领域驱动设计是分解复杂系统的一套方法，它并不依赖于任何技术

在面对一个复杂的系统时，如果你关注了局部，那你很难关注到整体，未经设计的系统，整体和局部是混杂在一起的，小到一个类，大到一个系统，这样就会衍生出一些问题，在面对上层决策时，你会去考虑数据库数据库是如何设计的吗，在开发某个功能时，你回去考虑整个系统是如何允许的吗？

领域驱动设计需要做的就是分解系统，使整体和局部分离，使各个局部之间相互独立，减少学习成本

按照领域驱动设计方法，在面对一个复杂的系统时，我们首先要做的就是划分上下文，除了共享核心模式的上下文，各个上下文就是一个独立的子系统，上下文内的改变不会影响其他上下文，在上下文层面，我们需要关注的是如何划分上下文，以及确定上下文之间的关系，以及上下文之间的交互，上下文层面主要是对接上层决策和物理架构的部署。

在上下文内部，我们会去分离出上下文的核心域，将上下文拆分成多个子域，我们的目的就是为了精简核心，这让我们知道当前上下文开发的核心，以及不要让其他辅助功能影响核心功能的开发

在域的下面，我们又对实体进行一个划分，我们将实体划分为多个聚合，各个聚合之间只能关联聚合根，而不能管理聚合内的实体，其目的是为了保证聚合的一个独立性，聚合内的变动不会影响到其他聚合的设计

上面讲的是领域模型的一个设计方式，其基本的原理就是分解，分离关注点，让我们更好的理解整体和局部

下面说一下领域驱动设计的一个分层

领域驱动设计分为4层

首先就是最核心的领域层了，领域层是我们整个上下文中最核心的部分，其设计是领域层只依赖于框架，其不依赖于任何层，领域层只有业务代码，其包含的对象有聚合，实体，值对象，规约，仓储，仓库，领域服务

然后是应用层，应用层面向的就是我们的应用功能了，一个应用服务就代表了我们的一个用例，应用层的职责包括验证DTO，获取实体，调用领域服务，调用基础设施，控制事务等

最外层是展示层，其职责就是展示页面，在前后的分离的项目中，展示层并不在后端项目中，取而代之的是API服务层，其提供了API接口并控制后端返回的错误码

最后是基础设施层，基础设施层提供了上下文所需的一些基础设施，比如日志，定时服务，文件的操作，仓库的实现等，均属于基础设施层

还有一个非必须的层，就是转换层或者称防腐层，当上下文需要访问其他上下文时，为了保证上下文内的稳定性，引入了这个上下文，其隔离了我们上下文了其他上下文