DDD的一些事儿：

goland项目：一个server服务对应一个app，所以有一个app文件夹，下面都是关于这个app的所有代码。

App下分为：biz目录业务代码、cmd目录是main函数启动代码、conf目录对应app的配置文件。

biz目录可以进行DDD分层：

第一层：adapter（controller），就是传说中的controller层，但是细分为http、rpc，都是对外提供的接口。

第二层：application（业务调度层），这一层看似是业务层，分为app目录、cqe目录、dto目录。

其中app就是业务调度层，看似是做业务，但是实则知识做业务调度，先做什么、再做什么。

cqe目录下存放的都是用于接收，外部调我们接口传入的值（比如存放json反序列化的数据、query参数、header参数等）。

dto目录下存放的都是返回给调用方的数据。

第三层：domain层（实际写业务的地方），分为entity、repo、service、vo、gateway。

其中entity是一个聚合类，并不是对应数据库中的一张表。聚合了种数据库对象，比如有一个用户表，还有用户关系表，那么我要处理这个业务就会创建一个entity聚合类，这个entity就是（用户+用户关系）

Repo:仓储层（并不是写sql的），是实际处理业务的一层，并且业务较为简单。Repo仅是api，具体实现在infrastructure层的persistence中。

Service：业务层，但是一般处理的业务都是非常复杂的

Vo:存放的是对entity中字段的封装（即entity的个别属性是vo包下的），目的是有的时候需要做字段校验，我们可以将字段封装起来，同时字段校验在vo封装的结构体中。

Gtateway:防腐层，封装了一些接口，比如消息队列的push、pull接口，具体实现gateway不关心，即使最后消息队列从kafka换成了rocketmq，使用gtateway的并不需要关心。

第四层：infrastructure层。下面主要的databse包等关于数据存储相关的。其中database最常见。Database下面的包有po、dao、persistence。

Po：就是对应数据库的一张表

Dao：就是写持久化语句的地方，sql、mongo等。

Persistence：实现了repo接口的地方。

DDD的调用关系：

首先adapter（controller）接收来自用户的请求，通过封装好的cmd\query对象去保存请求中数据（query、json、path等）。然后调用app接口，将数据发送给application层。

由app层对业务进行调度，即app层会将接收到的请求数据（cmd/query）封装为entity发送给domain的repo\service去执行真正的业务。

最后repo\service层会将entity封装为po对象，调用infrastructure层的dao对数据进行持久化或者查询数据库数据。

如果前端请求是get请求的话，需要查询数据库消息，那么最后数据库查到的po会返回给domain层（处理业务），再返回给app层封装为dto最后返回给adapter（controller）层返回给前端。