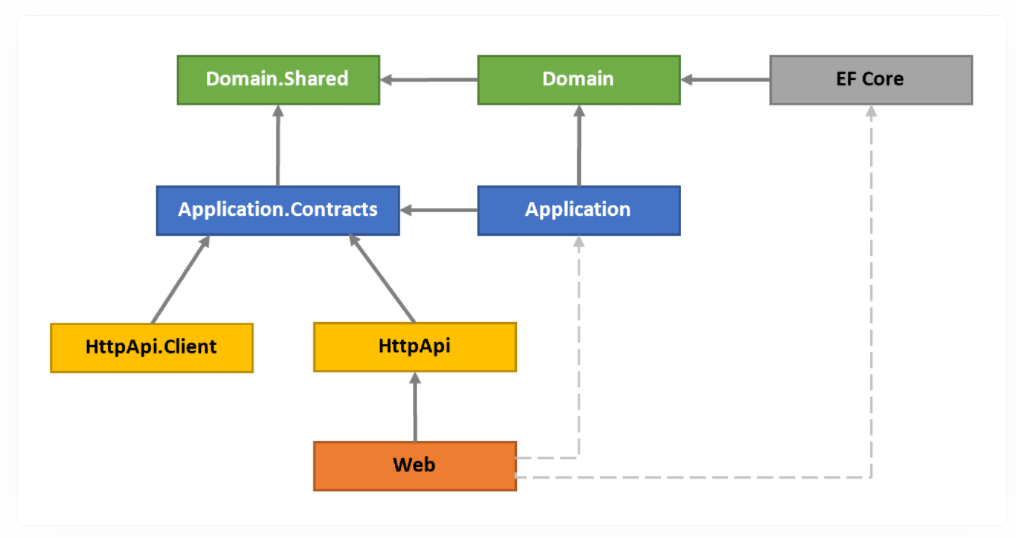
原文地址：<https://docs.abp.io/zh-Hans/abp/latest/Domain-Driven-Design-Implementation-Guide>

**项目依赖关系**



**领域层**

领域层分为两个项目:

Domain是领域层中必需的,它包含之前介绍的构建组成(实体,值对象,领域服务,规约,仓储接口等).

Domain.Shared是领域层中很薄的项目,它只包含领域层与其它层共享的数据类型的定义.例如,枚举,常量等.

**应用层**

应用层也被分为了两个项目:

Application.Contracts包含接口的定义及接口依赖的DTO,此项目可以被展现层或其它客户端应用程序引用.

Application是应用层中必需的,它实现了IssueTracking.Application.Contracts项目中定义的接口.

**展现层**

Web是一个ASP.NET Core MVC / Razor Pages应用程序.它是提供UI元素及API服务的可执行程序.

**远程服务层**

HttpApi包含了HTTP API的定义.它通常包含MVC Controller 和 Model(如果有).因此,你可以在此项目中提供HTTP API.

注：ABP框架的API自发现系统可以自动将应用服务公开为API,因此，通常不需要在此项目中再创建Controller

HttpApi.Client当C#客户端应用程序需要调用IssueTracking.HttpApi的API时,这个项目非常有用.客户端程序仅需引用此项目就可以通过依赖注入方式,远程调用应用服务.

注：因为只是演示,你可以删除此项目,再或者你认为IssueTracking.HttpApi不需要,同样可以删除.

**基础设施层**

你可能只创建一个基础设施项目来完成所有抽象类的定义及外部类的集成，但我们建议采用一种平衡的方法：为主要的依赖的库(例如 Entity Framework Core)创建一个独立的项目,为其它的依赖库创建一个公共的基础设施项目.

ABP的启动解决方案中包含两个用于集成Entity Framework Core的项目:

EntityFrameworkCore是必需的，包含数据库上下文，数据库对象映射，仓储接口的实现，以及其它与EF Core相关的内容都位于此项目中.

EntityFrameworkCore.DbMigrations是管理Code First方式数据库迁移记录的特殊项目

**虚线依赖**

你在上图中会发现用虚线表示了另外两个依赖.Web项目依赖了 Application and EntityFrameworkCore，理论上Web不应该依赖这两个项目，但实际上依赖了，这种依赖关系的设计，可能会让你有机会在展现层直接使用到EF Core的对象，但应该严格禁止这样的做法

**DDD模式的应用程序执行顺序**

下图展示了基于DDD模式下的Web应用程序执行顺序

