引言

企业应用的特点：持久化数据，不同的显示界面，多人并发访问，业务逻辑复杂。

基于Web的网站企业应用

性能要求：响应时间，吞吐量，负载，可伸缩性（增加内存云上的时代到来了）

GOF模式：方便交流，合适的场景，只有在使用中发挥作用。脚手架。

计算机的分层：例网络7层模型。解耦合，社会分工细化，专业化。透明（下层对上层是透明的）。分层的原因：分工，解耦，面向接口。

第一章

随着web的兴起，B/S架构，分层：表现层，领域层，数据源层。

表现层：HTML，控制台，exe，jsp，applet。

领域层：领域逻辑层，控制逻辑

数据源层：持久化数据

怎么确定领域逻辑层是分离的？当我更换控制台程序为jsp时，不用修改逻辑代码，当我更换数据库时，不用修改领域逻辑层代码。

领域层和数据源层不要依赖表现层，表现层只依赖逻辑层提供数据，对于表现层来说，数据源层是透明的。

第二章

领域逻辑层：事务脚本、领域模型、表模块

事务脚本：一个逻辑一个事务脚本，比较清晰明了，大多数开发者都能够看懂，但是复用性差，复杂的业务逻辑比较混乱。

领域模型：对象控制用户的某一个逻辑动作。每一个对象承担一部分的责任。复用性强。但领域模型的初始化开销是比较大的。

表模块：查询出来的数据集以表结构组织，然后再把表数据传到表现层（GUI）。一定程度上可以很好的利用数据。

通常的做法是将领域模型层分为两层：服务层、底层（领域模型和表模型），服务层只相当于接口，给表现层调用。

第八章

领域模型和关系型数据库： O/R映射工具

三种情况下的数据源：

事务脚本：行数据入口和表数据入口？什么意思？

领域模型：ORM工具访问数据。

表模块：具有记录行数据的框架，业务逻辑适中的比较合适

表现层：不同的应用有不同的表现，如HTML网页应用 MVC（模型层---视图层—控制层）表现层对应视图层，而控制层则处于表现层和领域层的中间层。因此，控制层可以很好的区分，分离表现层和领域层。

第九章

事务逻辑（Transaction Script）每一个过程处理表现层的单个请求。

领域模型（Domain Model）合并了行为和数据的领域对象模型。

数据和行为绑定才能叫对象，对象具有属性和方法。

关键的是怎么持久化的问题？对象的模型但是是关系型的数据库，最近听说了OSS对象存储。对象之间通过传递行为进行通信，减少判断。代码过于臃肿，本来一个类解决的问题，现在分职责，可能需要十个类来处理，增加了代码的复杂度。

表模块（Table Module）以数据库表（或视图）为蓝本。

需要数据库表的ID作为标识，

总感觉再重复之前的内容，如果是已经了解了，在看后面的这章的时候，只是加深印象，还不如实际敲敲代码，再回过来看，可能会理解的更好。

服务层（Service Layer）

应用的逻辑，使用多个领域模型的对象处理一个实际的业务逻辑。

属于中间层，封装领域模型的单个逻辑。

总结：

最后，关键的是：在选择这些分层的模式的时候，主要是以业务逻辑的复杂程度来确定的。当然我可以不以这个作为条件，我只会其中的一种，那就不必再说了。如果要判断业务逻辑的复杂程度的话，怎么判断？经验？阅历？我也迷糊了。