探索与实践

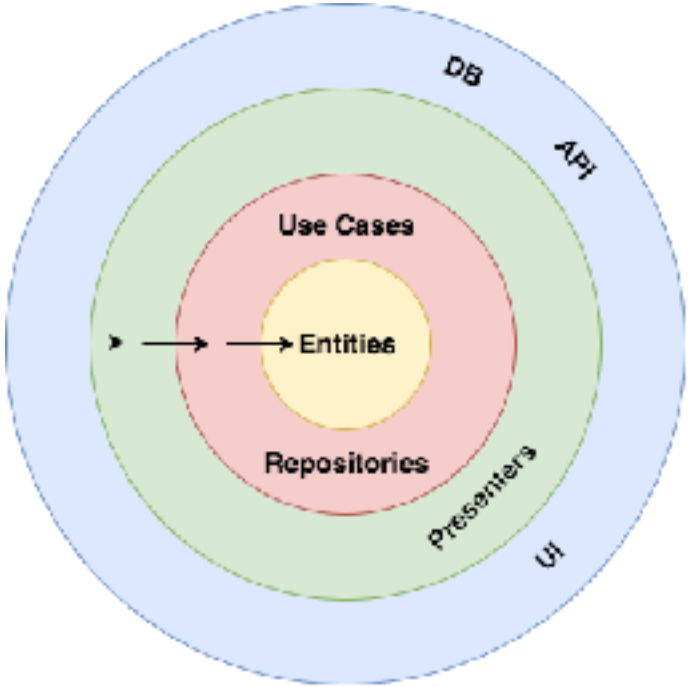
Viper

目录

1. Viper演进史
2. 新闻架构演进
3. 视频详情页重构
4. Viper的使用
5. 附录

Viper演进史

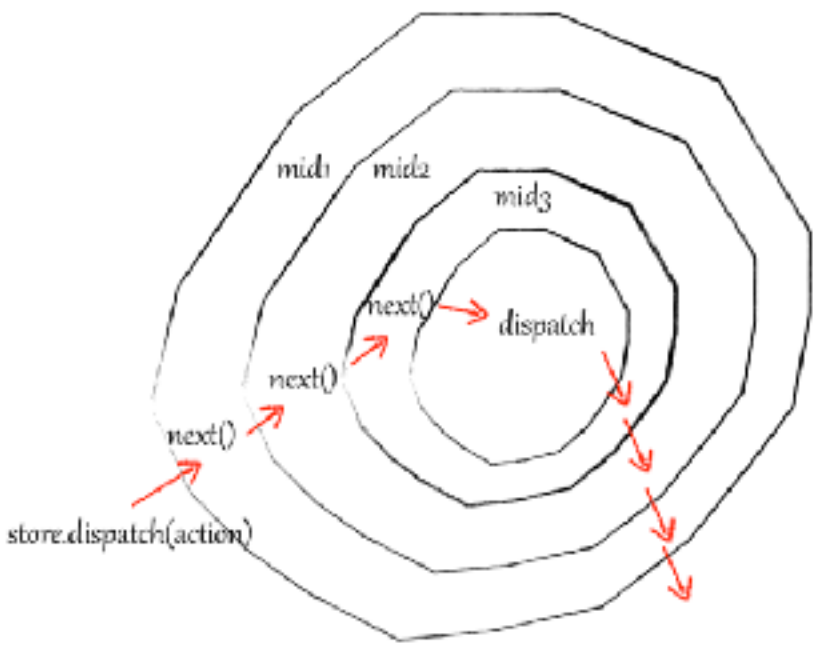
1. 洋葱模型
2. The Clean Architecture
3. MVP-Clean
4. Viper

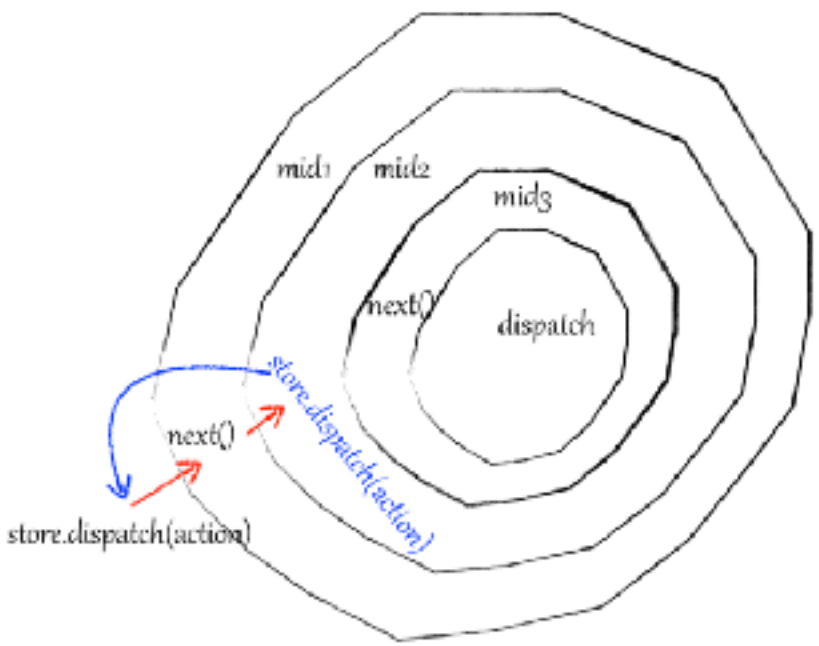


洋葱模型

洋葱模型，是从冰山模型上演变而来的，用来进行层次分析的模型，这是Redux的洋葱模型。

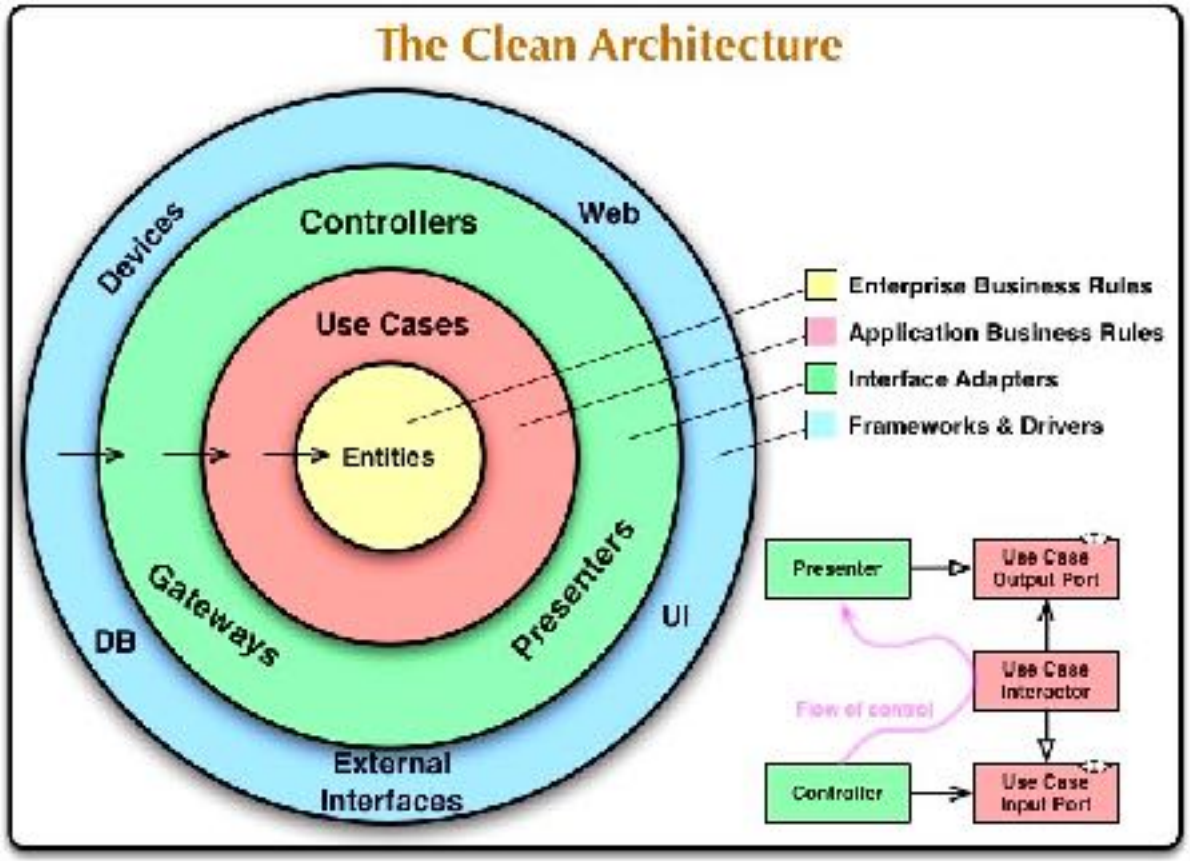
洋葱模型如今已经广泛应用于各个领域，进行更直观清晰的分层剖析。





The Clean Architecture

Robert C·Martin是《Clean Code》的作者，我们习惯称他为Uncle Bob。2012年8月13日，他在他的个人Blog上，提出了著名的The Clean Architecture。



The Clean Architecture 优点

- 独立的框架：框架可作为工具，不与系统混在一起；
- 可测试：没有UI、数据库和Web服务器，数据也可以测试；
- UI独立：UI改变更容易，而不会影响业务规则；
- 数据库独立：业务规则不会和数据库绑定，可以任意切换数据库；
- 外部代理独立：业务规则可以对外部世界无感知。

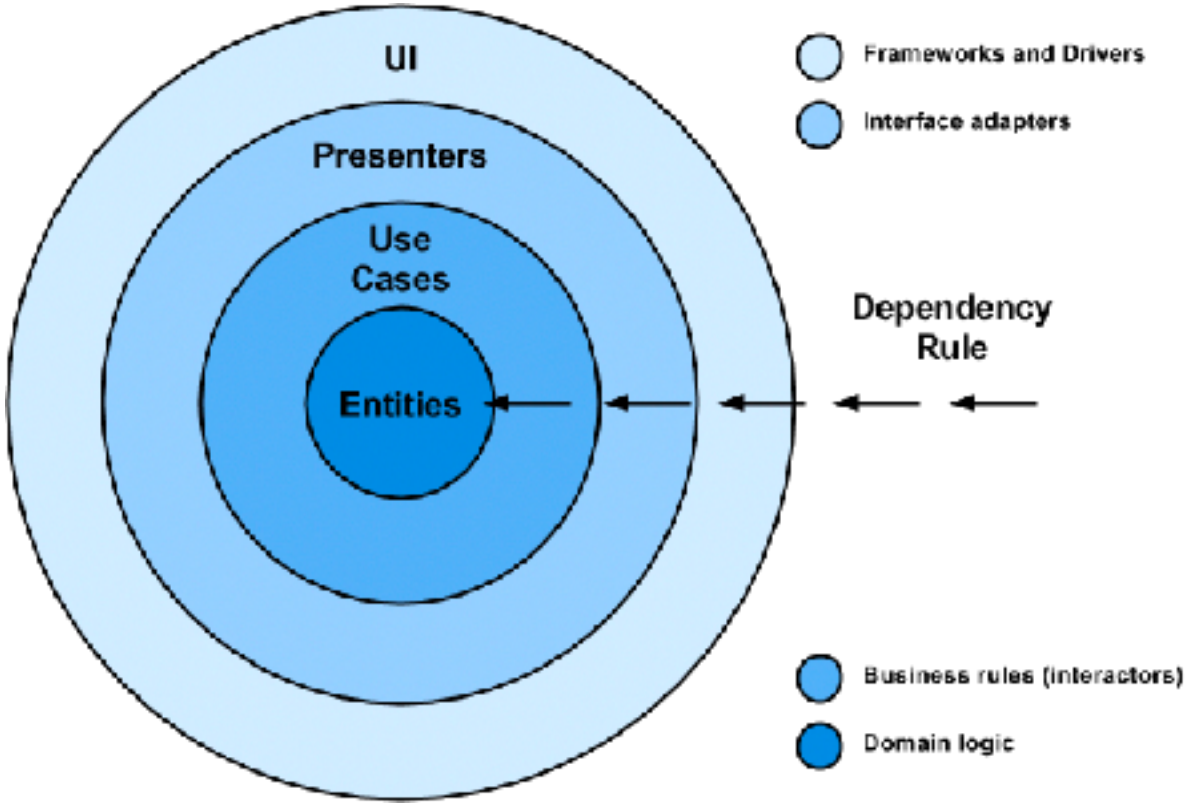
The Clean Architecture 评价

> For me, this has been the best way to develop apps so far. Decoupled code makes it easy to focus your attention on specific issues without a lot of bloatware getting in the way. After all, I think this is a pretty SOLID approach but it does take some time getting used to.

开发者Dario Miličić说，“对于我来说，这是目前最好的App开发方式。解耦的代码使你更容易将注意力集中在具体问题上，而不是被臃肿的代码所妨碍。我认为这是一个相当可靠的方法，但它需要一段时间去适应。”

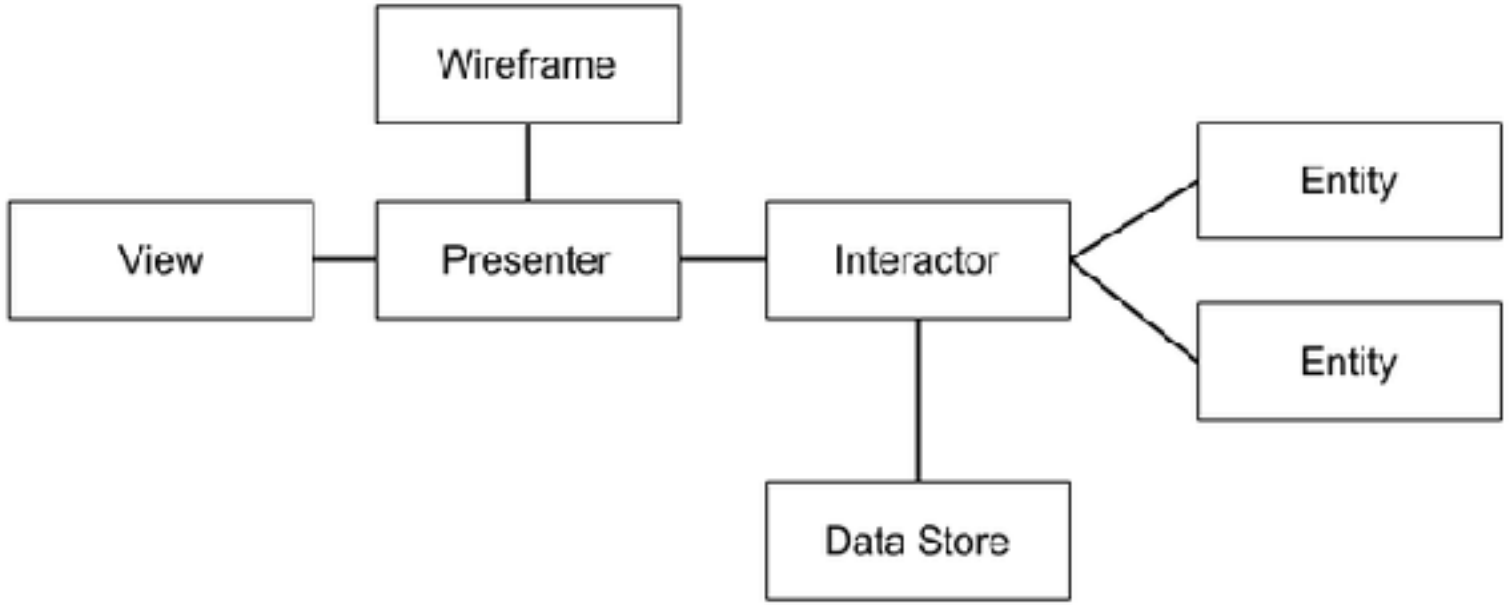
MVP–Clean

Google官方发布了安卓架构蓝图Android Architecture Blueprints，其中包括MVP + Clean Architecture，在MVP-Clean中，其在传统的MVP上，基于The Clean Architecture，对MVP做了改进。

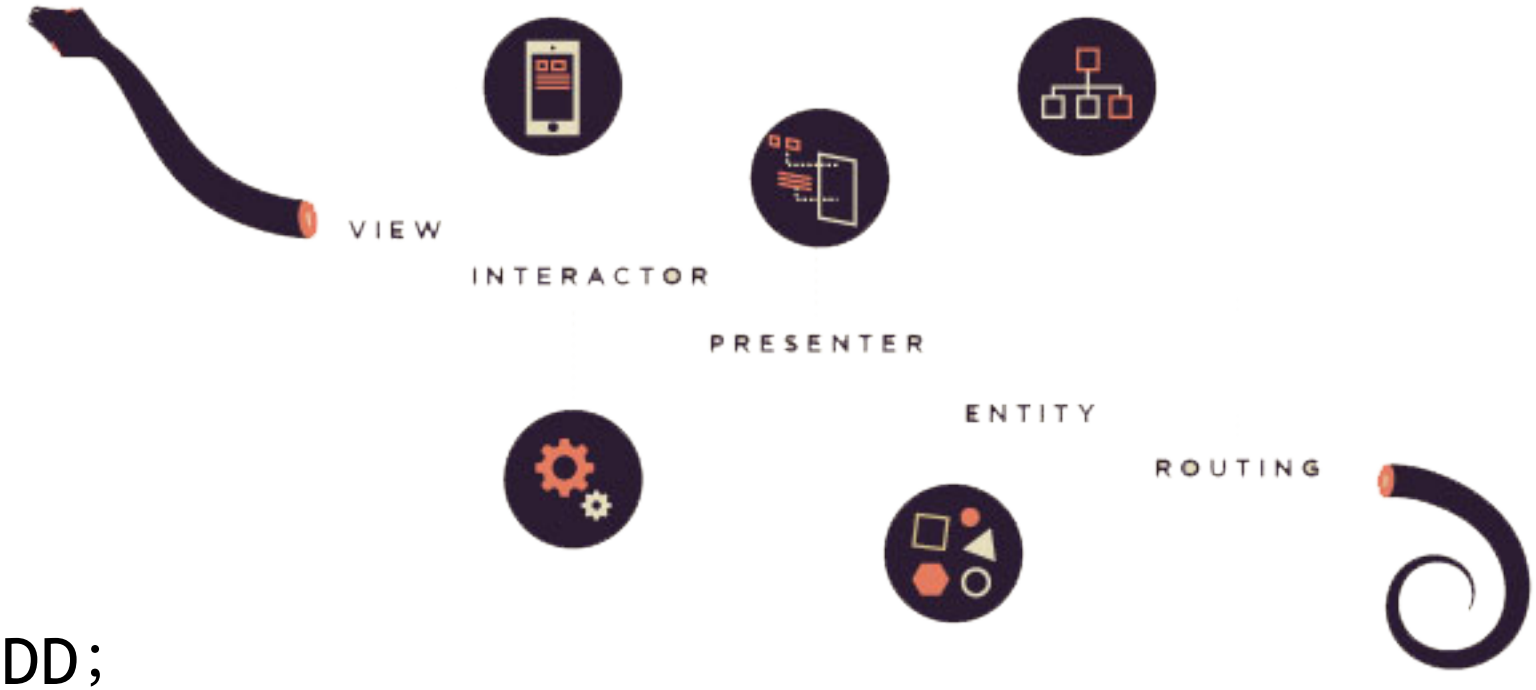


Viper

VIPER中各个字母分别代表View、Interactor、Presenter、Entity和Router。



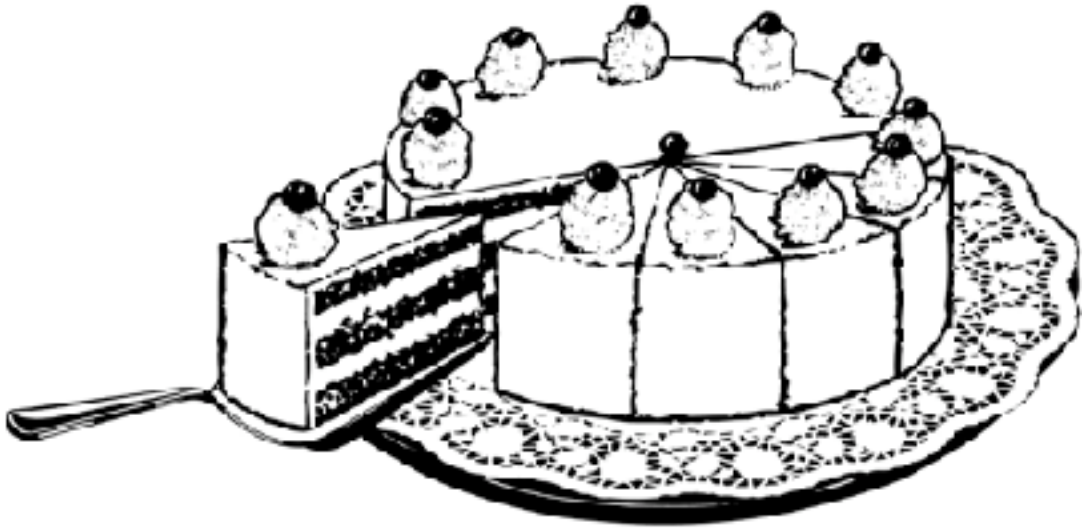
Viper优点



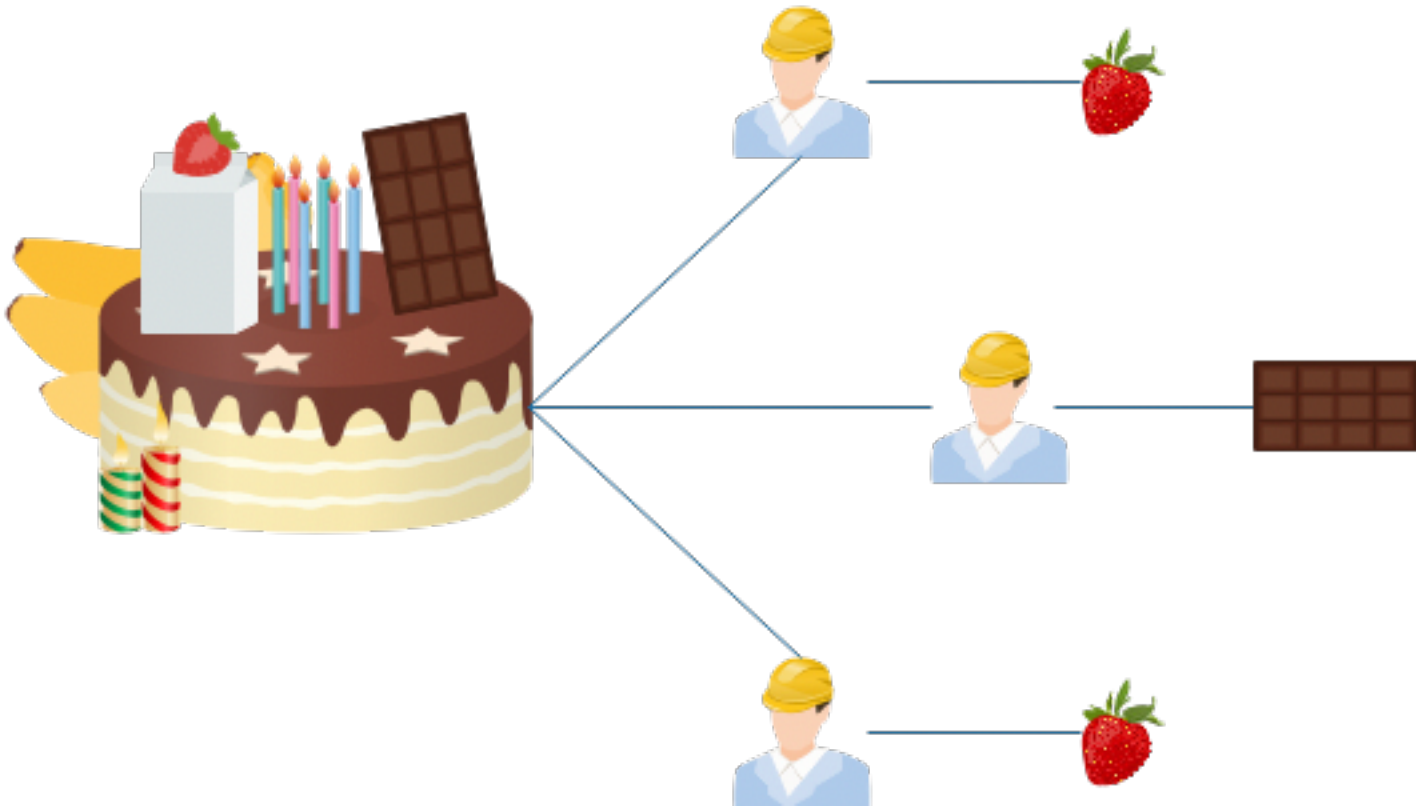
- 易于测试；
- 代码结构更清晰，符合DDD；
- 良好的分离关注点；
- 责任被划分的很清晰：做什么和怎么做。

新闻架构演进

1. Static Method（业务静态工具类阶段）
2. MVP（MVP一对一阶段）
3. MVPs（MVP一对多阶段）
4. Viper的探索



Static Method（业务静态工具类阶段）



优缺点

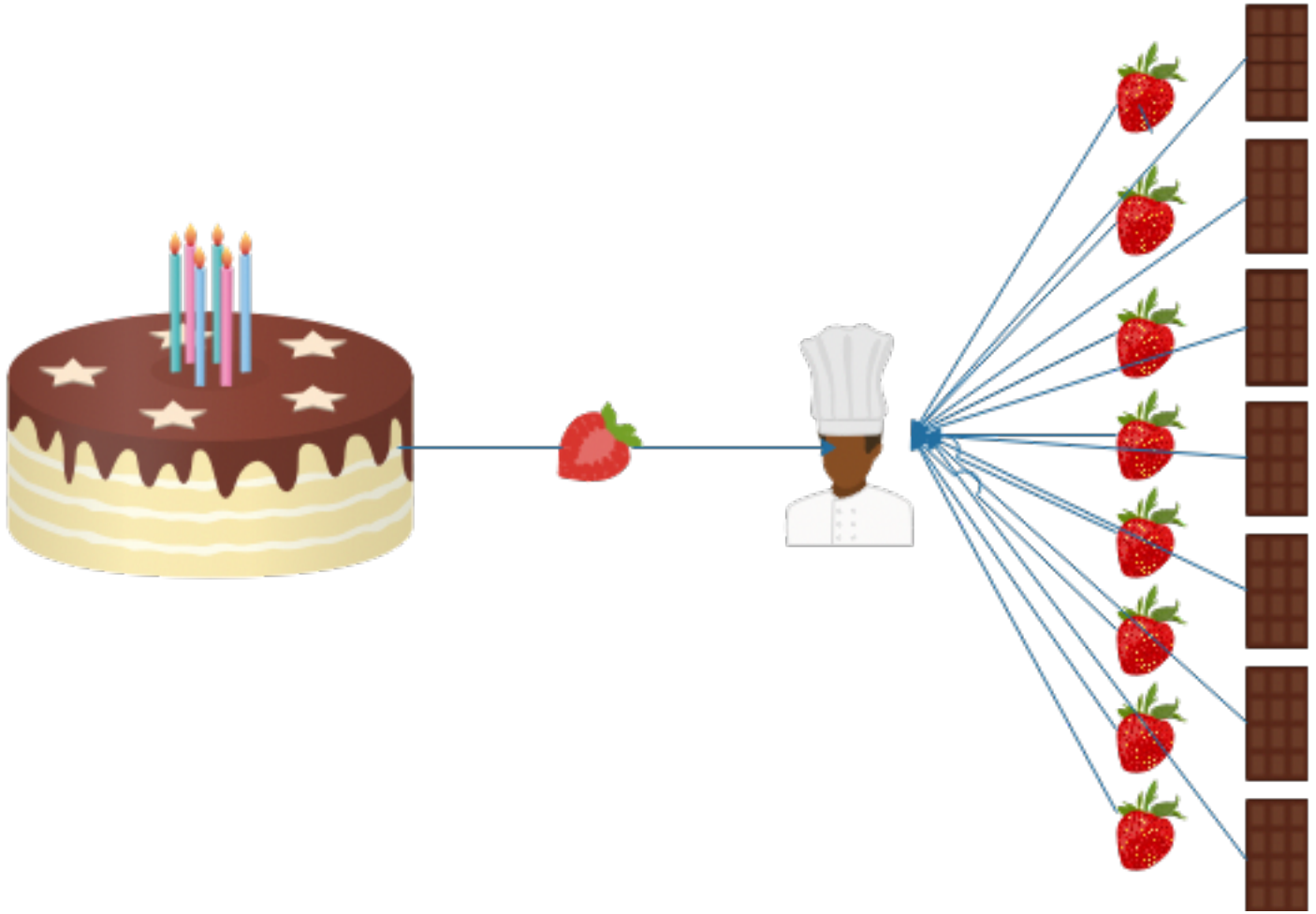
优点：

- 不用实例化即可使用；
- 使用方便简洁，学习成本低；
- 便于处理数据逻辑。

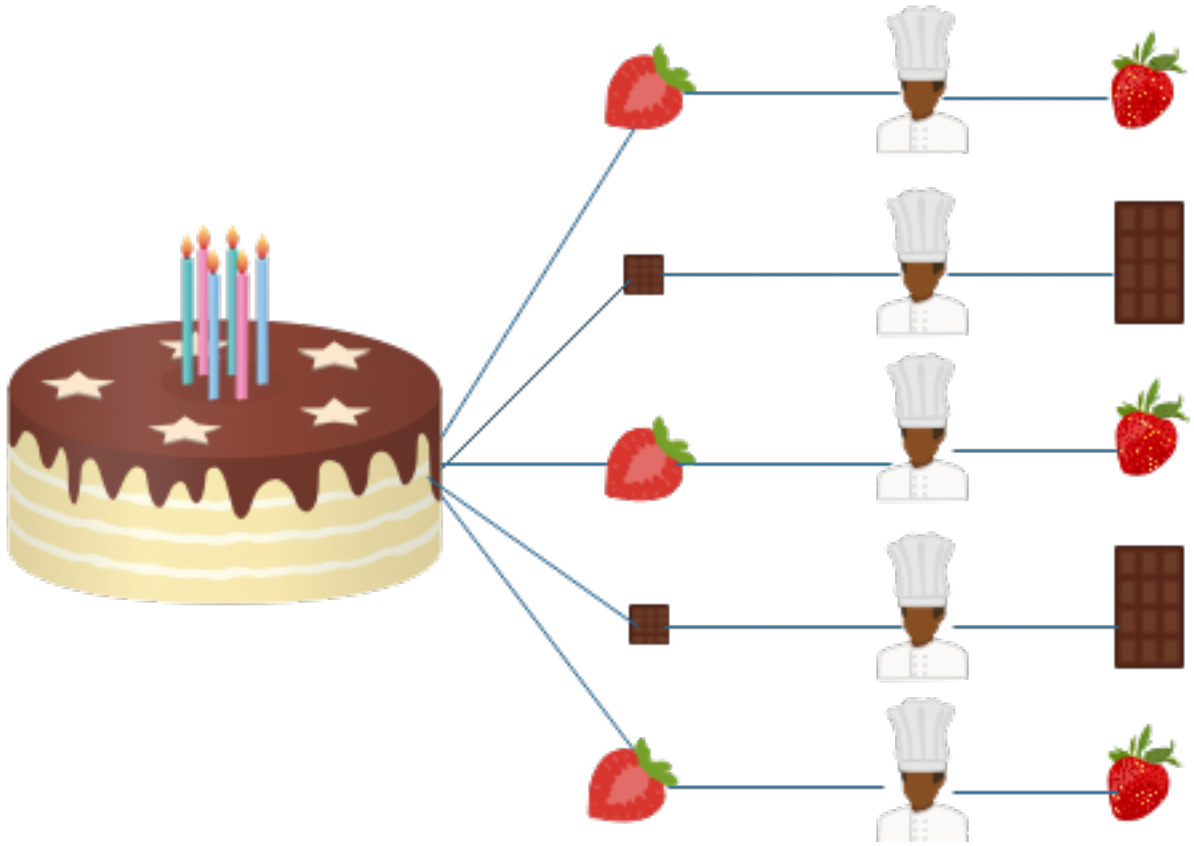
缺点：

- 不符合面向对象的设计原则；
- 接口不能定义静态方法；
- 不能调用非静态方法与变量；
- 容易造成参数列表的混乱。

MVP（MVP一对一阶段）



MVPs（MVP一对多阶段）



优缺点

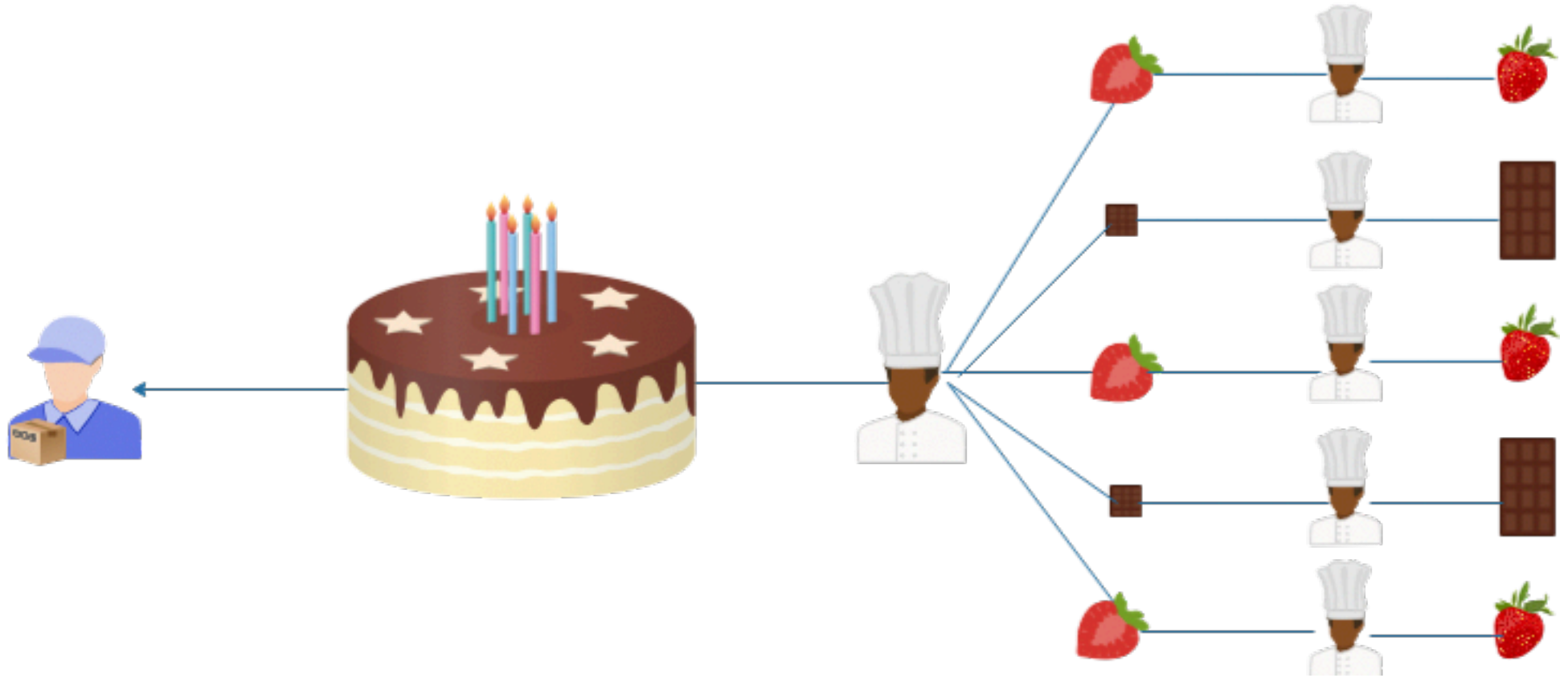
优点：

- 将原本臃肿的Presenter零散化，使得Presenter的负担不至于过重；
- 代码可以通过多个Presenter进行复用。

缺点：

- Presenter应该是处理中心，越多的Presenter的介入导致View的管理设计结构越混乱；
- Presenter的生命周期管理变得不易于控制；
- Presenter对应的View表现不再单一化，为了适配不同的View需要考虑多种场景；
- 更多View接口的实现导致产生更多空实现方法，影响了代码的整洁性，易使人产生阅读混淆。

Viper的探索



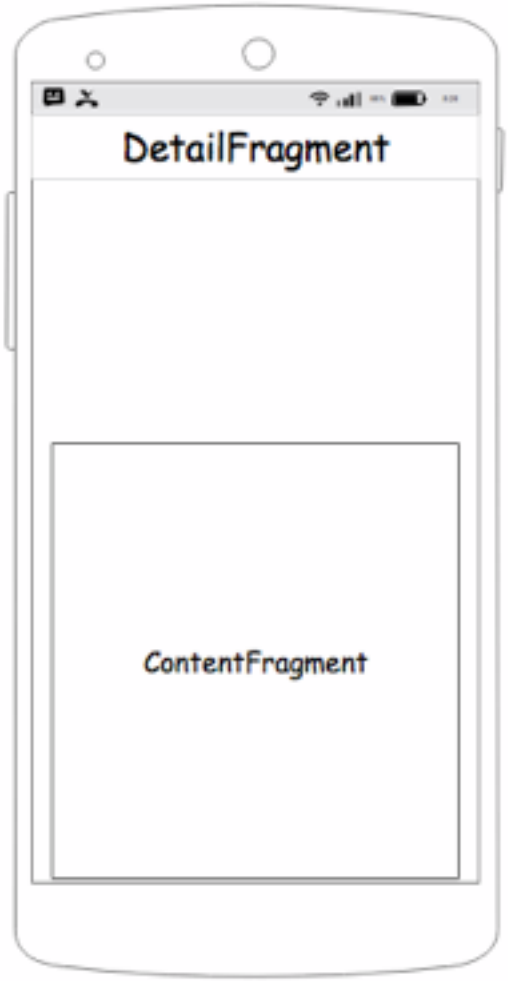
优点

- 有一个**中央控制器**，负责控制View的各种业务逻辑交互（MVP一对多不符合）；
- 业务逻辑代码可以**复用**（MVP一对一不符合）；
- 控制器的生命周期简单，**View与控制器一对一**，View只实现它需要的接口（MVP一对多不符合）；
- **符合面向对象**，代码整洁，参数列表清晰（静态工具类不符合）；
- View和Presenter都**不显得过于臃肿**（前述模式都不符合）。

视频详情页重构

1. 存在的问题

2. 解决方案



破窗效应

如果有人打坏了一幢建筑物的窗户玻璃，而这扇窗户又得不到及时的维修，别人就可能受到某些示范性的纵容去打烂更多的窗户。 久而久之，这些破窗户就给人造成一种无序的感觉。

破窗效应

任何一种不良现象的存在，

都在传递着一种信息，

这种信息会导致不良现象的无限扩展，

同时必须高度警觉那些看起来是偶然的、个别的、轻微的“过错”，

如果对这种行为不闻不问、熟视无睹、反应迟钝或纠正不力，

就会纵容更多的人“去打烂更多的窗户玻璃”，

就极有可能演变成“千里之堤，溃于蚁穴”的恶果。

存在的问题

1. 模块中没有传统的Adapter，让人很难理解holder的创建过程
2. holder数据匹配和创建都在Fragment中完成，holder在迭代中产生的新类型急剧扩增
3. 类行数超过2000行
4. Presenters过多，Presenters生命周期无法集中管理
5. View直接与部分Presenter进行交互，脱离面向接口编程思想
6. 太多Presenter的介入，需要Fragment实现多种View接口，产生部分用不到的空实现方法
7. Fragment中嵌套Fragment，而两个Fragment代码都在宿主Fragment中管理
8. Fragment中所有子View的数据都依赖于主Fragment的成员变量，产生强绑定关系
9. 锚点散乱易于出错

解决方案

1.MVP整理

2.Viper架构

3.变量共享域

4.AdapterDelegate

5.锚点梳理

6.注释调整

Viper的使用—基础准备

第1步 创建Presenter

第2步 重写onPresenterCreate()

第3步 重写getPresenter()

Viper的使用

第4步 创建UseCase

第5步 创建Interactor

第6步 在UseCase中处理业务逻辑

第7步 在Presenter中执行UseCase

第8步 创建Router

职责划分

- View职责
- Interactor职责
- Presenter职责
- Entity职责
- Router职责
- UseCase职责
- VarScope职责

View职责

- 各种findViewById；
- 更新UI的方法，暴露更新UI的接口（Toast、Dialog、Fragment、各种自定义View等）；
- Presenter、Router、Interactor的创建；
- View的相关监听（onClick()、onTouch()等）
- View中不应该有Entity和业务逻辑。

Interactor职责

- UseCase单实例懒加载初始化操作；
- 暴露UseCase调用接口。

Presenter职责

- 对View的更新操作（通知弹Toast等）；
- 网络请求的发起，回调处理；
- 数据的监听（数据库监听等）；
- 网络变化的监听等和View变化无关的事件监听；
- 对UseCase的调用、执行；
- 对UseCase的回调的处理，在回调中更新View；
- 对Router的调用，跳转。

Entity职责

- 定义数据结构；
- 提供Getter和Setter。

Router职责

- 调用Activity等进行跳转；
- 暴露跳转接口供Presenter使用。

UseCase职责

- 业务逻辑，数据处理；
- 回调通知Presenter处理。

共享变量职责

- 共享变量层负责进行跨页面的变量传递，以实现页面间的解耦；
- 共享变量层只能包含一般数据，不应该包含View相关的内容；
- 共享变量层跟随Activity等主宿主的生命周期进行变化。

注意事项

1. Presenter对View操作使用getView()，不应将View作为业务Presenter类成员变量，应由基类管理；
2. Presenter对UseCase请使用单实例懒加载方式（注意，非单例，而是单实例，不应加static修饰），在Interactor类初始化，交给Interactor管理。Presenter调用时使用getInteractor()调用UseCase；
3. Presenter调用Router为getRouter()；
4. 同一Activity下跨Fragment变量传递，可考虑使用Bundle或NRVarScope，但不推荐使用Setter；

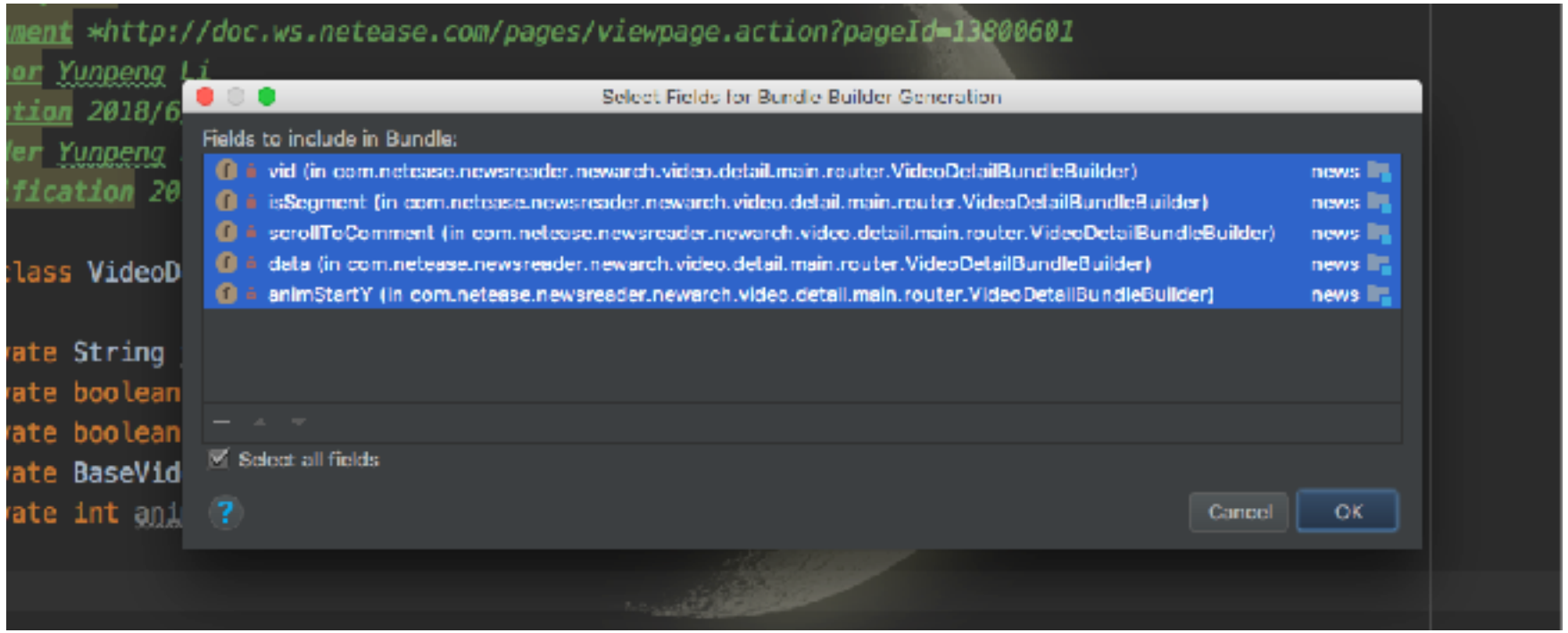
文档

The Clean Architecture 实践

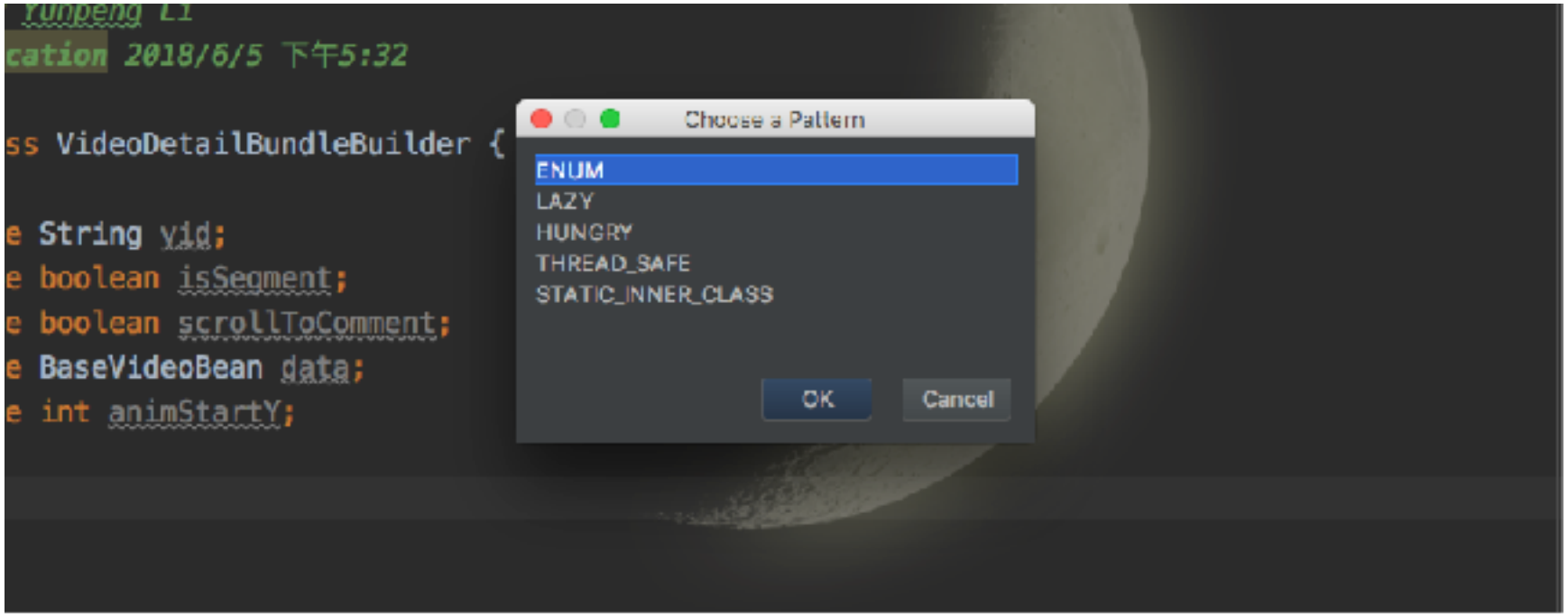
VIPER探索与网易新闻视频详情页重构总结.pdf

附录1 – NR Bundle Builder

Bundle自动生成构造器



附录2 – Singleton



谢 谢!