我们前面的教程都只涉及单个领域模型

在大型系统中我们会有多个应用，每个应用都会有自己的领域模型（有的应用甚至不进行模型设计），那我们要合并这些模型吗？答：

大型系统的领域模型难以做到统一，而且我们也不应该那么做，这会给我们带来如下麻烦

1. 学习成本加大
2. 重构成本加大
3. 为了适用不同的场合使模型变得复杂

同理：当一个领域模型太大时，我们应该对其进行划分，我们在领域模型上画上几个圈，将领域模型分割成多个领域模型，我们把这个圈称为边界上下文

**边界上下文**

边界上线文规定了哪些功能属于当前上下文

如：我们要开发一个仓库管理系统，要为其开发一个客户端和管理端，还有一个定时服务用于拉取数据更新到老系统上面，还有一个运输车辆管理系统管理进出仓库的车辆，我们要为这个仓库管理系统定义一个边界上下文（就叫仓库上下文吧）

1. 客户端管理端服务于仓库管理系统，所以其属于仓库上下文
2. 定时服务会更新老系统数据，放到老系统上下文会更合适
3. 运输车辆管理系统与仓库管理上下文有一些交集，但其拥有自己独立的模型和界面，我们不妨将他划分为独立的上下文，当然你也可以将其归属为仓库上下文，但这样会使你的模型变大，而好处就是你不用为上下文之间的通信烦恼

不同上下文之间的模型“一般”共享

**划分上下文**

1. 寻找关注点，将一个大的领域模型分为几个小的领域模型
2. 当你的领域模型中出现的对歧义时，如点餐的客户和开房的客户不是同一类客户，是否想过将系统分为点餐系统和开发系统
3. 如果你有2个团队正在开发2个应用，而这2个团队的交流较少，那你应该将这2个应用划分为2个独立的上线文

**边界上下文是一个领域概念，其声明了当前模型的应用范围**

前台客房预订系统（位于客房预订上下文中）与后台客房管理系统（位于客房管理上下文中）不在一个边界上下文内，因为他们的模型不能共享（如：前台客房预订有订单的概念，但后台客房管理并不需要订单的概念），当我们新增一个概念“清洁员”，那么这个概念应该放在哪？明显应该放在后台客房管理系统

**不同的边界上下文不应该共享代码，除非分离出共享核心**

我们现在想要vip预订系统，vip预订系统需要使用前台客房预订系统的模型，所以vip预订系统属于客房预订上下文，所以我们的vip预订系统不应该使用后台客房管理系统的模型

**边界上下文不是module**

边界上下文可以包含多个module，但不要在一个module中包含多个边界上下文

**持续集成**

把一个上下文内的工作频繁的合并到一起（概念和代码），这样可以迅速的发现问题

（上下文有很多子领域组成）程序员在域A改了AClass，域B使用域A，如果程序员更改错误，在集成时进行单元测试就会发现

**边界上线文与微服务**

边界上下文与微服务的概念相似，我们可以依据边界上下文来划分微服务