国家税务总局综合征管软件V2.0推广组 编

综合征管软件核心技术指南 V2.0

技术篇

中国税务出版社

ZONGHE ZHENGGUAN RUANJIAN V2.0 HEXIN JISHU ZHINAN

JISHU PIAN

ISBN 7-80117-893-9

9 787801 178930 >

ISBN 7-80117-893-9
F·814 定价:30.00元

综合征管软件 V2.0
核心技术指南

技

术

篇

中国税务出版社

图书在版编目(CIP)数据

综合征管软件 V2.0 核心技术指南——技术篇/国家税务总局综合征管软件 V2.0 推广组编. - 北京:中国税务出版社,2006.4
ISBN 7-80117-893-9

Ⅰ.综… Ⅱ.国… Ⅲ.税收管理-管理信息系统-应用软件-中国-指南 Ⅳ.F812.423-39

中国版本图书馆 CIP 数据核字(2006)第 032819 号

书　　名:综合征管软件 V2.0 核心技术指南——技术篇
作　　者:国家税务总局综合征管软件 V2.0 推广组　编
责任编辑:崔　玮
责任校对:于　玲
技术设计:刘冬珂
出版发行:中国税务出版社
北京市宣武区槐柏树后街 21 号　邮编:100053
http://www.taxph.com
E-mail:fxc@taxph.com
发行部电话:(010) 63182980/81/82　63182983(传真)
邮购部电话:(010) 63043870　63028884(传真)
经　　销:各地新华书店
印　　刷:北京凌奇印刷有限公司
规　　格:787×1092　1/16
印　　张:12.25
字　　数:310000 字
版　　次:2006 年 4 月第 1 版　2006 年 4 月第 1 次印刷
书　　号:ISBN 7-80117-893-9/F·814
定　　价:30.00 元

编者前言

本套技术指南对综合征管软件 V2.0 的核心技术架构做了一个较为详细的介绍，以便税务系统的技术人员能够依据本书对综合征管软件 V2.0 有一个系统的了解，并能够对简单的功能进行扩展和优化。

本套技术指南分为技术、应用两篇。技术篇涵盖了综合征管软件 V2.0 的所有技术内容，包括 Web 层、应用层以及数据层的架构设计，给出了模块开发的示例；此外，对工作流、权限等具体的技术应用也做了详细介绍。应用篇分为上册、中册和下册。上册系统地介绍了管理服务子系统，包括税务登记、认定管理、发票管理、证件管理、文书受理、文书管理、税收证明、信息采集等内容；中册系统地介绍了申报征收子系统，包括申报、征收、催报催缴、数据维护、申报征收查询等内容；下册系统地介绍了稽查法制子系统，包括税务稽查、税收法制、税收会计、系统管理等内容。

本书面向的读者为税务系统的技术人员，读者需要的基础技能包括：Java 语言、web 开发技术、XML 基础、UML 基础、J2EE 基础、weblogic8.1，以及 oracle9i 的基础知识及相关的业务知识。

本书由国家税务总局信息中心组织编写，李伟、赵璇、袁立炫、苏小全、王玉娟、朱斌、李志、朱会彦、王晔、鲁钰峰、童咪娜、刘翔宇、陈亚萍、李朝学、崔永毅、孙炜、王凌峰、李春明、彭义勇、古盛新、张雯雯、郑敏虹、蔡培文、侯董宁、刘建华、朱煜权、蔡堉林、林鸿波、余世文、廖伟峰、郑海勇、周凌霄、刘瑞忠、颜友宁、黄智、王国洪、江洪、安冬敏、李红伟、吕志军等同志具体参与了编写工作，王秀、姚琴主审，本书在编写过程中还听取了陈梦林、黄海军、戴文忠、张孟雷、陈北等同志的意见。

本书在编写过程中得到了广东省、深圳市、浙江省、河南省等国家税务局的大力支持，神州数码（中国）有限公司参与了编写工作，在此一并表示衷心的感谢。

由于时间紧，水平有限，本书难免有错误和不足，如有疏漏之处，敬请批评指正。

国家税务总局综合征管软件 V2.0 推广组

2005 年 11 月

目 录

目录

第1章 引 言

1.1 架构

软件体系结构通常被称为架构，ANSI/IEEE 610.12－1990 软件工程标准词汇对于体系结构定义是：“体系结构是以构件、构件之间的关系、构件与环境之间的关系为内容的某一系统的基本组织结构以及指导上述内容设计与演化的原理（principle)”。可以简单理解为：体系结构＝｛构件（component)，连接件（connector)，约束（constrain)｝.其中构件可以是一组代码，如程序的模块；也可以是一个独立的程序，如数据库服务器。连接件可以是过程调用、管道、远程过程调用（RPC）等，用于表示构件之间的相互作用。约束一般为对象连接时的规则，或指明构件连接的形式和条件，例如，上层构件可要求下层构件的服务，反之不行；两对象不得递归地发送消息；什么条件下此种连接无效等。

一般而言，架构有两个要素：首先，它是一个软件系统从整体到部分的最高层次的划分，一个软件系统中的元件首先是逻辑元件，这些逻辑元件如何放到硬件上，以及这些元件如何为整个系统的可扩展性、可靠性、强壮性、灵活性、性能等做出贡献，是非常重要的信息；其次，建造一个系统所作出的最高层次的、以后难以更改的，商业的和技术的决定，也就是说进行软件设计需要做出的决定中，必然会包括逻辑结构、物理结构，以及它们如何影响到系统的所有非功能性特征，这些决定中会有很多是一旦做出，就很难更改。

在开发一个系统之前会有很多的重要决定需要事先做出，而一旦系统开始进行详细设计甚至开发，这些决定就很难更改甚至无法更改。显然，这样的决定必定是有关系统设计成败的最重要决定，必须经过非常慎重的研究和考察。引入架构就是要从更高的层面去考虑问题，把注意力集中在“不变”的因素上，使得最终的软件系统可以实现如下目标：

- 可靠性（Reliable）——软件系统对于用户的商业经营和管理来说极为重要，因此软件系统必须非常可靠。
- 安全性（Secure）——软件系统所承担的交易的商业价值极高，系统的安全性非常重要。
- 可定制化（Customizable）——同样的一套软件，可以根据客户群的不同和市场需求的变化进行调整。
- 可扩展性（Extensible）——软件必须能够在用户的使用率、用户的数目增加很

快的情况下，保持合理的性能。只有这样，才能适应用户的市场扩展的可能性；在新技术出现的时候，一个软件系统应当允许导入新技术，从而对现有系统进行功能和性能的扩展。

- 可维护性（Maintainable）——软件系统的维护包括两方面。一是排除现有的错误；二是将新的软件需求反映到现有系统中去。一个易于维护的系统可以有效地降低技术支持的花费。
- 客户体验（Customer Experience）——软件系统必须易于使用。
- 市场时机（Time to Market）——软件用户要面临同业竞争，软件提供商也要面临同业竞争。以最快的速度争夺市场先机非常重要。

1.2　框架

框架，即 framework。框架是在给定问题领域范围内建立起可重用解决设计问题而互相间协作的一组类的集合（它包括了具备缺省行为的大量对象），然而框架不仅仅是行为的集合，还包括了支配着把这些行为组合在一起的方法的一组规则，或称为协议。框架包含了一组相互关联类一起工作的方法，框架在一个给定领域范围中，解决了大量具体问题的通用的设计。框架其实就是某种应用的半成品，就是一组组件，供你选用完成你自己的系统。简单说就是使用别人搭好的舞台，你来做表演。

框架一般处在低层应用平台（如 J2EE）和高层业务逻辑之间的中间层。框架一般都是遵循好莱坞原则设计的，否则就不叫框架。所谓好莱坞原则，说的是 You don’t call us-we will call you. 意思就是在一个框架下的代码，都是被动地被框架调用，而不是相反（如图 1－1 所示）。

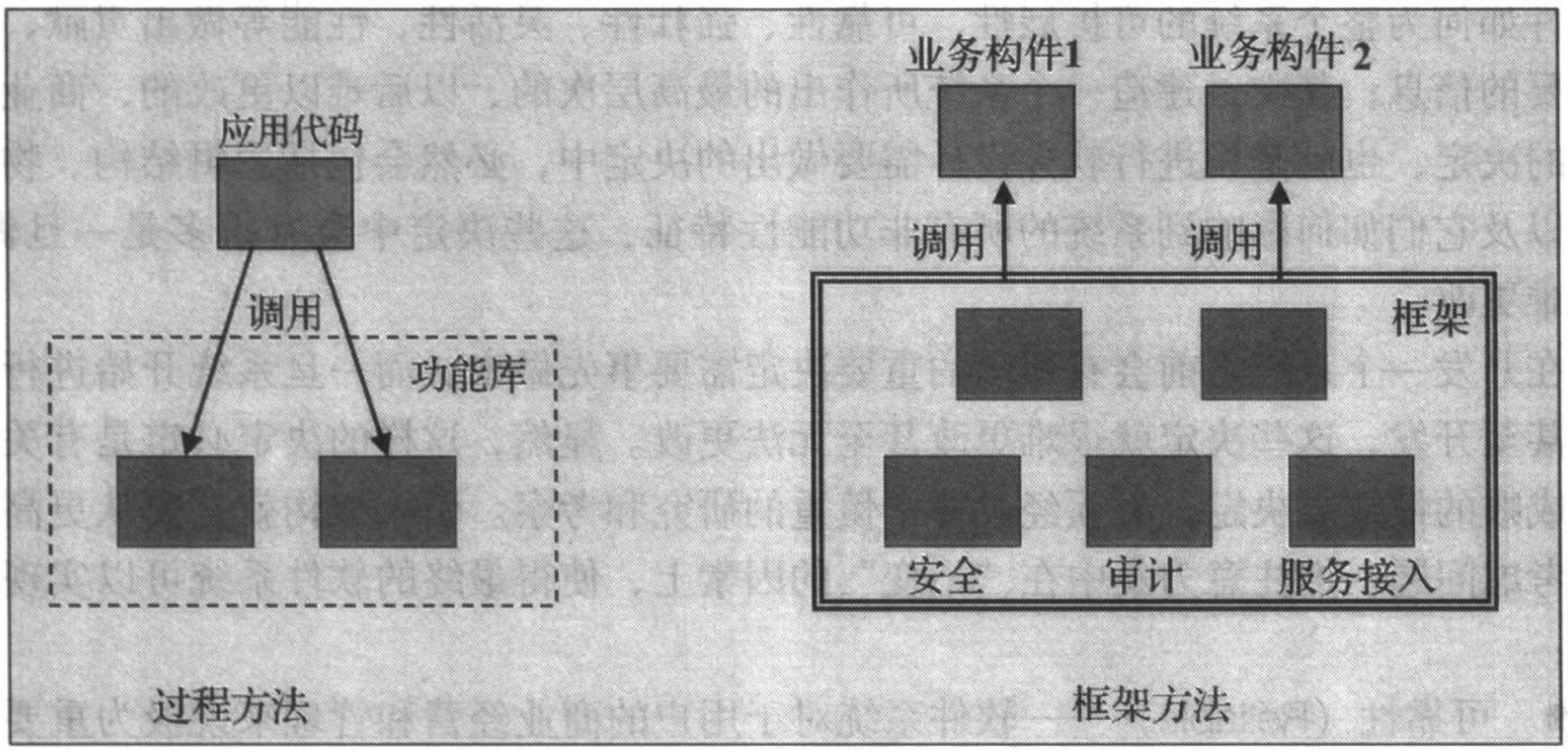

图 1－1

通过这种方式，大量重复的代码就可以隐藏在框架里面，需要特别设计的代码以预定接口的方式交给开发人员，写好后由框架调用。譬如 JSP 就是一个框架。你写的 JSP 脚本会被 JSP 引擎编译成 servlet 的一部分。这种 servlet 都带有大量重复的代码，是 JSP 程序员不需要考虑的，由框架负责。框架的主要优点有：减少编程的代码、增加代

码的可靠性和健壮性、更能保证一致性和模块化、提供了通用领域的问题（如用户接口，图形界面或网络操作等等）服务 。

因为软件系统发展到今天已经很复杂了，特别是服务器端软件，涉及的知识，内容，问题太多。在某些方面使用别人成熟的框架，就相当于让别人帮你完成一些基础工作，你只需要集中精力完成系统的业务逻辑设计。而且框架一般是成熟，稳健的，他可以处理系统很多细节问题，比如，事物处理，安全性，数据流控制等问题。还有框架一般都经过很多人使用，所以结构很好，扩展性也很好，而且它是不断升级的，你可以直接享受别人升级代码带来的好处。

1.3 设计模式

在说明设计模式之前，我们先了解一下什么是“模式”。

模式（Pattern）的概念最早由建筑大师 Christopher Alexander 于 20 世纪 70 年代提出，应用于建筑领域，80 年代中期由 Ward Cunningham 和 Kent Beck 将其思想引入到软件领域。Christopher Alexander 将模式分为三个部分：首先是周境（Context，也可以称为上下文），指模式在何种状况下发生作用；其二是动机（System of Forces），意指问题或预期的目标；其三是解决方案（Solution），指平衡各动机或解决所阐述问题的一个构造或配置（Configuration）。他提出，模式是表示周境、动机、解决方案三个方面关系的一个规则，每个模式描述了一个在某种周境下不断重复发生的问题，以及该问题解决方案的核心所在，模式既是一个事物（thing）又是一个过程（process），不仅描述该事物本身，而且提出了通过怎样的过程来产生该事物。这一定义已被软件界广为接受。

在不同的层面上，模式提供不同层面的指导。根据处理问题的粒度不同，从高到低，模式分为 3 个层次：架构模式（Architectural Pattern）、设计模式（Design Pattern）、实现模式（Implementation Pattern）。架构模式是模式中的最高层次，描述软件系统里的基本的结构组织或纲要，通常提供一组事先定义好的子系统，指定它们的责任，并给出把它们组织在一起的法则和指南。比如，用户和文件系统安全策略模型，N－层结构，组件对象服务等，我们熟知的 MVC 结构也属于架构模式的层次。一个架构模式常常可以分解成很多个设计模式的联合使用。设计模式是模式中的第二层次，用来处理程序设计中反复出现的问题。例如，［GOF95］［2］总结的 23 个基本设计模式——Factory Pattern，Observer Pattern 等等。实现模式是最低也是最具体的层次，处理具体到编程语言的问题。比如，类名，变量名，函数名的命名规则；异常处理的规则等等。

在 Frame 中用到了大量的设计模式，下面举例说明策略模式在 Frame 中的应用：

策略模式的模型如图 1－2。

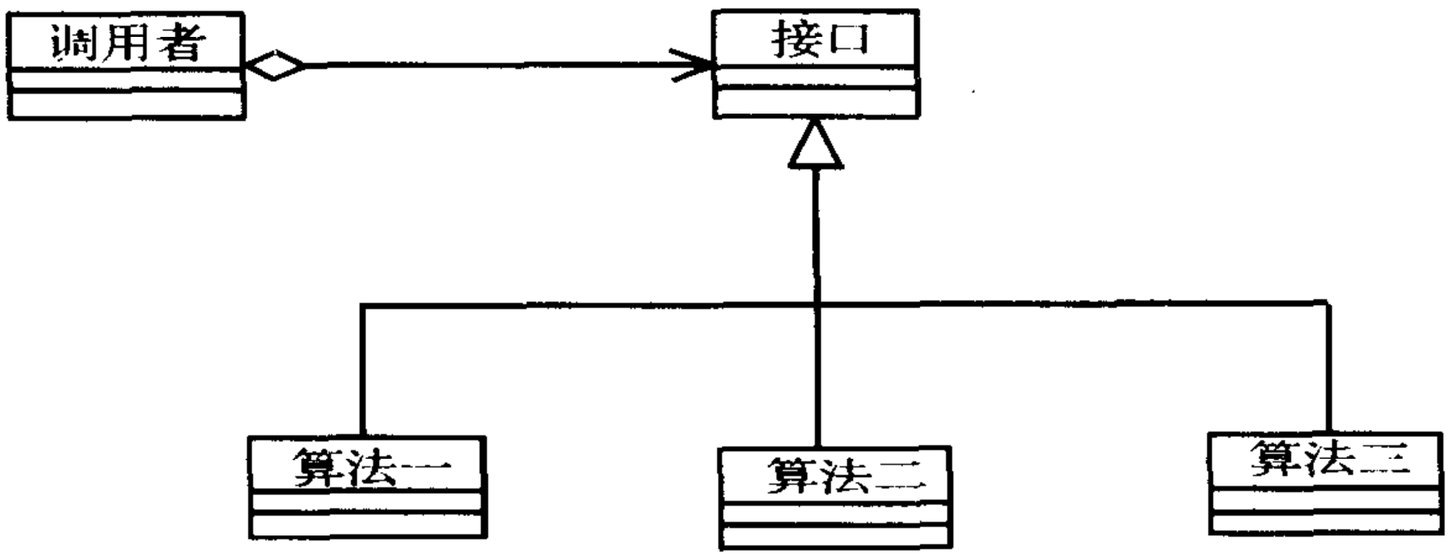

图 1-2

Frame 中对抽象类 AbstractConfig 的五个继承类是对 AbstractConfig 的五种不同实现，通过 ConfigManagerServer 对 AbstractConfig 的调用选择五种算法的一种，是策略模式的应用，如图 1-3。

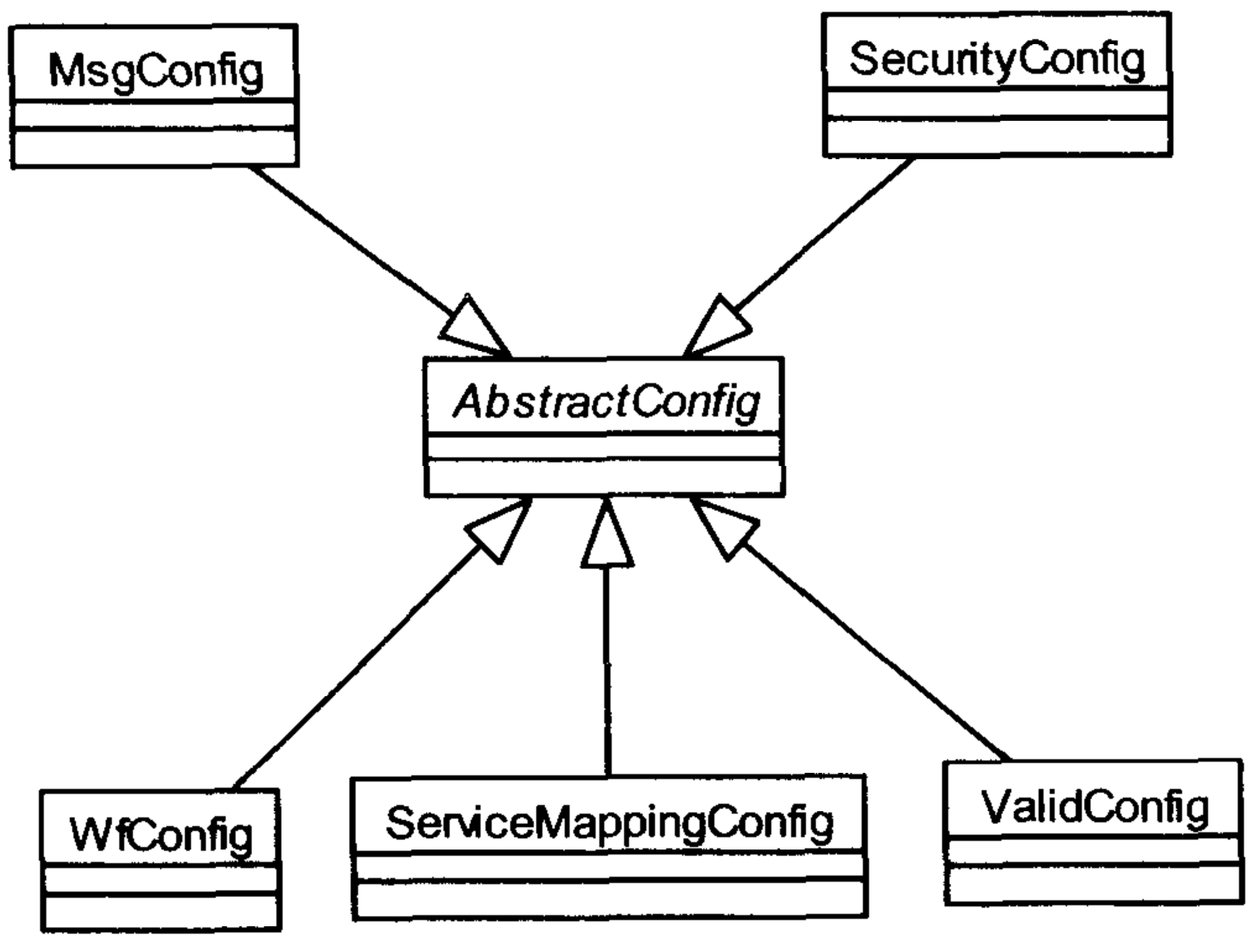

图 1-3

设计模式的应用对软件开发产生了重大的作用。设计模式是人们在长期的设计软件、管理组织软件开发等实践中大量经验的提炼和抽象，是复用软件设计方法、过程管理经验的有力工具。设计模式类似于拳击中的组合拳，它提供了一系列软件开发中的思维套路。如，通过设计模式的使用，有利于在复杂的系统中产生简洁、精巧的设计。设计模式为我们提供了一套简洁通用的设计、管理、组织方面的词汇，同时设计模式也为我们提供了一个描述抽象事物的规范标准，可大大促进软件开发过程中人与人之间的交流，而软件开发中的交流是至关重要的，“软件项目失败的原因最终都可追溯到信息没有及时准确地传递到应该接收它的人”。

从前面的说明中我们知道软件的大尺度结构就是架构。一个软件不管好坏，都会有一个架构。软件架构中可以利用框架，也可以不利用框架。譬如前面提到 JSP 是一种框架，而你的系统可以利用 JSP 框架，形成自己的架构。从另一个角度来看，如果用房屋作比喻，架构就是忽略掉细节的抽象建筑结构，在图纸可以看到，存在于人脑之中，不

体现为房屋的某一个物理部分。框架是房屋的骨架，房屋的骨架是物理存在的。

架构和设计模式应该是一个属于相互涵盖的过程，但是总体来说 Architecture 更加关注的是所谓的 High-Level Design，而设计模式关注的重点在于通过经验提取的“准则或指导方案”在设计中的应用，因此在不同层面考虑问题的时候就形成了不同问题域上的 Pattern。设计模式的目标是，把共通问题中的不变部分和变化部分分离出来。不变的部分，就构成了设计模式，因此，设计模式是一个经验提取的“准则”，并且在一次一次的实践中得到验证，在不同的层次有不同的模式，小到语言实现（如 Singleton）大到架构。

对于熟悉架构设计的系统架构师而言，似乎可以这样来解释架构和模式之间的关系：架构是 Hight-Level Design，着眼于不同业务中共性的解决方案，而模式是 General Principle（通用原理）。

1.4　参考资料

- 《软件体系结构（影印版）》，科学出版社 2004 年 1 月 1 日出版。Mary Shaw、David Garlan 合著，原文书名《Software Architecture: Perspectives on an Emerging Discipline》。
- GoF95，《设计模式——可复用面向对象软件的基础》，Erich Gamma、Richard Helm 等著，英文版本《Design Patterns：Elements of Reusable Object-Oriented Software》，这是设计模式领域的经典之作，它结合设计实例从面向对象的设计中精选出 23 个设计模式，总结了面向对象设计中最有价值的经验，并且用简洁可复用的形式表达出来。
- ［Buschmann96］Buschmann，Frank，《Pattern-Oriented Software Architecture》，John Wiley & Sons Ltd，1996。中文版《面向模式的软件体系结构》，2003 年 1 月机械工业出版社出版。

第2章 总体概述

2.1 逻辑架构

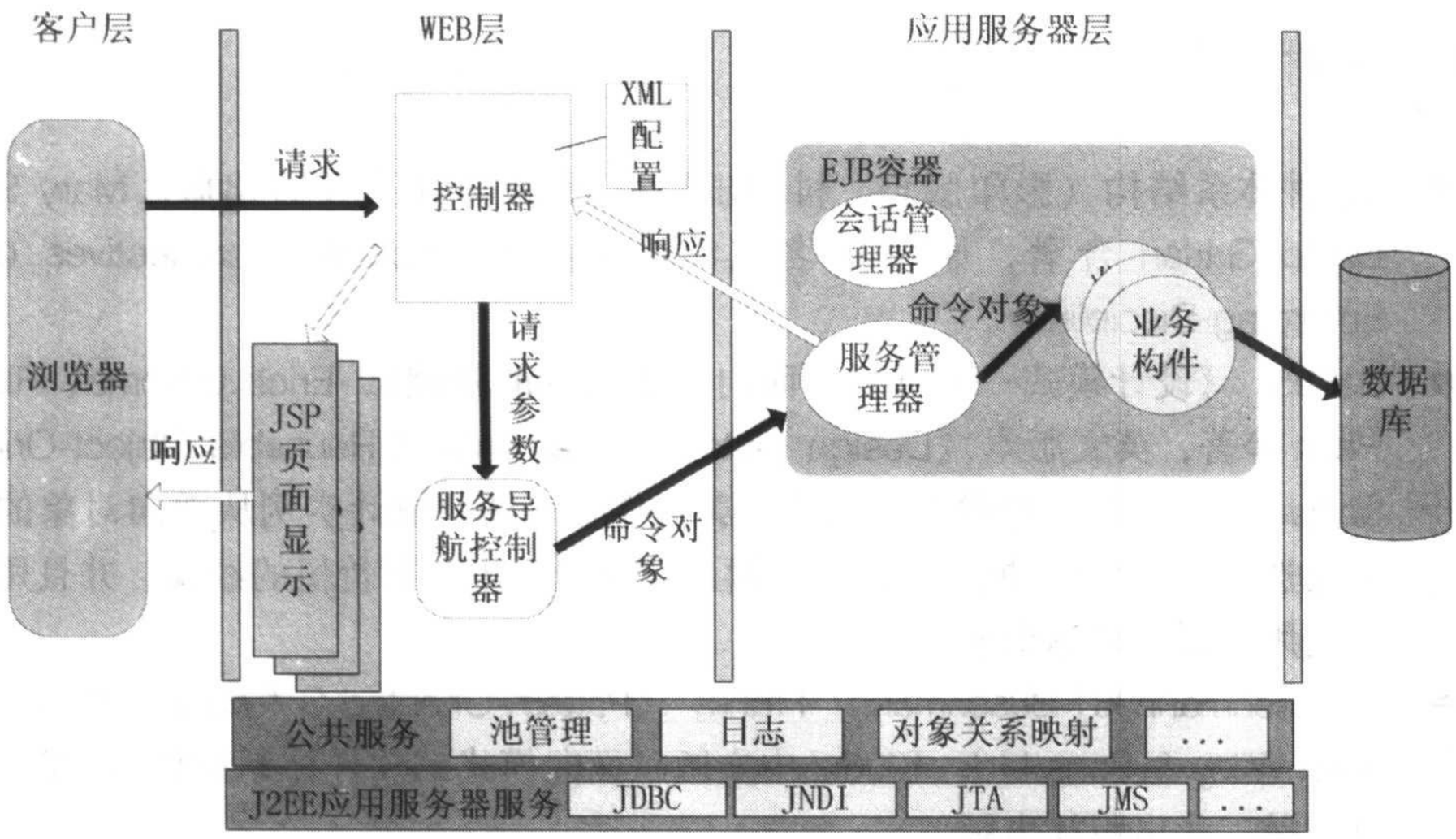

图 2-1 逻辑架构图

客户层

客户层负责提供不同渠道的展示与呈现能力，它能够根据要求，将同一应用层的返回结果以不同的技术手段展现给用户。由于客户层和应用层的交换基于 XML，因而客户层支持不同的实现方式，包括浏览器、电话、手机、语音等。

Web 层

web 层基于 Servlet 和 Jsp 技术开发，采用了 MVC（视图 – 模型 – 控制器）架构模式。

- 模型：表示企业数据和管理对该数据的访问和更新的业务规则。模型采用 EJB 技术实现。
- 视图：展示的内容。它通过模型访问企业数据，并指定应该如何表示该数据，视图将负责在它的表示中保持一致性。视图可以是 jsp，html，Swing GUI 等。