企业应用架构模式

一.分层

多层蛋糕：上层使用下层的服务，而下层对上层一无所知，且对上层隐藏其下层的细节；

好处：

- 减少层次间的依赖性
- 层次不能封装所有东西，例如增加一个在用户界面上显示的数据域数据库和每层都必须修改
- 层次过多影响性能，可以优化事务控制层

1.企业应用中层次的演化

1.1客户/服务器

适合数据的简单显示和修改，问题在于领域逻辑（业务规则，验证，计算等）；

将领域模型嵌入用户界面，容易产生冗余代码；

将领域模型放入数据库端，作为存储过程，但存储过程只提供有限的结构化机制，而且不能更换数据库厂商；

tier物理上的分离，layer旨在强调无需把不同层次放在不同的计算机上运行；

1.2

表1-1 三个基本层次

层 次	职 责
表现层	提供服务，显示信息（例如在Windows或HTML页面中，处理用户请求（鼠标点击，键盘敲击等），HTTP请求，命令行调用，批处理API）
领域层	逻辑，系统中真正的核心
数据源层	与数据库、消息系统、事务管理器及其他软件包通信

领域层：计算输入数据以及确定调度那些数据源逻辑；

一个单独的企业应用，可能在三个层次上都包含多个软件包。客户端或者命令行，对于不同的数据库也需要多个数据源软件。

表现层是系统对外提供服务的接口，数据源层是系统使用外部服务的接口；

领域层和数据源层绝对不要依赖于表现层；

1.3为各层选择运行环境

用户在使用胖客户端时，表现层运行在客户端；用户在使用WEB的HTM1网页时表现层运行在服务端；表现层最好全部运行在服务端，软件不用担心于客户端同步更新的问题，也不必考虑软件和客户端的兼容问题，但是运行在客户端更能满足客户需求，操作简单。

领域逻辑可以运行在客户端也可以运行在服务端，最糟糕的时将领域逻辑分割在客户端和服务端。

二．领域逻辑模式

领域模式的组织可以分为三种主要的模式：事务脚本，领域模型，表模块。

事务脚本：用来保存领域模型最简单，每一个动作由一个过程来驱动，容易理解但容易产生重复代码。领域模型：因为复杂的逻辑必然要引入对象，所以就产生了领域模型。领域模型主要围绕领域中的名词进行组织，还包括完成业务逻辑的方法。从过程到对象来控制逻辑。领域模型的好处在于当增加新的算法时只需增加相应的策略对象而事务脚本需要在脚本的判断逻辑中增加许多新的条件。领域模型的开销来自使用和数据源层上的复杂性。表模块：和领域模型相似，关键区别在于领域模型对数据库中每一个合同都有一个相应合同类的示例，而表模块只有一个公共的合同类示例。表模块与记录集一起工作，如果要针对某一个合同操作，必须在调用方法时附上该合同ID，它围绕表来组织逻辑提供了更多结构，但无法应用继承、策略和其他面向对象的设计模式。

处理领域逻辑常见的方法是将领域层细分成服务层和领域模型层或表模块。表现逻辑与领域模型的交互完全通过服务层，服务层像一个API。尽可能最小化的使用服务层。

通盘考虑：领域层：表模块模式是一个比较好的折中。数据源层：事务脚本无需映射，在行数据和表数据接口中选一个，表模块需要一个与记录集配合良好的映射模式即表数据接口，领域模型最使用一种映射工具。表现层：领域模型使用实体Beans。

模式：当领域逻辑朴素，使用实体beans实现领域模型；当领域逻辑复杂，比如使用了继承、策略或其他，使用

POJO领域模型和数据映射器。策略类提供了一个良好的插入点来进行扩展。