涵盖了 J2EE 体系结构的表示层、业务层和数据持久层。另外，还包括一个应用服务器程序开发框架，以满足信息系统开发中常见的自己构建服务程序的需求。框架解决 J2EE 常见的使用问题，避免在 J2EE 项目中常犯的低级而高价的错误，封装 J2EE 各种服务和复杂的 API，以注重实际经验和实际效果为设计理念编写代码，实现代码最大的重用，以便读者能够根据时间和预算，快速地构造出自己的软件系统。

关于 J2EE 框架设计，笔者并没有像很多书籍一样，对当前流行的“Struts+Spring+Hibernate”组合模式进行跟风。其原因有：首先，纯粹的功能介绍说明，对提高读者的 J2EE 应用水平作用不大，高层次的开发人员应该了解框架设计内核原理而不是仅仅停留在功能使用上；其次，所谓的 SSH 经典开发组合，只能算是 J2EE 应用开发领域的一个解决方案，面对大型复杂企业应用需求，往往表现得力不从心。对 SSH 的盲目而过度的吹捧，仿佛开发 J2EE 应用就是开发 SSH 应用，这种观点必须加以矫正。还有对 Spring 带来的“轻量”风气也必须有一个本质上的认知，因为 J2EE 应用开发领域所涉及的技术和问题的复杂度并不是想象的那样简单；最后，笔者认为现行的框架不能满足所有开发的需求。任何框架都不是万能的，都有其各自设计关注的领域而导致不能面面俱到地满足企业开发的各种要求。所以，需求的多样性和设计理念的差异性必然导致框架产品的多样性：如 web 表示层框架有 struts、webwork、jsf、echo 等，业务层框架有 spring、realmethods、keel 等，持久层框架有 hibernate、jpa、ibatis 等。

显然，作为开发者只有理解和掌握了 J2EE 框架内在原理，能自由地按需设计自己的框架才能以不变应万变，才能使自己立于不败之地。即使不开发自己的框架，了解 J2EE 框架设计核心

原理对维护和扩展系统功能也是大有好处的。由于 J2EE 的规范性和标准性，事实上当前所有的 J2EE 框架在原理上都可以说是相似或者是共通的。任何深层次的开发都不能只简单停留在了解所谓流行框架使用的层面上，而忽视其核心技术原理。经验表明，不了解一个框架的核心原理而盲目地使用它，往往会使自己很被动。

笔者根据自己多年在 J2EE 领域的工作和实践经验，从简单实用出发来探讨 J2EE 应用框架设计和实现，通过本书，可以揭开框架设计技术的神秘面纱，改变普通开发人员对国外框架顶礼膜拜的局面，让读者具备开发自己的框架的能力。

本书的第二部分，基于构建的框架来讨论如何构建具体的 J2EE 软件项目。在此部分，笔者十分注重项目开发实战性，与当前蜻蜓点水般简述项目开发的书籍不同，本书从需求开发、分析设计、代码构建等软件开发阶段去详细论述，尝试探讨开发一个具体应用系统所经历的各个阶段的设计方法、思路及其所面临的各种问题，以期向读者呈现出一个真正的、可参考的项目案例。

本书的读者对象是掌握 J2EE 基础知识而且对开发 J2EE 软件项目感兴趣的开发人员。笔者采取通俗易懂的日常语言而非枯燥无味的技术性语言，使得本书更为平易近人。每论述完一个技术话题，笔者一般都会把自己在同类问题上的亲身经历和经验描述出来与读者分享，使得本书从另一个角度来说也是一本 J2EE 编程设计经验的参考指南。

虽然我们已经尽了最大努力来减少本书文字和代码错误的出现，但是人无完人，出现错误也在所难免。为此，我们建立了www.beetlesoft.net网站，如读者发现本书错误之处或者有任何疑问，都可以登录网站与笔者交流互动。同时，本书的示例代码都可以通过该网站下载。我们也会通过该网站发布最新的代码版本。

本书从编写到出版，得到了很多亲朋好友的支持和帮助，藉此机会致谢如下：

感谢我可爱的妻子惠佳，在整个漫长而枯燥的写作过程中给予我爱的支持和鼓励。

感谢我父亲、母亲和岳父、岳母，虽然他们很多时候不明白我在写什么，但是，长辈们不时的问候和关心，激发了我以很大的动力和责任去完成这本书。

感谢已移民美国的 Sabrina，记得我当初把本书的写作创意说给她听时，立即得到她的肯定和鼓励。她是本书的第一位读者和前两章的审稿者。

感谢刘畅、何保林、杨珂等同事，和他们的交流讨论，激发了我不少的灵感和创意。

感谢清华大学出版社为本书出版而辛勤付出的所有工作人员，特别是战晓雷编辑为此付出大量精力。

最后，感谢购买本书的读者，你们的支持是对笔者完成本书所付出的辛苦劳动的最大肯定。衷心希望本书能给您开发 J2EE 应用带来帮助，谢谢！

余浩东

2007 年 9 月

目录

第 1 章　J2EE 体系结构

J2EE，即 Java2 企业版，是美国 Sun 公司为了开发安全、可靠、可扩展、高效可用的企业级应用系统所提出的一套现行技术规范。它定义了开发企业级应用的各种服务，第三方开发商可以根据此定义，开发能够部署和运行 J2EE 兼容的应用程序的应用服务器。比如，大家熟悉的 WebLogic、JBoss 等。关于 J2EE 规范的最新定义和标准，请浏览网址：http://java.sun.com/j2ee/ 了解详细的情况。

1.1　体系结构

我们现在来回顾一下 J2EE 的体系结构。见图 1.1：

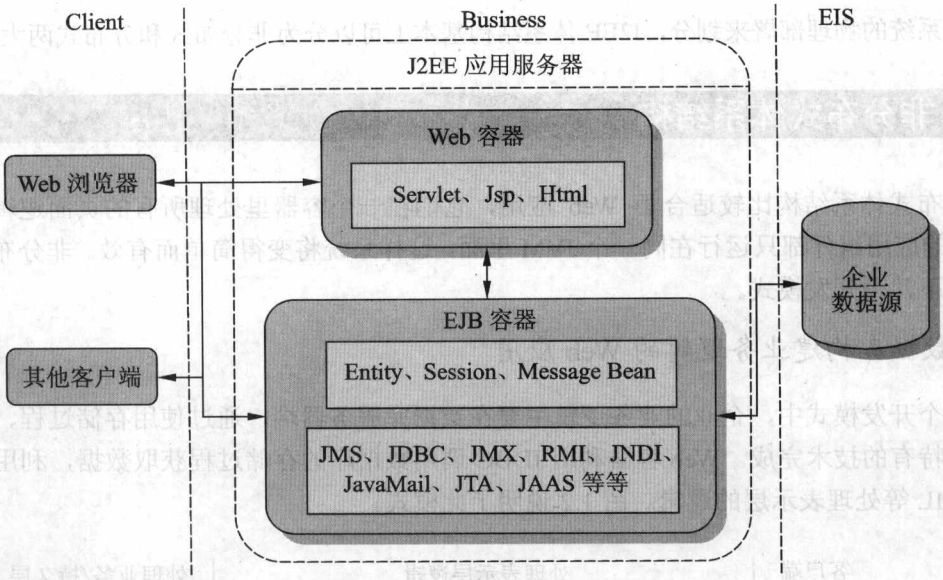


图 1.1

由图 1.1 可知，J2EE 体系结构分 3 个主要层。

- Client 层

即用户的使用界面，它与 Business 层通信。客户端最常用的是 Web 浏览器，也可以是 Java 语言或者其他编程语言编写的传统 UI 应用程序。

- Business 层

即逻辑处理层，主要由 J2EE 应用服务器组成，包含 Web 容器和 EJB 容器。其中，Web 容器

处理表示层逻辑，响应客户端的请求，和 EJB 容器通信，来构建用户界面。主要应用的技术有：Servlet 服务小程序，Jsp 页面和 HTML。EJB 容器负责处理业务逻辑，与 EIS 层的数据源通信，完成数据的存储。主要技术有：EJB（包括实体 Bean，会话 Bean 和消息 Bean）、JTA、JMS、JDBC 和 JNDI 等等。

- EIS 企业信息系统层

由 J2EE 完成业务逻辑所必须访问的企业资源所组成，所以称作数据源。这些资源主要包括数据库管理系统（DBMS）、文件数据、邮件服务和企业原有遗留下来的应用资源等等。虽然 EIS 不受 J2EE 应用服务器控制，但是，J2EE 提供了强有力的访问 EIS 层的接口。比如：访问 DBMS 的 JDBC API，访问目录服务的 JNDI，以及连接其他 EIS 系统的 JCA。J2EE 应用服务器负责建立访问 EIS 资源的连接池，事务管理以及保证 J2EE 应用不会危及 EIS 系统的安全等等。

上面基本上就是 Sun 公司描绘的 J2EE 技术蓝图。在这套技术体系下，开发企业信息系统，就是所谓 J2EE 项目了。下面再讨论一下 J2EE 体系结构的类型。

1.2 体系结构的划分

按照系统的物理部署来划分，J2EE 体系结构基本上可以分为非分布式和分布式两大体系结构。

1.2.1 非分布式体系结构

非分布式体系结构比较适合于 Web 应用，它们在一个容器里处理所有的页面逻辑和业务逻辑，所有的应用组件都只运行在同一个 JVM 里面，这样系统将变得简单而有效。非分布式结构下一般有以下几个开发模式。

1. 数据库构建业务逻辑的 Web 应用

在这个开发模式中，企业的业务逻辑主要在数据库服务器端，通过使用存储过程、触发器等数据库所特有的技术完成。Web 容器利用 JDBC 调用数据库的存储过程获取数据，利用 Servlet、jsp、HTML 等处理表示层的逻辑。图 1.2 说明了此模式。

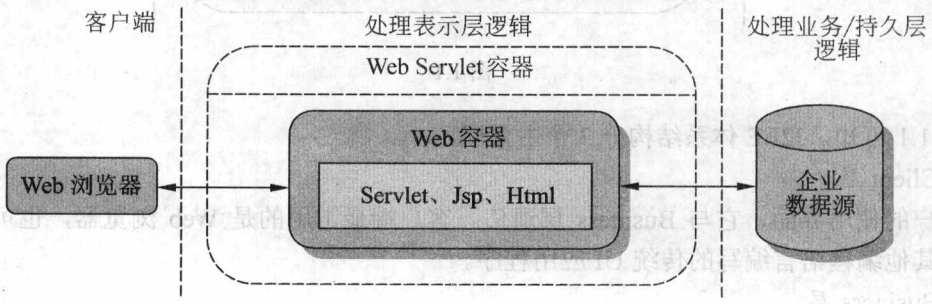


图 1.2

1）优点

（1）简单性。结构简单清晰，页面只需显示数据，代码简单，业务的事务交给数据库管理。如果对某个数据库语言熟悉的话，比采用其他模式开发周期短。

（2）速度。由于业务逻辑在数据库服务器端计算，Web 容器只是负责显示数据，系统开销很小，速度响应快。

（3）测试容易。业务逻辑用数据库存储过程编写，容易开发和调试，无须 Web 层，就可以对业务接口进行测试。

（4）系统部署简单。Web 容器只有表示层逻辑，很容易按要求部署。

2）缺点

（1）整个应用对数据库的功能和容量要求高。需要支持存储过程、触发器等技术。数据库配置要求大，适合 Oracle、SQL Server、DB2 和 Sybase 等大型商用数据库。而对于某些不支持存储过程的数据库，如 MySQL3.25，根本就不能应用这个开发模式。

（2）维护不易。业务用数据库的特有技术实现；表示层用 Java 实现；对开发人员技术要求高；维护两套不同技术实现不是很容易。

（3）数据库程序代码可能会很复杂。由于一般的数据库编程语言都不是面向对象，而是过程语言。而且，一般存储过程在重用的时候（比如，在一个存储过程里调用另外一个存储过程），数据的一致性和完整性没有保障，容易出现并发同步问题。当然，有些数据库编程语言支持面向对象，比如，Oracle 8i 以后的版本支持 Java，可以用 Java 语言编写存储过程等，给我们多提供了一个选择。

（4）系统扩展性差，数据库性能要求高，性能调整代价大。在这个开发模式中，数据库不单单负责存储数据，而且还负责处理大量的业务逻辑。这样对数据库的性能要求甚高。当系统用户数超过原系统设计目标，并发大的时候，数据库系统性能会急速下降。业务逻辑不能从数据库抽离出来，自由部署，系统变得很难扩展。这时候就只能考虑增加系统物理的硬件或使用数据库系统集群解决方案了，代价大。

不少技术经理对这个开发模式不屑一顾。认为这根本就不属于 J2EE 开发，没有技术含量可言。但是笔者发现早期传统的行业，如银行、证券、电信、保险等，应用这种模式开发的系统比比皆是。这种模式的优缺点上面已经讨论了。由于它应用了 J2EE 的 JDBC、Servlet 和 Jsp 等技术，笔者依然认为它是属于 J2EE 开发的范畴。在很多时候，这种模式是快速而高效的，不能因为“纯”技术的原因而否定它。技术应以项目为本。应该以项目为中心，根据其需求和容量等特点来决定应用什么样的开发模式，而不是反其道而行之。

2. 具有处理业务逻辑组件的 Web 应用

在这个开发模式中，Web 容器不仅负责表示层，还要处理业务逻辑。利用普通的 Java Bean 调用 J2EE 提供各种服务的 API 来访问企业数据源，完成各种业务逻辑的开发。虽然整个应用的 Web 表示层和中间业务层都运行在同一个 JVM 中，但是，设计开发中在逻辑上使两层保持分离是很重要的，尽量不要在表示层的 UI 组件与中间层的业务组件之间的责任归属问题上模糊不清。

这也是构建一个简单、可缩放系统的基础。在本书的系统开发部分，我们会详细讨论项目系统的逻辑层的划分问题。

1）优点

（1）结构简单。从图 1.3 可见，开发模式一目了然。不需 EJB 容器支持的 J2EE 应用服务器（如：Tomcat、Resin 等）就可以很好地完成工作。但是，代码开发不一定就变得简单，如果事务管理和线程化问题要求高的话，代码就会变得很复杂，这时候，使用 EJB 反而更为简单。

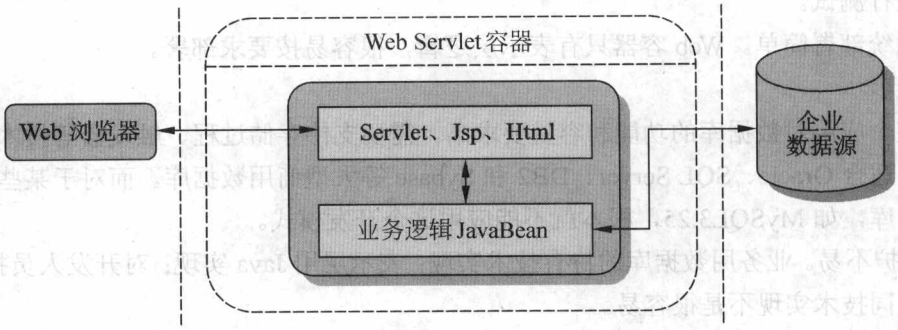


图 1.3

（2）响应速度快。整个应用运行在同一个 JVM 内，没有远程调用，系统开销很小，速度快。

（3）调试和测试容易。没有使用 EJB 容器，代码本地化，无须部署就可调试。同时，如果设计合理的话，不需要 Web 接口就很容易进行业务组件的调试和测试。

（4）系统部署容易灵活。整个应用在同一个 JVM 中，部署在一台物理机器上；当系统用户量增大的时候，也很容易利用 Web 应用服务器的负载均衡技术，在多台物理机器上部署同一个应用，以满足系统容量扩大的需求。

2）缺点

（1）对外只有一个 Web 的接口，除非增加一个 Web Service，否则很难支持其他独立 GUI 客户端。

（2）因为整个应用只能运行在一个 JVM，虽然提高了组件之间的访问性能，但是当一台物理机器的 JVM 不能满足并发访问的时候，我们无法将这些组件自由地部署到其他物理机器上。

（3）由于该开发模式没有用到 EJB 容器的事务支持，我们需要自己在代码里创建和管理事务，增加了开发的难度。

（4）应用服务器不支持 EJB 容器，意味着丢失了 EJB 容器带来的各种方便好用的服务，对于开发复杂的企业应用往往显得力不从心，对项目的开发速度也存在影响。

（5）由于中间业务层不能剥离出来，部署在内部网络之中，这样对业务层数据安全要求高的应用，就不是很适合了。入侵了 Web 层就等于入侵了中间业务层，因为它们绑定在同一 JVM 中。

这也可能是 J2EE 中最常用、结构最简单的开发模式了，也就是现在很多人推崇的所谓“轻量级”企业开发模式。应用很广泛，网上很多著名的开源软件都是使用这个模式开发的。比如 Jive 论坛、OpenCMS 等等。笔者认为，对于系统投资和容量都不大，系统部署和安全性要求都不高的企业应用，使用这个模式还是比较适合的。

3. 使用本地接口 EJB 的 Web 应用

在这个开发模式中，Web 层和中间业务层基本上和上面讨论的“**具有处理业务逻辑组件的 Web 应用**”相同。而与其不同之外在于：本模式的中间业务层的业务接口使用本地 EJB 实现，而不是采用普通 JavaBean。与远程调用 EJB 不同，本地 EJB 无须暴露一个远程接口，而仅仅是业务接口实现的一个选择罢了。这样既能利用 EJB 容器现成的各种服务，而又不至于把代码复杂化或者把应用变成了分布式。见图 1.4。

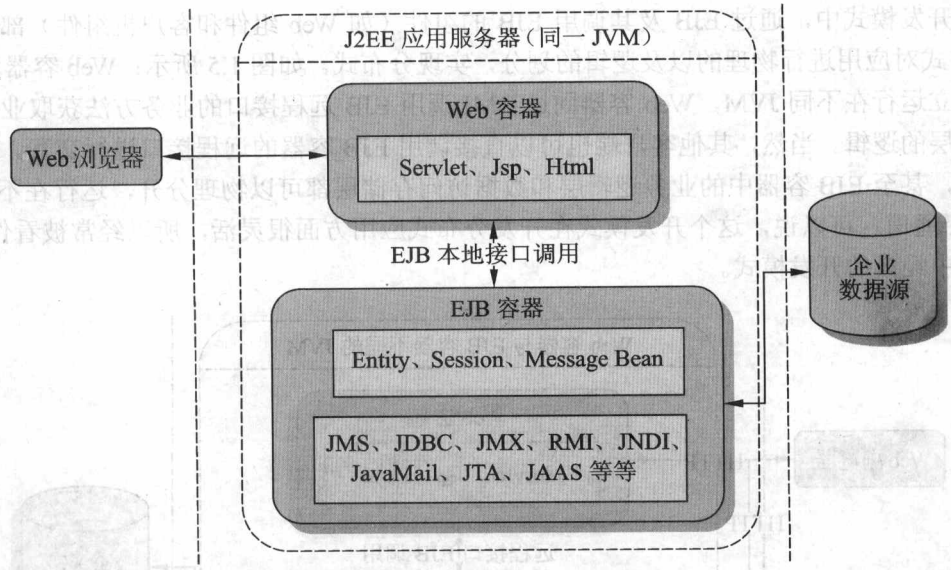


图 1.4

1）优点

（1）基本上具备了“**具有处理业务逻辑组件的 Web 应用**”模式的所有优点。

（2）充分利用了 EJB 容器的各种服务，而又没有分布式 EJB 应用那么复杂。

（3）本地 EJB 的使用只是业务接口实现的一个选择，不会引起基本设计的更改。

（4）本地 EJB 不用远程调用，不需串行化，系统性能开销相当小。

2）缺点

（1）开发相对复杂。

（2）对外也只是一个 Web 接口，很难支持其他客户端，除非多加一层 Web Service。

（3）整个应用运行在同一 JVM，所有的组件必须部署在同一台物理机器上。同样，如果对业务层数据安全要求高的应用，就不适合该模式了。

（4）调试不容易。本地 EJB 调试需要运行 J2EE 应用服务器。

（5）值得注意的是，即使是本地调用，仍然慢于普通方法调用。

这个开发模式，可以看作是在非分布式开发中“**具有处理业务逻辑组件的 Web 应用**”模式和使用 EJB 容器的一个有效的折中模式。这个模式最吸引人的地方就是可以利用 EJB 容器的事务与线程管理带来的各种好处。如果你手头上有 WebLogic、Websphere 或 JBoss 等高效 EJB 应用服务器的话，没有理由不优先考虑采取该模式。

1.2.2 分布式体系结构

J2EE 多层的系统结构，可以说天生就是为了解决企业多层分布式应用的。下面讨论 J2EE 分布式体系结构中两个常见的开发模式。

1. 使用远程 EJB 的分布式应用

在该开发模式中，通过 EJB 及其调用 EJB 的组件（如 Web 组件和客户机组件）部署在不同 JVM 的方式对应用进行物理的以及逻辑的划分，实现分布式。如图 1.5 所示：Web 容器和 EJB 容器各自独立运行在不同 JVM，Web 容器同过 RMI 调用 EJB 远程接口的业务方法获取业务数据来实现表示层的逻辑。当然，其他客户端也可以直接调用 EJB 容器的远程接口进行交互。如果系统需要的话，甚至 EJB 容器中的业务逻辑层和数据访问存储层都可以物理分开，运行在不同 JVM，通过 RMI 通信。可以说，这个开发模式在开发分布式应用方面很灵活，所以经常被看作是 J2EE 体系结构中经典的开发模式。

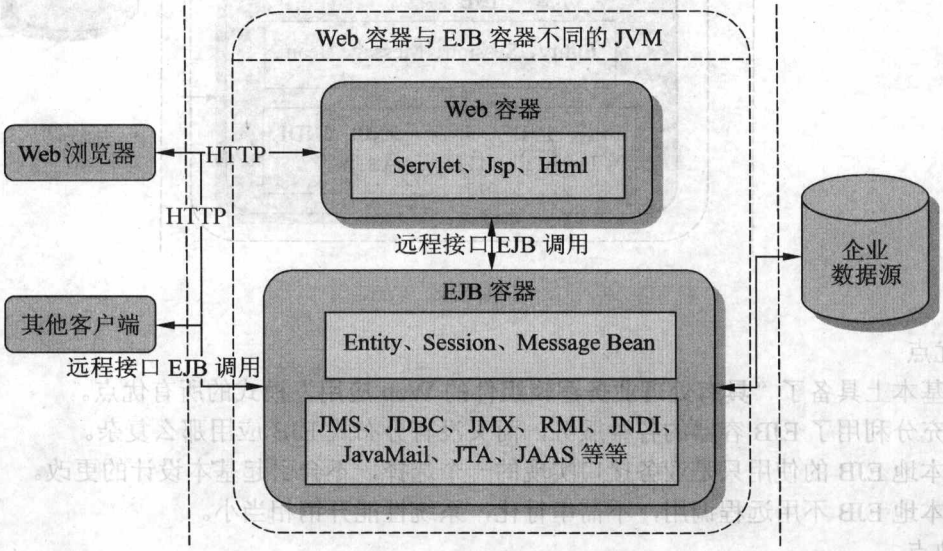


图 1.5

1）优点

（1）它是分布式系统的经典开发模式，可以很容易使用 J2EE 应用服务器的各种现成的服务，减低开发分布式系统的难度。

（2）可以通过标准的 RMI 远程接口来支持所有的 J2EE 客户类型。

（3）提供了最为灵活的部署方式。它允许应用组件在不同物理机器上的部署。合理地使用无状态的 Session Bean，会给 J2EE 应用带来极大的可缩放性。

2）缺点

（1）复杂，开发难度大。这种难度不光表现在技术上，而且很多时候反映在业务逻辑和表示逻辑的合理划分和设计上。

（2）影响性能。因为使用了 RMI 远程调用，而远程调用往往会比本地调用慢许多倍。每个网络传输对象都必须序列化，而太多的序列化也是影响性能的一个重要因素。

（3）所有的业务层组件都必须运行在 EJB 容器内，对 J2EE 应用服务器要求高。

（4）分布式应用的调试和测试都是最难的。

（5）分布式系统使系统异常的处理变得复杂。

（6）还必须考虑网络故障等引起的传输故障问题。

我们罗列了这个模式的不少缺点，但并不是说我们就否定了这个模式。远程接口带来的性能问题，也没有想象中那么严重，其性能影响大小取决于远程调用的数量，而这些我们可以通过合理的设计把调用次数减到最低。同时，如果把远程 EJB 和调用 EJB 的组件放在一起，大多数 J2EE 应用服务器会进行优化，把远程调用替代为本地调用。还有，随着服务器物理硬件的性能提升和网络速度的提高（基本都是 100M 以上），这种远程调用基本上都是毫秒级。故此，如果你的应用有明确的分布式要求的话，笔者推荐你使用本模式开发。但是同时也要注意，不要因为在你的应用中使用了具有远程 EJB，就把整个应用变成分布式的。

2. 具有 Web Service 接口的分布式应用

这个分布式应用模式可以看作是上述非分布式开发体系结构中各种开发模式为了实现分布式开发的一个扩展模式。它在原有的各种模式上增加一个 Web Service 服务层。它不需要 RMI 和 EJB 来支持远程客户，而是采取 Web Service 标准技术来实现分布式应用。任何支持 Web Service 技术的客户端都可以通过标准的协议（SOAP）与中间业务服务器通信。在 J2EE 开发分布式应用中，提供了不使用 EJB 远程接口的另一种选择。

1）优点

（1）Web Service 是标准技术，通信协议 SOAP 比 RMI/IIOP 更开放，可以支持 Java 外其他语言的客户端。

（2）对企业来说，提供标准的 Web 服务接口比提供远程 EJB 接口更有好处。

（3）Web Service 基于 Http 协议，与 RMI 相比，防火墙更友好。

2）缺点

（1）性能差。通过 SOAP 和 XML 协议传递对象的性能比 RMI 传递对象的性能更差。

（2）如果所有的客户端都是基于 J2EE 技术，那么就没有必要采取这样的一个技术来实现分布式。

（3）传输对象编码复杂。可能需要在 Java 对象与 XML 之间作转换，对复杂的 Java 对象支持效果差。

其实，把一个 EJB 转化成一个 Web Service 是很容易的一件事情。

1.3 小结

本章回顾了 J2EE 的体系结构，并讨论了 J2EE 体系结构下非分布式和分布式开发的种种开发模式。读者需要清楚的是，这些开发模式不是独立的，只是作者为了讨论方便根据经验划分的。

往往一个具体的 J2EE 项目都是本章讨论的一种或几种开发模式的组合。

与非分布式结构相比，分布式结构更复杂，更难实现、测试和维护。如果设计不合理的话，分布式的性能往往令人失望。不过，分布式应用在某些情况下会很坚固可靠，更具可缩放性，而且分布式也可能是实现某些业务需要的唯一方法。故此，还是那句话，技术以项目为本，决定是否使用分布式结构开发，要根据项目的具体要求决定。如果项目没有分布式业务的需求，最好避免使用分布式结构，除非它真的给项目带来真正的好处。

值得一提的是，很多开发人员认为“J2EE＝EJB”，认为 J2EE 应用服务器已经处理了复杂的事务管理、数据存取和线程问题，并提供数据缓存功能，其 J2EE“设计模式”是得到公认的方法，因而无须太多地关心效率问题——这种把 EJB 当成是万能药的偏激思想，是危险而盲目的，是又落入了一个“纯”技术陷阱。这观点将会导致一个幼稚而低劣的设计。总体而言，EJB 是一项复杂的技术，能很好地解决一些问题，但是很多时候也给应用增添了比实用价值更大的复杂性。它具有双面性，不加思考地滥用必将是灾难，合理使用才会产生良好的效果，同时让应用具备灵活的缩放性。后面会详细讨论各种 EJB（实体、会话、消息）的优缺点，并希望给出一个关于何时使用、如何正确使用 EJB 的准则。

本章只是回顾了 J2EE 体系结构，对体系结构的各种技术并没有具体讨论，相关技术问题将在框架设计的时候进行详细分析。