掌上链家组件化探索历程

DevLink

LIANJIA 链家.com

璩介业

链家网移动端资深工程师

王小虎的程序猿人生

对复杂业务下的架构设计和研发效率提升有浓厚兴趣

为了梦想寻找属于自己的one piece

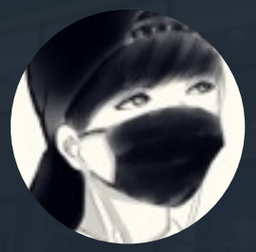
链家网移动端基础架构组

Mobile Architecture Group

PengLev

高级工程师

09年开始一直从事iOS研发工作，积累了丰富的APP开发经验，现负责链家网部分组件重构工作，在开发中提倡精简统一、有迹可循的编程方式，用最简单、高效的Code解决问题



寒哥

高级工程师

16岁第一次coding，基本功扎实，大二开始iOS开发，热爱新技术，功能小王子，技能小王子，Swift爱好者，曾主导丁丁租房进入混编开发。

简书@南栀倾寒

夕才

研发工程师

擅长OC，熟悉iOS、WatchOS开发，懂点JS，曾就职于百度，负责宝宝知道iOS端开发，目前就职于链家基础架构部，负责掌上链家代码的重构和组件化的研究。

大羊驼

研发工程师

C/C++基本功扎实，擅长Objective-C和iOS开发。热爱开源，喜欢新事物，对移动产品有自己的认识和经验。当前在链家网移动端基础架构组，投身于组件化的浪潮之中

内容摘要

业务特点

掌上链家App介绍

频道 * 内容 * 工具

买方服务

二手房　新房　学区房　海外房产

地图找房　旅居地产　成交记录　...

卖方服务

估价　委托卖房　钥匙托管　出租

房屋实堪　专家咨询　销售月报　...

租房服务

普通租房　地铁租房　通勤租房

自如找房　...

掌上链家App特点

App角色众多

涉及的角色：租房用户（Renter）、买房用户（Buyer）、业主（Seller）、经纪人（Agent）

业务模式多样

北京、天津、上海、杭州、青岛等17个城市，各个城市的业务模式不一样（直连、经纪人、第三方代理）

频道和功能模块丰富

有二手房、新房、租房、学区房、海外房产、旅居地产、地图找房、业主委托、钥匙托管、租房、专家咨询、行情分析，买房工具等模块

产品迭代周期短

作为业界房产App的代表，产品两周一迭代，每个迭代需求众多：各条产品线都会提出不同的需求

方案制定

现状和需求

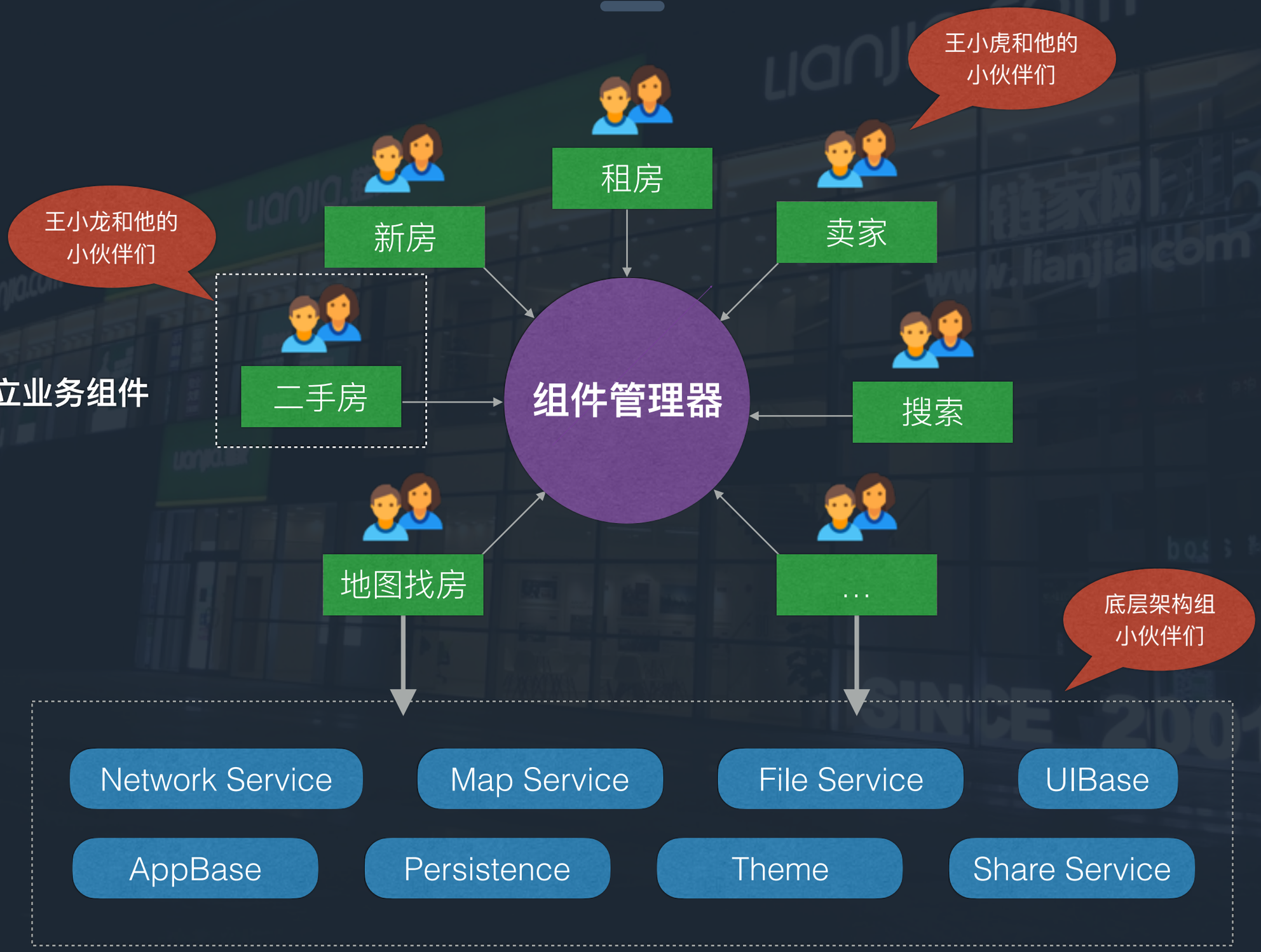
- **代码结构不清晰**
  文件夹分类不清晰，类名没有统一前缀
- **继承关系众多**
  代码耦合性太强，难以扩展
- **模块化程度不高**
  开发分支众多，提交代码冲突频繁，上线前分支合并工作量大
- **重复造轮子**
  公共控件和基础库重复利用率低，为了快速满足产品需求并且不影响现有功能，重复拷贝和继承现象严重
- **代码耦合度高**
  UI展现层、业务实现层和底层公共服务耦合度太高，双向依赖严重

- **统一编码规范**
  建立整个移动端开发的编码规范并有效执行
- **业务模块组件化**
  App按照业务进行组件化拆解
- **开发环境动态切换**
  开发环境、测试环境、生产环境在程序运行状态下动态切换
- **基础服务抽离可复用**
  公共控件和基础库能够从App中抽离，单独做成Pod私有库，避免重复造轮子
- **多团队开发互不影响**
  减少开发分支，尽可能降低不同团队在同一个分支上开发提交代码的冲突

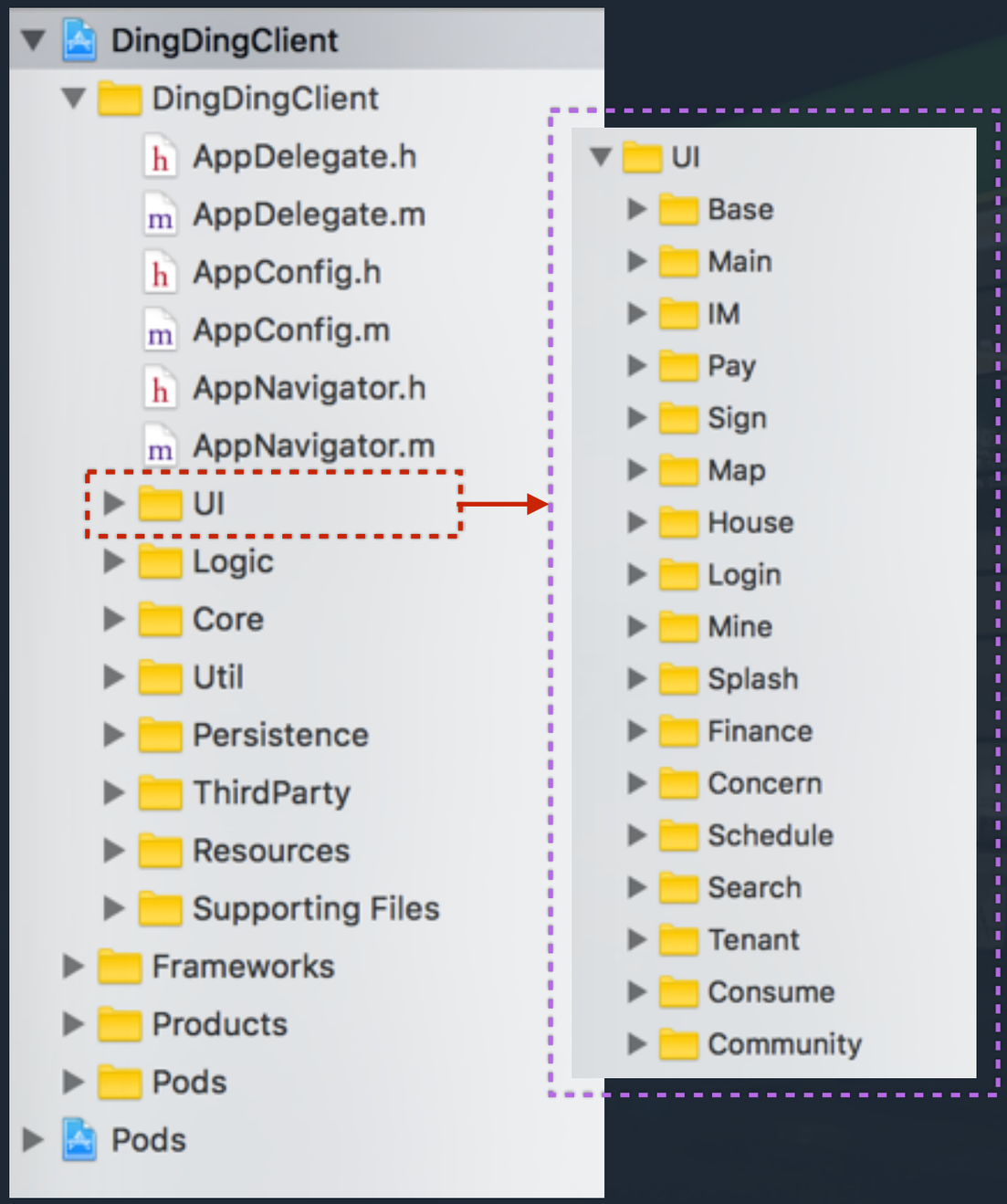
传统的MVC架构


UI
二手房
新房
租房
卖家
搜索
地图找房
...
Model
二手房
搜索
新房
学区房
租房
经纪人
卖家
...
Logic
HouseManager
DataManager
LoginManager
ChatManager
LocationManager
PushManager
CityManager
...
基础服务
Persistence
Network
Log
Resource

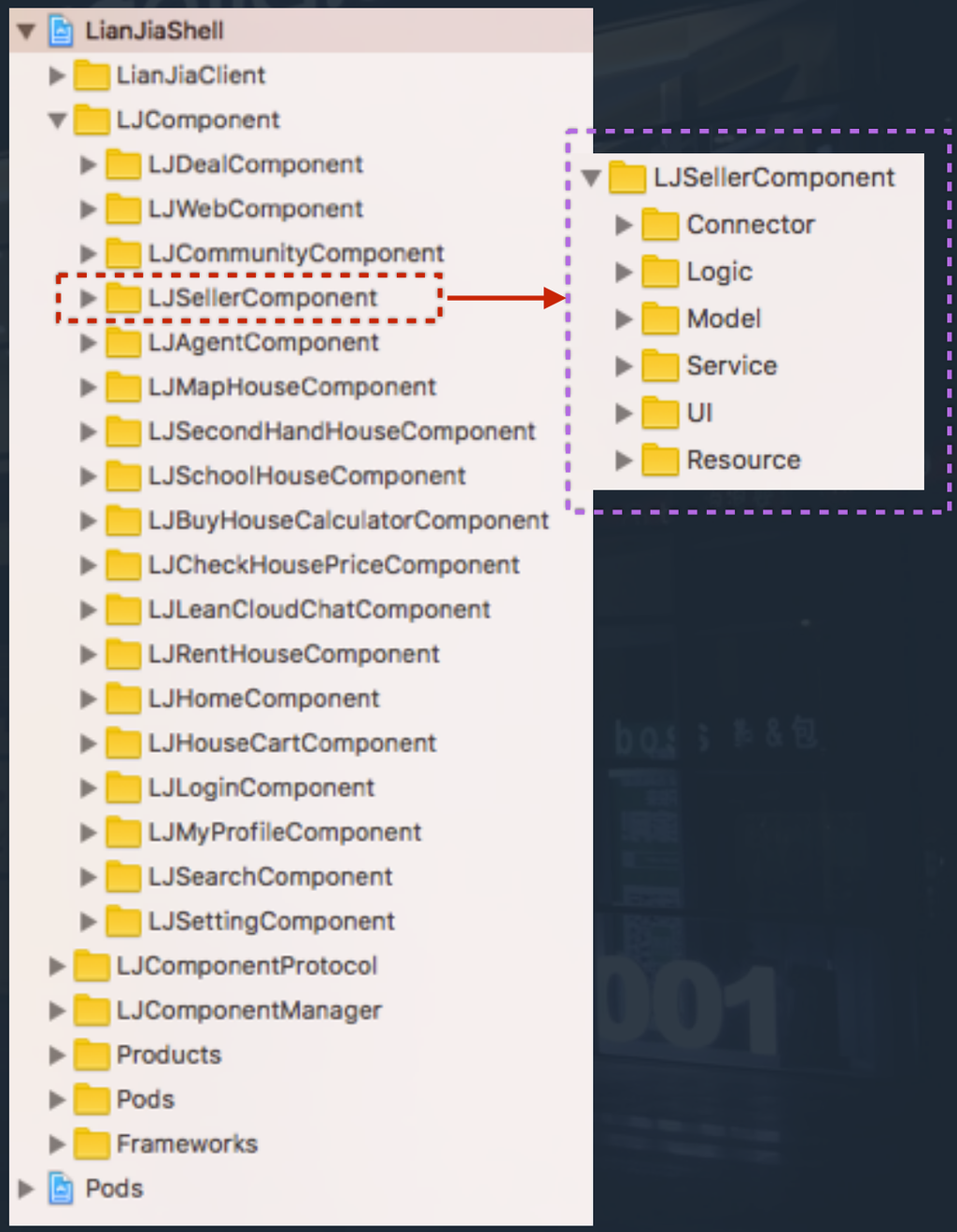
组件化的架构


王小虎和他的
小伙伴们
王小龙和他的
小伙伴们
租房
新房
卖家
独立业务组件
二手房
组件管理器
搜索
地图找房
...
底层架构组
小伙伴们
公共库
基础服务
Network Service
Map Service
File Service
UIBase
AppBase
Persistence
Theme
Share Service

传统架构 vs 组件化架构



VS



步步为营

准备工作

制定代码规范基础服务独立成库

抽离公共基础组件

跨产品使用包括：Log日志、网络、第三方SDK管理、地图服务、社会化分享、文件服务、Category扩展

制定代码规范

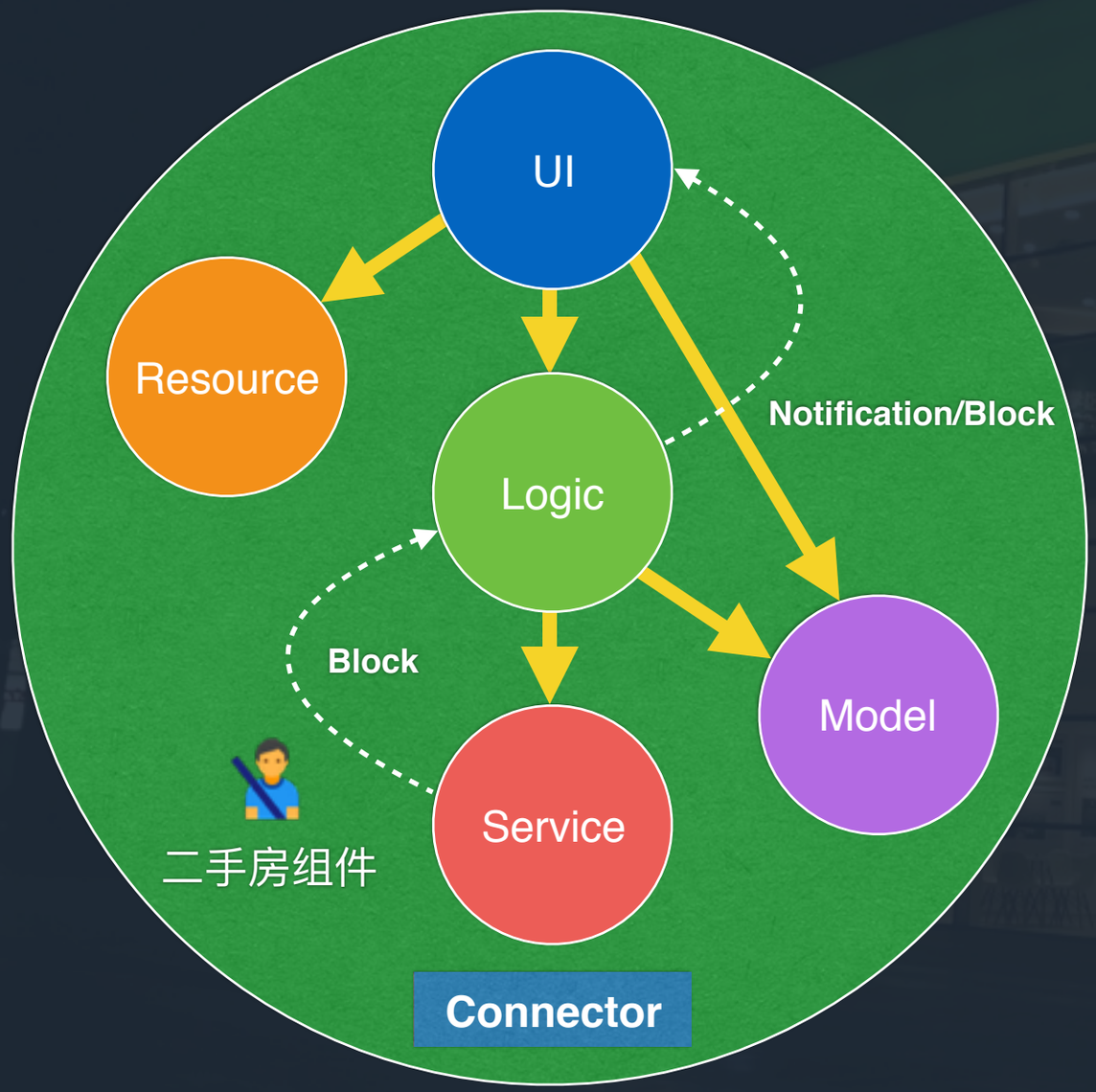
代码风格统一，提高可读性、便于维护

链家产品基础库

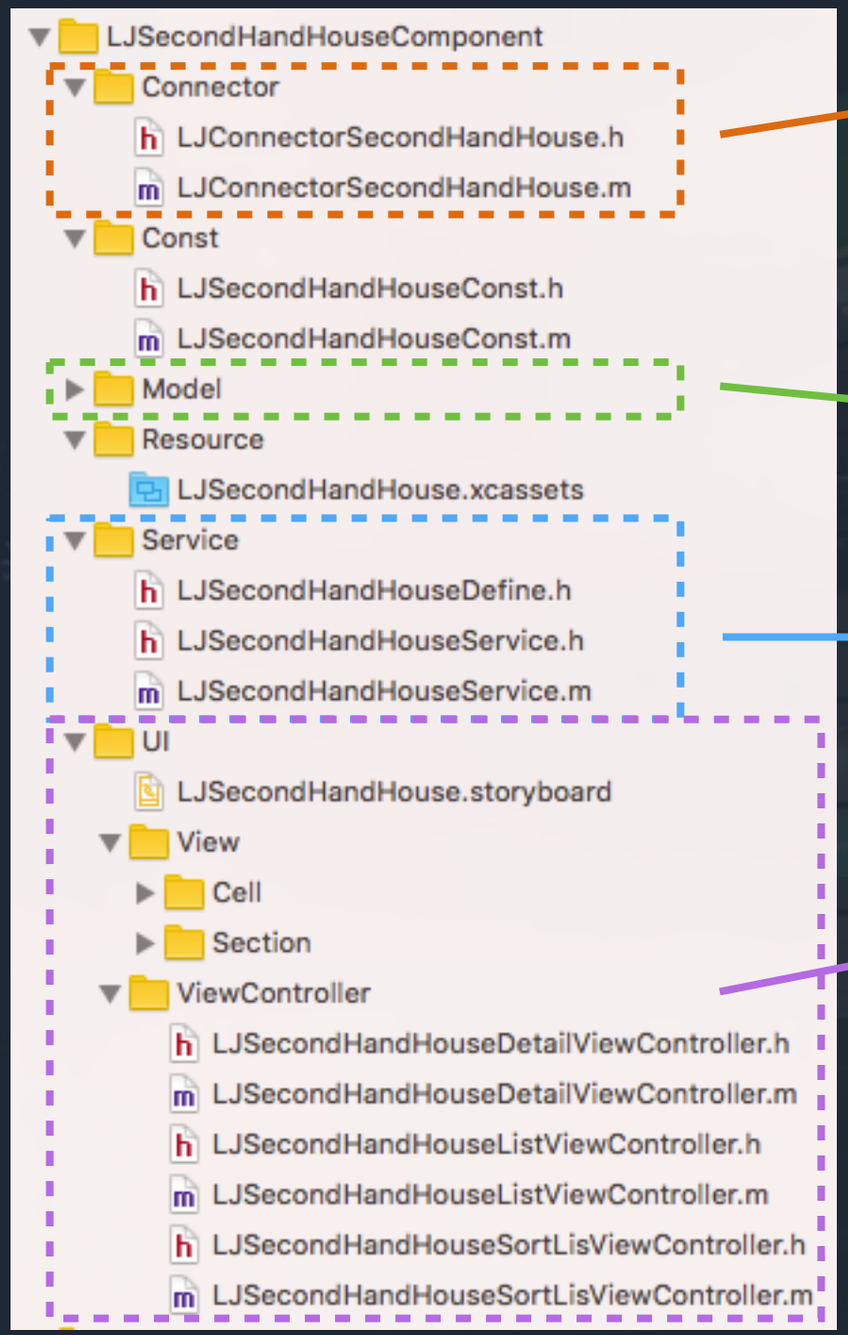
和掌上链家强相关包括：通用UIBase、自定义HUD、图片选择器、图片浏览器、用户行为统计库、应用基本配置信息（token、签名、cookie、加解密相关）以及其他自定义的UI基础组件

初具雏形

业务模块解耦并接入组件管理中心



单个组件示例



接入点

接入点必须遵循LJConnectorProtocol协议，这样才能接入组件管理中心，从而被发现和调度

接入点同时可以遵循该组件对外提供的LJComponentServicePrt协议，这样就可以通过该接入点向外提供组件服务

Model

存放该组件所有内部使用的数据模型，无任何外部依赖

Service

服务层集中处理该组件的网络接口请求以及网络回调回来后的数据转model工作

UI

页面全部存放在该组件对应的storyboard中，便于查看整个组件的业务流程，界面的事件处理放在对应的ViewController中

Logic

如果组件中牵涉到复杂业务逻辑，需要保存状态机或者处理各种复杂业务情况，可以放在这边处理：例如登录组件的LJLoginManager

组件之间页面跳转

URL导航去中心化

接入点协议

只要符合接入点协议：LJConnectorProtocol的组件都可以接入组件管理中心，具体的接入方式是在每个接入点的Load方法将自己注册到组件管理中心

```
/**
 *  接入点协议
 */
@protocol LJConnectorProtocol <NSObject>

@optional

// 当前业务组件可导航的URL判断
- (BOOL)canOpenURL:(NSURL *)URL;

// 业务组件注册自己能够处理的URL，返回对应的vc
- (UIViewController *)handleURL:(NSURL *)URL
                         params:(NSDictionary *)params;

// 业务组件注册自己提供的service，返回服务实例
- (id)handleService:(Protocol *)servicePrt;

@end
```

LJComponentManager

组件管理器通过服务发现的方式对登记在册的组件进行遍历，找到传入URL对应的viewcontroller通过LJNavigator进行路由跳转

LJConnectorNewHouse

业务组件注册自己能够处理的URL，返回对应的vc

LJConnectorMapHouse

如果某个url无法处理或参数不对，直接返回一个nil

LJConnectorMyProfile

业务组件无需care如何进行页面跳转，只需返回对应vc

组件之间服务调用

业务组件对外服务协议集合

对外服务协议集合

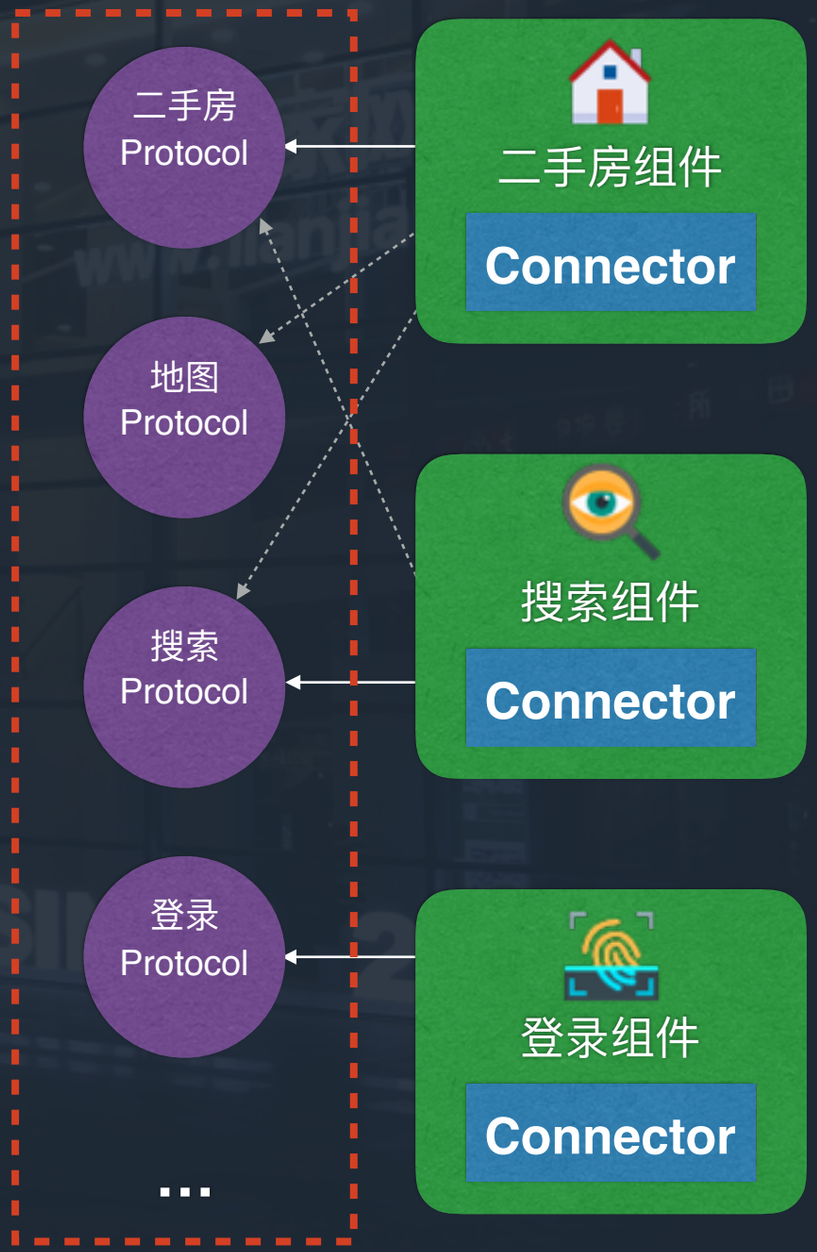
每个业务组件将自己对外提供的服务接口（例：LJComponentHomeServicePrt）抽象到一个统一的业务组件协议集合中， 业务组件对外提供的能力依赖自己的对外暴露的服务接口

对外服务协议集合

业务组件的版本和其对外提供服务的协议版本要保持一致

单独设立协议版本仓库

组件协议集合单独通过一个Git地址进行管理，单独配置podspec，单独通过协议的版本仓库进行管理



组件之间通讯示例

```
// 打开二手房列表页（默认以push的方式）

[LJComponentManager handleURL:@"lianjia://ModuleSecondHandHouseList"];


// 打开二手房详情页，并传入相应的参数

[LJComponentManager handleURL:@"lianjia://ModuleSecondHandHouseDetail"]
withParameters:@{@"houseCode":  model.houseCode}];


// 获取Home组件提供的服务：获取当前App的基础配置数据

[[LJComponentManager handleService:@protocol(LJModuleHomeServicePrt)]
getAppBaseConfigInfo];
```

更进一步

URL动态部署和容错处理

lianjia://ModuleOverSeaHouseList?url=http://m.lianjia.com/nations

lianjia://ModuleSchoolHouse?url=http://m.lianjia.com/school

....

公共Model下沉

二手房组件　卖房组件　新房组件　...

公共Model

- 房子
- 搜索
- 经纪人
- ...

- **单独设立公共Model仓库**

  公共Model通过一个Git地址进行管理，是一个独立的私有仓库，没有任何依赖

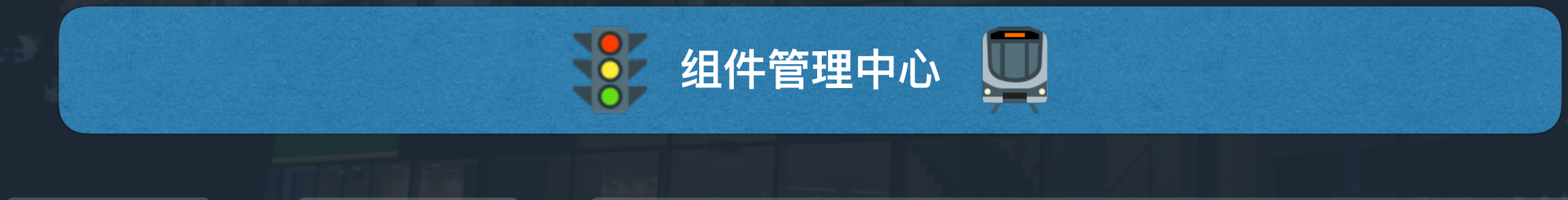
- **严格的版本管理**

  公共Model仓库对所有组件可见，因此需要严格的版本管理，新增和修改必须同步所有相关业务方并且内部不允许有任何依赖和继承

掌上链家组件化架构


二手房组件
地图组件
SALE
卖房组件
登录组件
新房组件
设置组件
...


二手房 Protocol API
地图 Protocol API
卖房 Protocol API
登录 Protocol API
新房 Protocol API
设置 Protocol API
... Protocol API


组件管理中心
基础服务
网络
日志
文件
...
Model
房子
搜索
经纪人
...
管理器
LocationManager
DataManager
VersionManager
PushManager
CityManager
AdvertisementManager
ContactManager
...
Persistence
Navigator
UIBase
AppGlobal
Theme

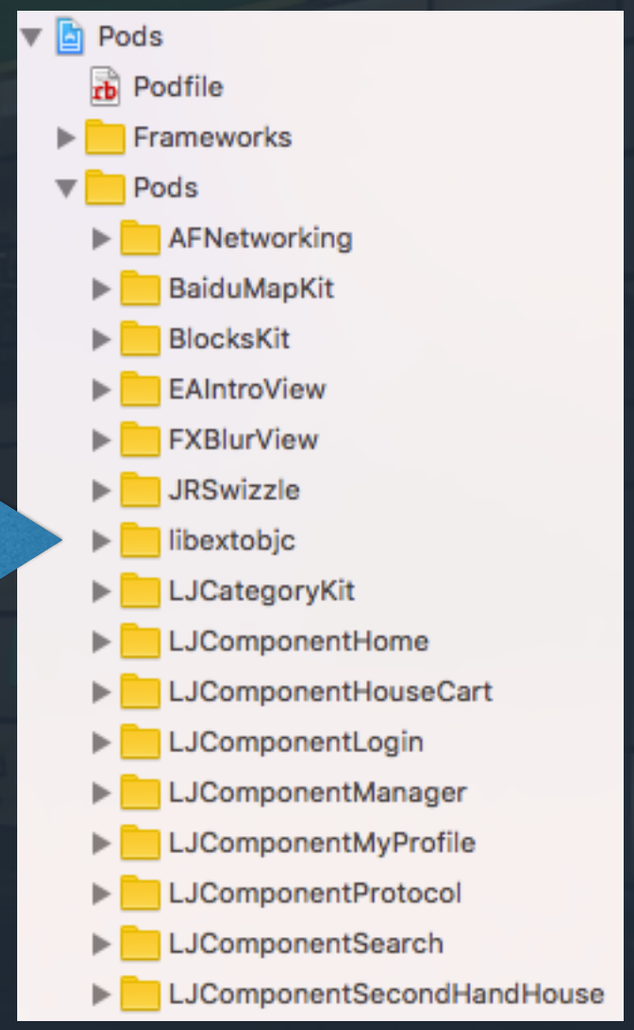
工程组织方式

Pod file

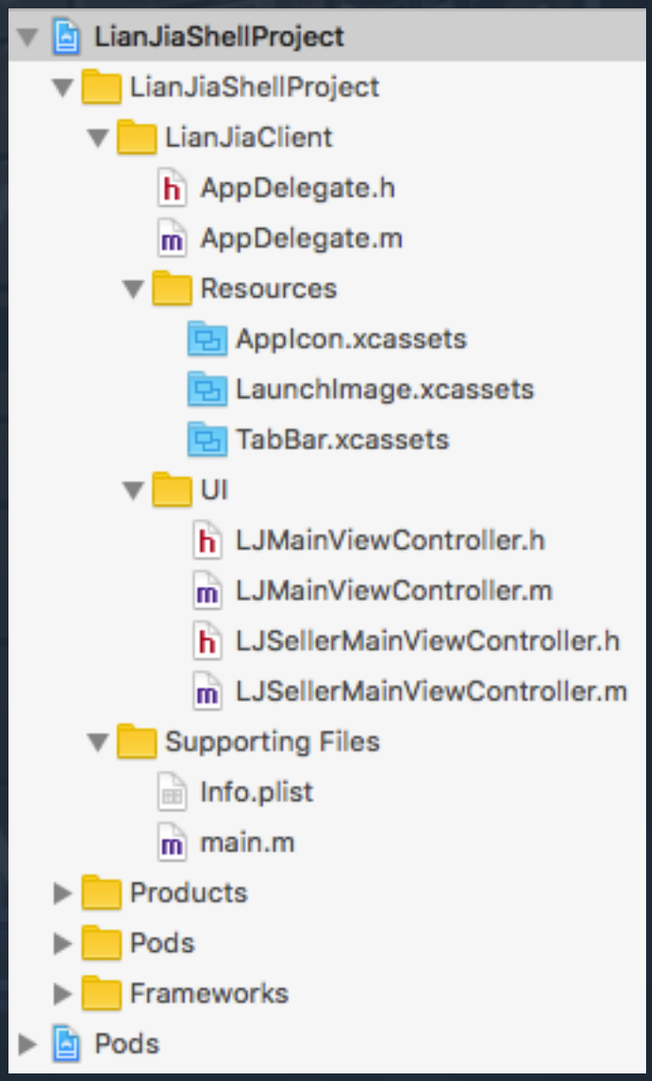
```
target 'LianJiaShellProject' do

    # 基础服务
    pod 'LJService'
    # 持久化
    pod 'LJPersistence'
    # App全局配置
    pod 'LJAppGlobal'
    # 基础类扩展
    pod 'LJCategoryKit'
    # 主题管理器
    pod 'LJTheme'
    # 页面导航器
    pod 'LJNavigator'
    # 组件化中心管理器
    pod 'LJComponentManager'
    # 首页组件
    pod 'LJHomeComponent'
    # 二手房组件
    pod 'LJSecondHandHouseComponent'
    # 搜索组件
    pod 'LJSearchComponentManager'
    # 购物车组件
    pod 'LJComponentHouseCart'
    # 登录组件
    pod 'LJComponentLogin'
    # MyProfile组件
    pod 'LJComponentMyProfile'

end
```



掌上链家壳工程



披荆斩棘

遇到的问题

组件的拆分粒度

组件拆分多细，拆到哪一层级合适，需要和具体业务相结合，并且在实际开发的过程中不断的调整和变化

不同组件中资源重复

一些公共的图片，例如：导航栏icon，错误提示图片等可以打成一个私有Pod。每个组件管理自己的资源，不能依赖其他资源，就算重复拷贝也没太大关系

Objective-C和Swift混编

Swift库必须通过动态链接库引入，开启use_frameworks!选项之后，所有以源码引入的pod都会编译成动态链接库，如果此时源码pod又依赖fake framework，CocoaPods会提示下面这个错误：transitive dependencies that include static binaries: (xxx.framework, xxx.framework)

同一工程中多个组件协同开发

所有组件在一个Development工程中按照文件夹区分的方式进行协同开发，各组件的开发负责人保证不引用其他组件的头文件，杜绝pch文件的使用，开发完成后每个组件编写自己对应的pod spec文件并上传至私有仓库中，以供壳工程install使用

一些建议

- **少用或者不用宏**

  字体大小、文字颜色、margin、padding等值不要用宏，统一使用Theme组件去处理或者直接写死固定值

- **相同cell和控件的复用**

  组件之间如果用到同一个cell，建议多份拷贝，不要沉底或者单独为这个cell做一个Pod

- **使用storyboard设计界面**

  使用storyboard的好处是：界面开发快并且容易维护

- **不允许出现Common组件**

  单个组件一定要求分工明确，功能单一

总结

经验分享

- 同步产品经理并且协调好测试资源
- 制定一个编码规范并且让大家都遵守
- 处理好组件化和现有产品迭代的关系
- 选择一个适合自己项目的组件化方案

推荐资料

- **模块化与解耦**

  https://blog.cnbluebox.com/blog/2015/11/28/module-and-decoupling/

- **iOS应用架构谈，组件化方案**

  http://casatwy.com/iOS-Modulization.html

- **iOS组件化实践方案－LDBusMediator炼就**

  http://www.jianshu.com/p/196f66d31543

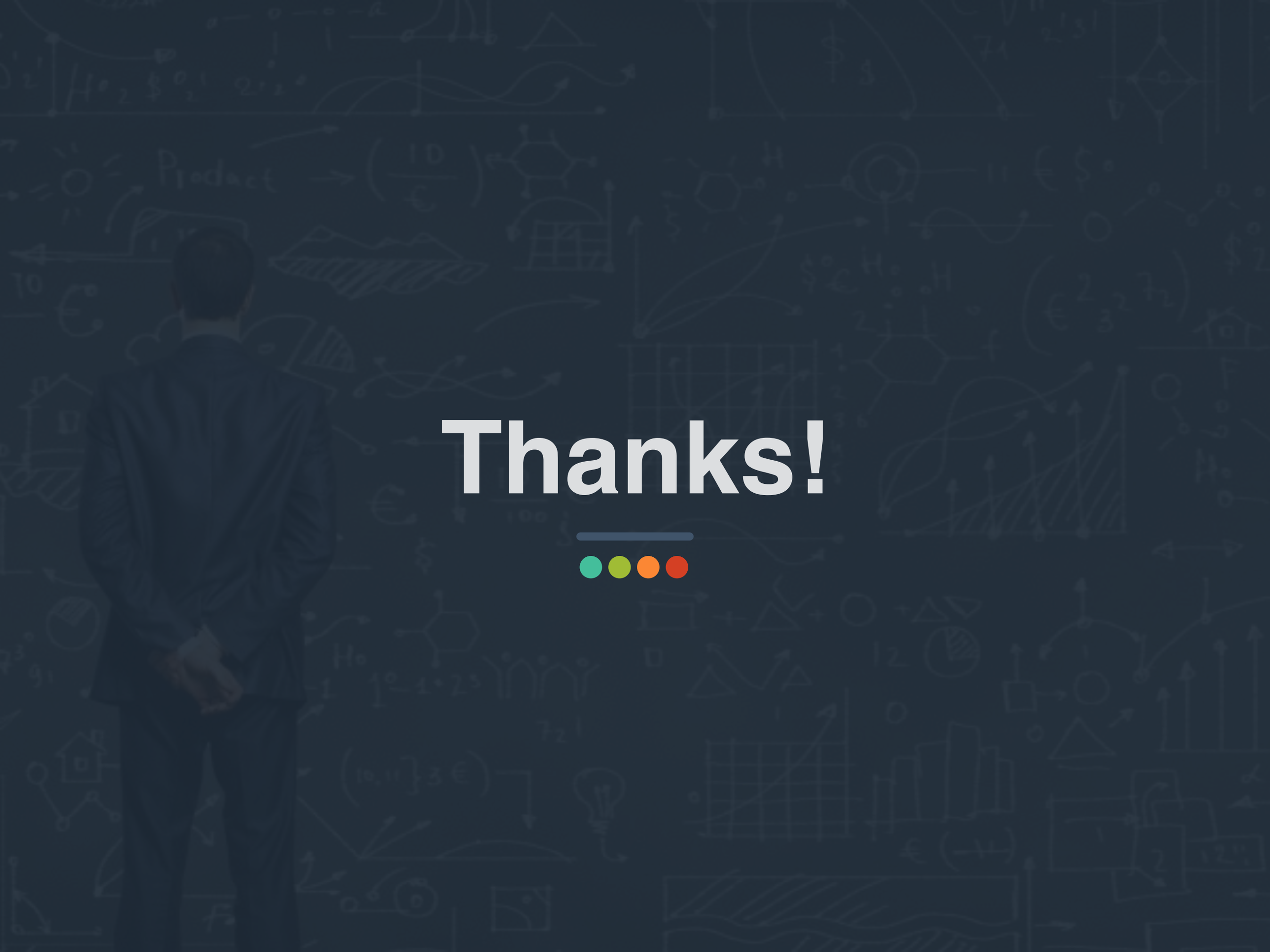
- **解耦神器-统跳协议和Rewrite引擎**

  http://pingguohe.net/2015/11/24/Navigator-and-Rewrite.html

- **寒哥细谈之AutoLayout全解**

  http://valiantcat.com/2015/10/14/LearnAutoLayout/

Q&A



Thanks!