内容型 App 的客户端架构

范怀宇 @ 豌豆荚

个人介绍

- 2007 年开始接触 Android，写相关的技术文章，并著有《Android 开发精要》一书
- 2009 年加入网易有道，专注于 Symbian 和 Android 等移动平台的开发
- 2011 年加入豌豆荚，现负责新产品的研发工作

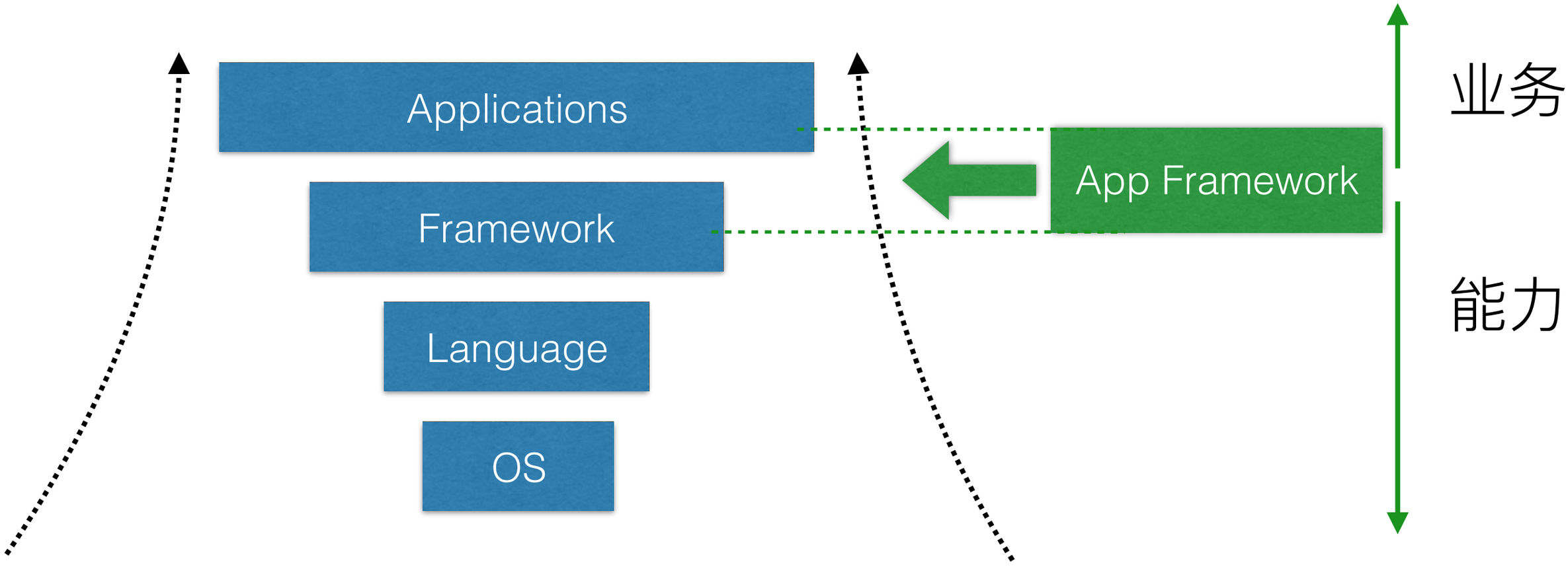
背景

- 豌豆荚，专注在移动内容和功能的发现上
- 围绕着这个目标，会设计和研发一系列的相关产品来满足相关的用户需求（豌豆荚 5.0，豌豆荚一览，Snpalock 效率锁屏...）
- 业务领域是稳定的，但产品形态需要持续探索

挑战

- 如何来加速新产品的研发和老产品的迭代?
- 如何提升运营效率?
- 如何确保设计语言能够精准落地?
- 如何保证客户端具有流畅的交互体验?
- 如何构建统一的数据体系?

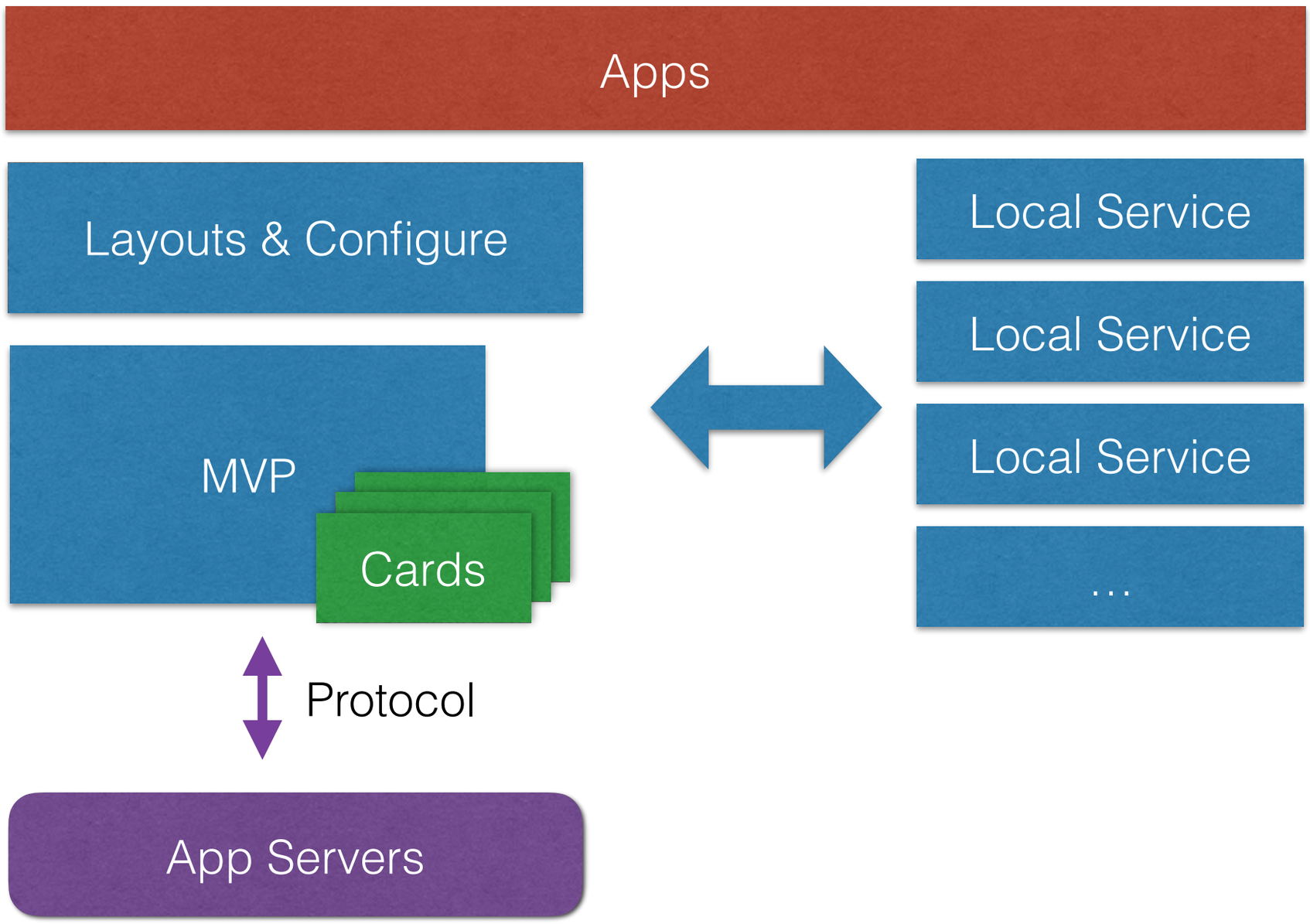
移动架构



“Content Browser”

- **Browser = Html (with link) + CSS + JS**
  - 数据、样式、行为
- **Content Browser（Nirvana）**
  - Html：基于统一数据协议，基于 Intent 进行页面跳转
  - CSS：基于统一设计语言的、模版化的卡片框架，可以按照不同的 Layouts 进行组织
  - JS：本地服务化框架
- **数据、样式、行为相互隔离：行为预设置、样式模版化、数据可运营**

Nirvana 架构


Apps
Layouts & Configure
MVP
Cards
Protocol
App Servers
Local Service
Local Service
Local Service
...

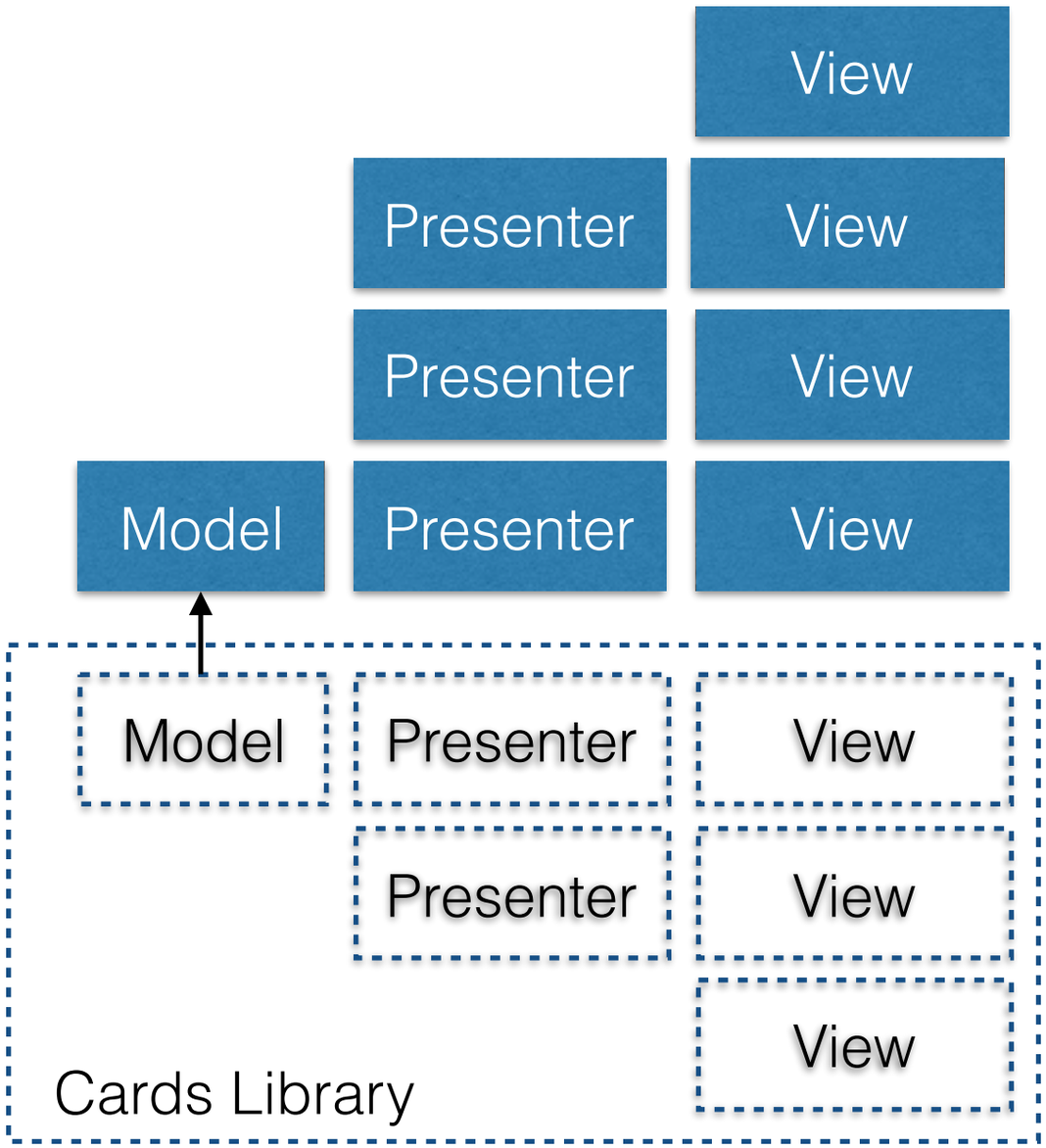
数据和样式

MVP

- **Model - View - Presenter**
  - 分层隔离
  - 绑定数据和变量
- **Android 上的实现**
  - Finsky for Google Play Store
  - Leanback for Android TV
  - Data Binding Library for Android

One Model

- **不同的数据抽象成同样一个的数据模型**
  - 应用、视频、音乐、场景、富文本 …
  - Model 中包含界面上需要呈现的内容、版式、和动作
  - 列表用概要数据，详情用明细数据
- **优点：极大的可复用性**
- **缺点：抽象会变得繁重**



数据源

- **统一的数据源（Data Provider）**
  - 本地和网络统一
  - 所有数据源都返回 Model 列表
- **将复杂的数据获取细节封装到数据源内**
  - 基于 Volley 实现的 Http 数据源
  - 客户端/服务端可配置的缓存模型，支持软硬过期
  - 支持各种翻页策略
  - …

卡片

- **界面层的复用单元是卡片**
  - 卡片其实就是一个界面 Component
  - 卡片中的控件没有实现要求，仅需遵循 ID 规范
    - 每个卡片一个版式 ID（Template ID），定义了样式、排版、动画
- **Model 中仅包含所需的 Template ID，几乎不包含其他的排版信息**
- **优点：样式和数据完全分离，简化运营**
- **缺点：不能调整 Template 的样式细节**
  - 插件化
  - 基于动态 Xml、Html 的卡片

Card Presenter

- **原子化的 Presenter**
  - 每个 Presenter 可以控制一个或一组 ID 的特定属性
  - 一个控件的一个属性，只能有一个 Presenter 来处理
- **每个卡片的 Card Presenter，有一组 Presenter 组合实现**
  - 组合成的 Card Presenter 是最小复用单位
  - 可以独立使用，也可以在列表中使用
  - 每个 Card Presenter 中可以固化一些 Presenter，比如数据统计
- **优点：既满足细粒度的复用，又简化了使用方式**
- **缺点：Presenter 的设计需要正交，有一定复杂度**

操作和事件

- **Presenter 统一处理和卡片相关的变化**
  - 点击等主动操作
  - 数据的变化等被动变化
- **Presenter 的 Bind/Unbind 必须成对调用**
  - Bind 中进行事件的订阅，Unbind 中进行取消
  - 将 Presenter 池化管理，并在组件生命周期函数中统一处理

排版

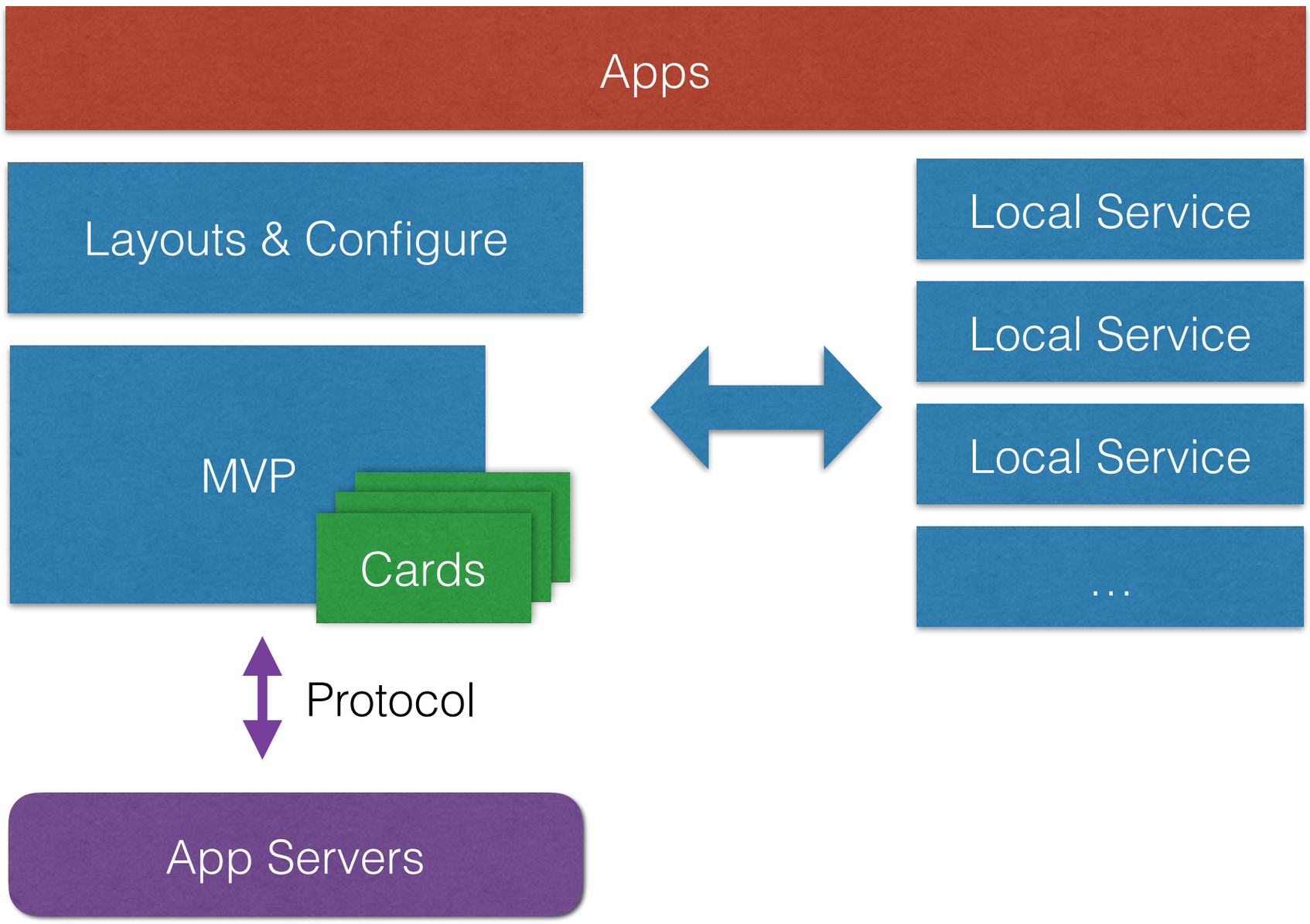
- **卡片基于排版（Layouts）进行组装**
  - Layouts 也是通过 Model 进行定义
  - 基于 RecyclerViews 实现的列表，可动态配置的 Tab Layouts，等等
- **One Activity**
  - 应用可配置生成
  - 页面样式、卡片样式、跳转策略均可完全由 Model 决策，可运营性好
  - 大部分的场景，可以通过 Intent 传递给 Activity 完成环境配置，并通过 PageContext 对象把环境配置传递给页面和卡片
  - 对于特殊的页面结构和需求，可以通过重载实现

总结

- **可配置生成 Web 1.0 型的 App**
  - 统一数据模型
  - 卡片化的界面框架
  - 可定制的排版
- **Web 2.0 型 App?**

行为

Nirvana 架构


Apps
Layouts & Configure
MVP
Cards
Protocol
App Servers
Local Service
Local Service
Local Service
...

服务化

- **预设各类服务**
  - 收藏、分享、统计、下载、支付 …
  - 不同应用可以选配或增改
  - 面向接口
- **动态查询**
  - Android System Service
- **优点：简单可依赖**
- **缺点：无法动态变更行为能力**
  - Web 化
  - 插件化

接口设计

- **服务面向接口**
  - 可随时变更实现
- **接口设计没有物理约束**
  - 无需继承其他接口
- **调用模型规范化**
  - Google Play Service（GMS）
  - 所有接口提供同步调用
  - 异步事件基于 EventBus

答案

- **如何来加速新产品的研发和老产品的迭代?**
    - Nirvana 框架
    - 框架提供 80 分能力，应用层专注于剩下 20 分
- **如何提升运营效率?**
    - 通过统一的数据模型，实现页面排版、卡片样式、跳转策略、内容数据均可配置
- **如何确保设计语言能够精准落地?**
    - 统一的卡片库
- **如何保证客户端具有流畅的交互体验?**
    - 基于统一卡片库、排版样式的集中性能优化
- **如何构建统一的数据体系?**
    - 基于统一卡片库可以动态的和本地服务关联，实现统一的统计策略
    - 不同应用，可以用不同的统计服务

Thanks

Q&A