下面列出了上下文之间的关系

**共享核心**

当两个边界上下文需要共享一些模型，在这个时候，我们通常需要分离出这些共享的模型，放到共享核心（一个module），共享内核通常由一个或多个 核心域 或 通用子领域 （后面会讲）组成 ，共享核心由负责这两个上下文的团队负责，测试也是由这两个团队编写

建立的共享核心的上下文可以共享模型

共享内核的2个上下文在同一个项目中，其中共享的代码放在一个module中

如下示例（C#），上下文1和2其共享了Order域和User域



注：共享核心只共享领域层对象，不共享如应用层、展示层、基础设施层等

**客户/供应商**

一个系统服务于另一个系统，那么一个上下文服务于另一个上下文

客户/供应商不只是定义了上下文之间的关系，也定义了2个团队之间的关系，即供应商团队要为客户团队提供一组稳定的接口

客户/供应商的2个上下文一般在不同的项目中，他们没有共用的模型，处于这种关系的客户上下文一般具有简单的转换层，其目前是为了使客户上下文不因供应商上下文的变动而变动

**墨守成规**

新开发的系统需要应用现有系统的部分模型，但应用现有系统的开发团队不会根据新开发系统团队的需求来提供接口，这时候可以选择使用现有的模型进行开发。

意思是我们想使用现有的模型，但现有的模型不会因为我们的使用而做任何改变

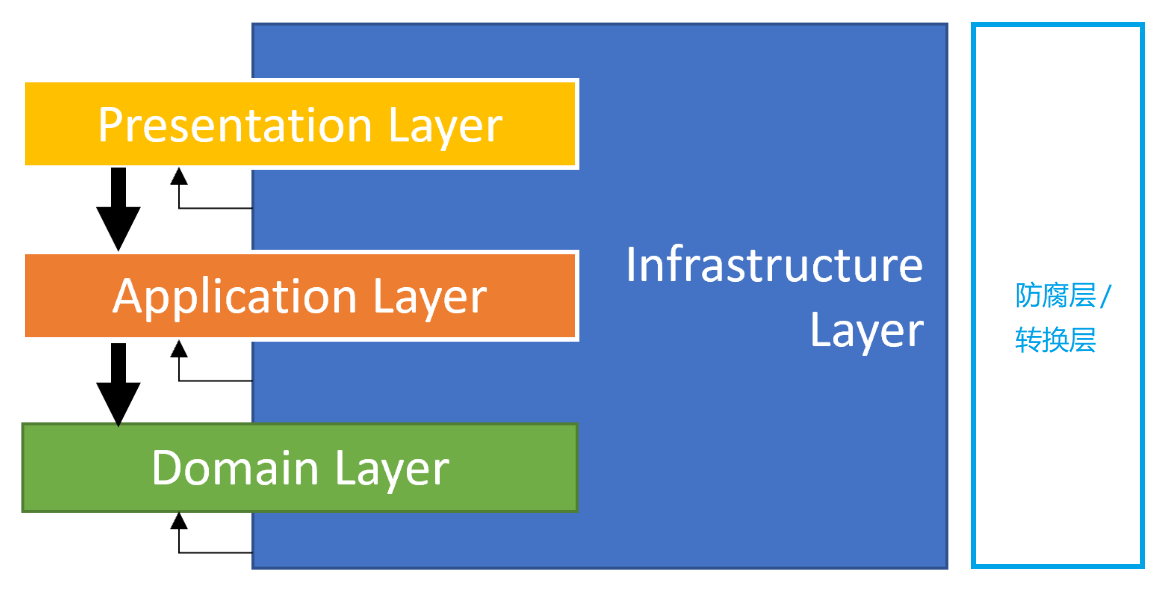
墨守成规关系的2个上下文在同一个项目中，上下文B可以使用上下文A的模型，但不能改变上下文A的模型

**防腐层**

如果你不想使用上下文A的模型，但又必须与上下文A进行交互，你可能会想到客户/供应商模式，但如果上下文A不想成为供应商，那你只能增加防腐层来隔离上下文A，使其不干扰现在的设计。

防腐层是上下文的一部分，是一个module，其提供了一组service的集合，使用外观模式，返回我们系统期望的模型

防腐层是链接两个上下文的一种方式



防腐层/转换层依赖与我们的领域层，并为应用层和展示层提供Service，当需要调用其他上下文时，我们会委托防腐层/转换层的Service去调用，其可以接收或返回我们所需实体

**各行其道**

集成（将现有功能应用放到当前上下文中）的代价总是昂贵，如果一个与其他上下文不相关的上下文那会很容易开发。

不用集成也就意味着我们可以将代码放在各自的项目中，不用担心其他上下文的改动会影响到我们的代码了

微服务的开发采用的便是这种方式，各个微服务只关系消息，而不关系其他微服务的存在与否