

# 愉悦行Android客户端组件化演进之路

June Yang

# 目录

1. Android 十年技术变化
2. Android开发技术特点
3. Android客户端组件化实践
4. 如何实现组件化
5. 未来开发技术方向

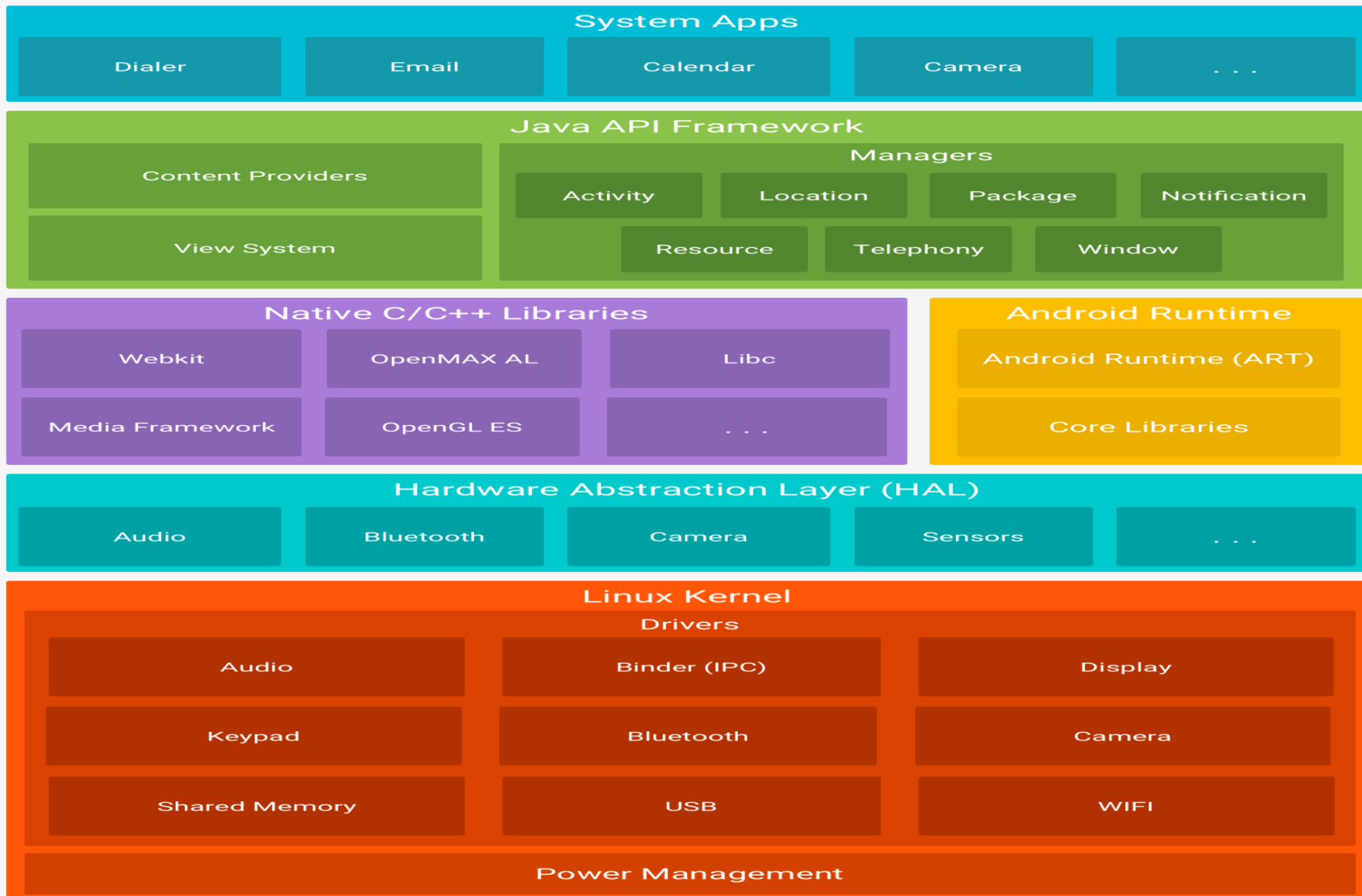
# Android的前世今生

- 2008.9月 Android 1.0 (初出茅庐)
- 2009.4 Android 1.5 (技能提升)
- 2009.9 Android 1.6 (初露锋芒)
- 2012.2 Android 2.2 Android与Linux开发主流分道扬镳
- 2013.11 Android 4.4 (重大版本)
- 2014.10 Android 5.0 (Material Design设计风格)
- 2015.5 Android 6.0 (运行时 (动态) 权限申请)
- .....
- 2018.8 Android 9.0 最新稳定版
- 2019.3 Android Q Beta 1 支持可折叠屏幕

# Android系统架构图（初始版）



# Android系统架构图（完善版）



# 了解系统架构意义

- 若是从事Android应用开发，那应该研究Android的应用框架层和应用程序层；
- 若是从事Android系统开发，那应该研究Android的系统库和Android运行时；
- 若是从事Android驱动开发，那应该研究Android的Linux内核。

Android开发语言

Java → Kotlin



# Kotlin 代码是什么样的？

KOTLIN

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        ...  
        fab.setOnClickListener { view ->  
            Snackbar.make(view, "Hello $name", Snackbar.LENGTH_LONG).show()  
        }  
    }  
}
```

Nullable and NonNull  
types help reduce  
NullPointerExceptions

Use lambdas for concise  
event handling code

Use template expressions  
in strings to avoid concatenation

Semicolons are optional

# 为什么选择Kotlin?

Kotlin 解决了一些 Java 中的问题:

- 空引用由类型系统控制
- Kotlin 中数组是不变类型的
- Kotlin中没有受检异常
- .....

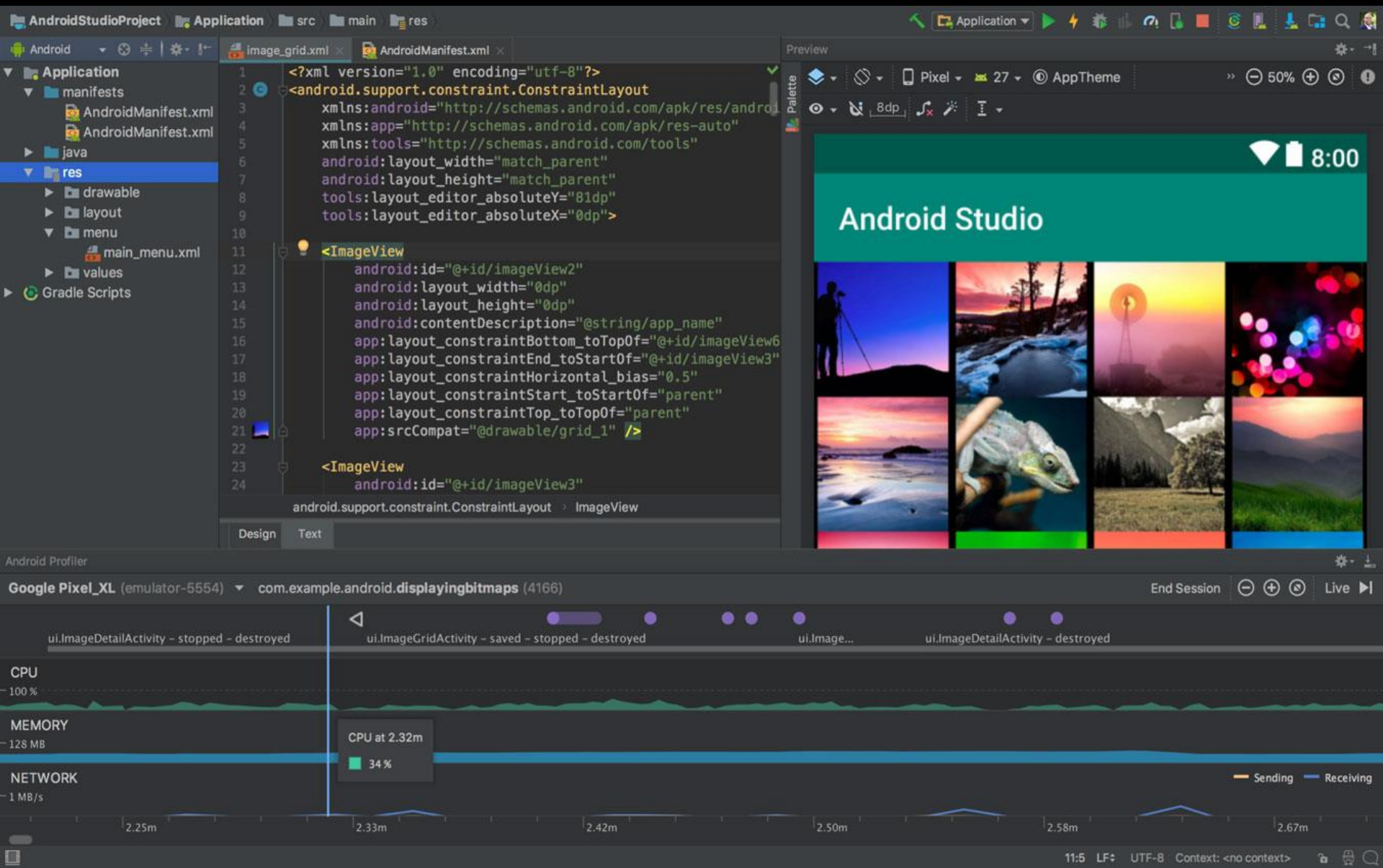
# Kotlin 有而 Java 没有的东西

- 扩展函数
- 智能类型转换
- 变量与属性类型的类型推断
- 字符串模板
- 单例
- 区间表达式
- 空安全
- 协程
- .....

Android开发工具

Eclipse → Android Studio

# Android Studio是什么样子的？



# 以前的App





# 现在的App



下拉刷新

快速开发框架

侧滑菜单

Material Design

模块化

Volley、OKHttp

热修复

插件化

# 我们做了什么

RN、WEEX、Flutter

MVP、MVVM

Hybird



## 现在的 APP

用户中心

列表页/详情页

业务 N

推送

路由

UI统一

广告

RN/WX

数据上报

异常统计

性能监控

配置下发

网络库

图片库

跨进程

音视频

Kotlin

MVP、VM

国际化

模块化

热修复

打包平台

# 渝悦行Android客户端组件化实践

无服务 424 B/s

424 B/s 100% 17:57

21°C 不宜洗车  
不限行



停车缴费



千里眼



实况导航



共享车位



停车场



年检



违章



驾驶证



车险



4S店保养



一键救援



全部



驾驶证

03-18

恭喜您！成功添加了啦啦啦的驾驶证



年检消息标题

03-08

列表：有车



车辆估值

为您的爱车 川A9WX92 估个值吧



首页



服务



导航

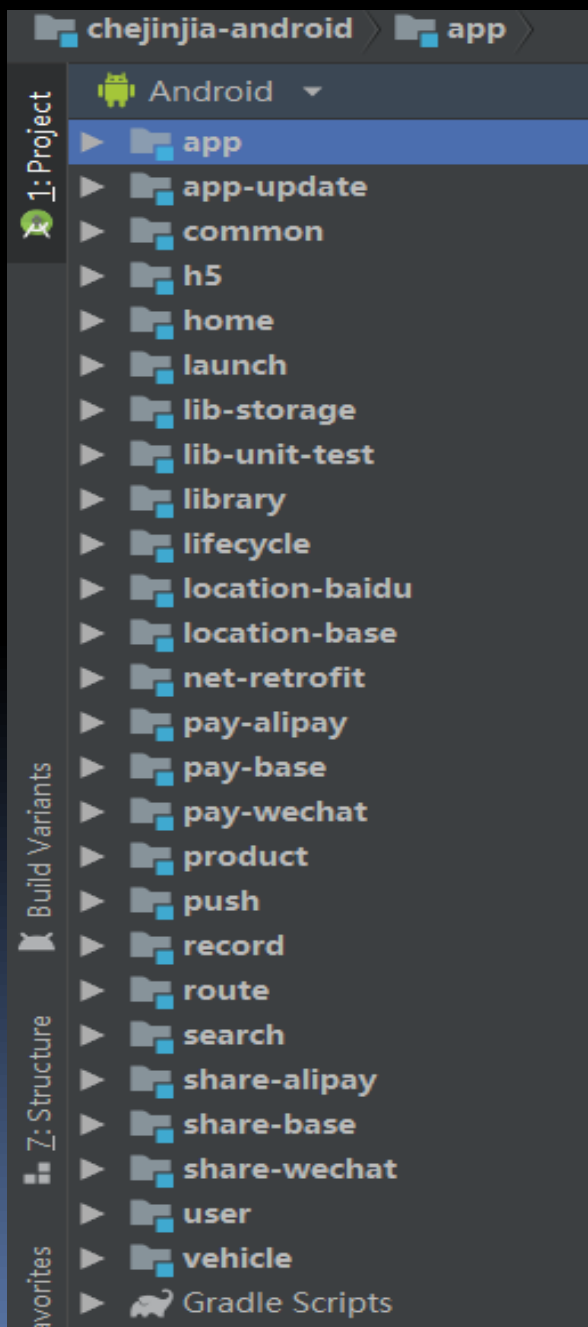


停车缴费



我的

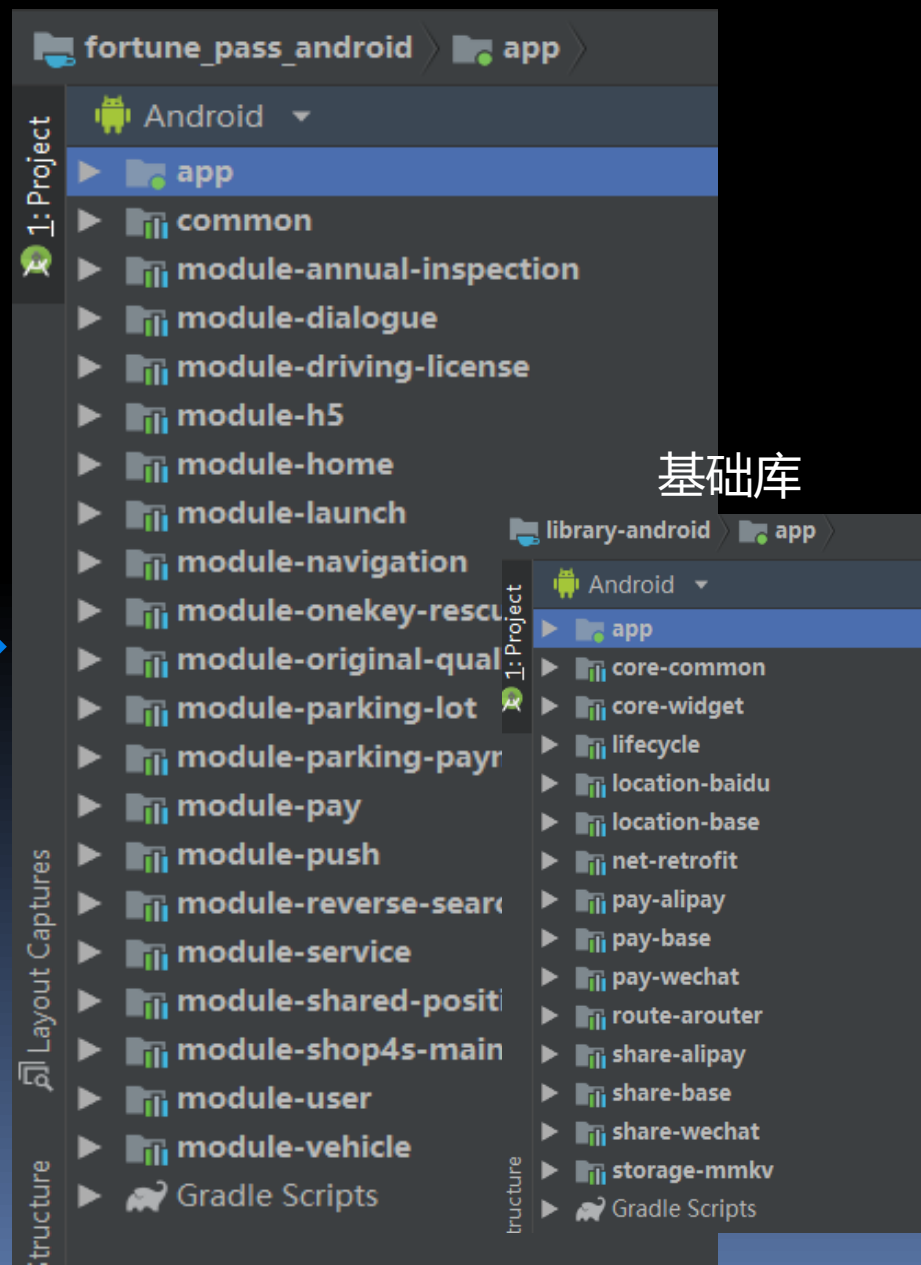
## 车金家Android客户端



重构后



## 渝悦行Android客户端



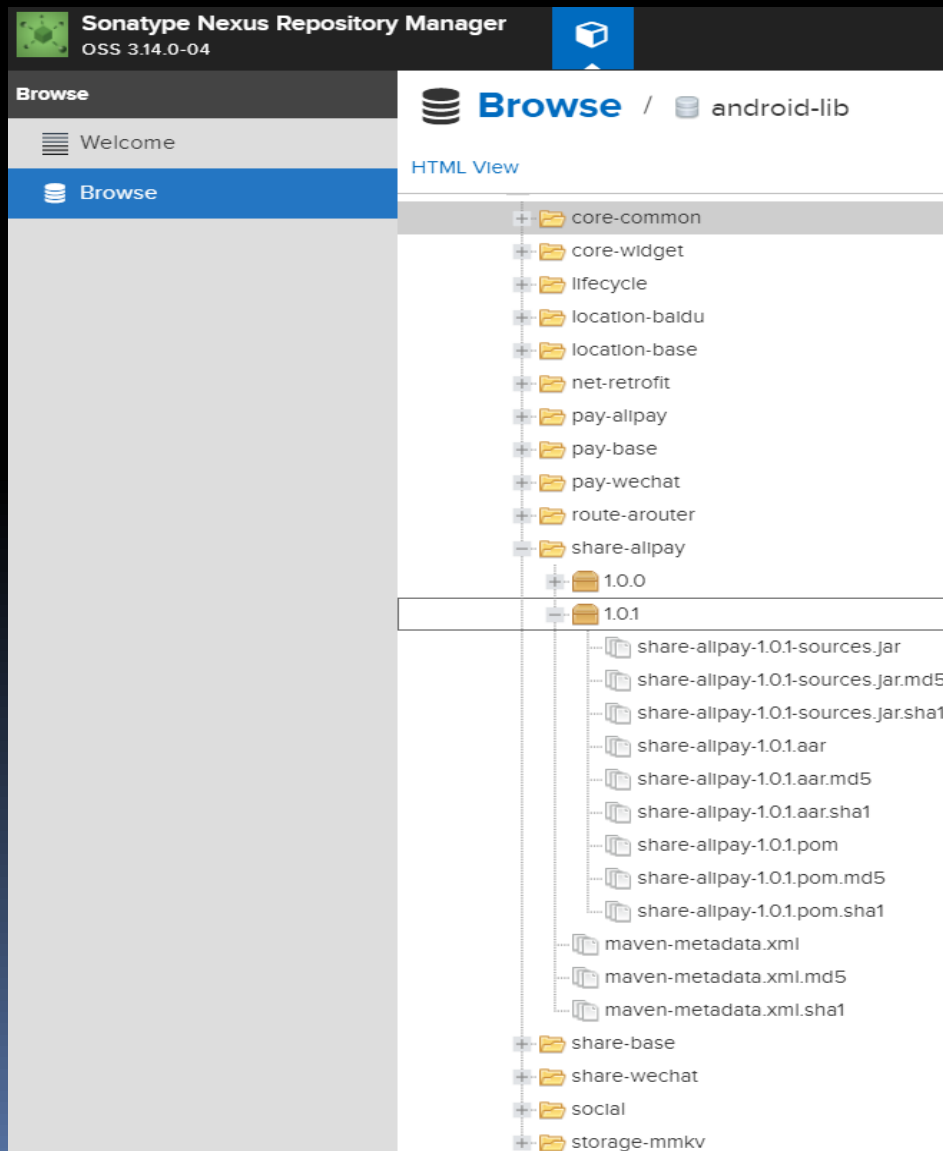
# 为什么要组件化？

复用

解耦

效率

# 组件化流通



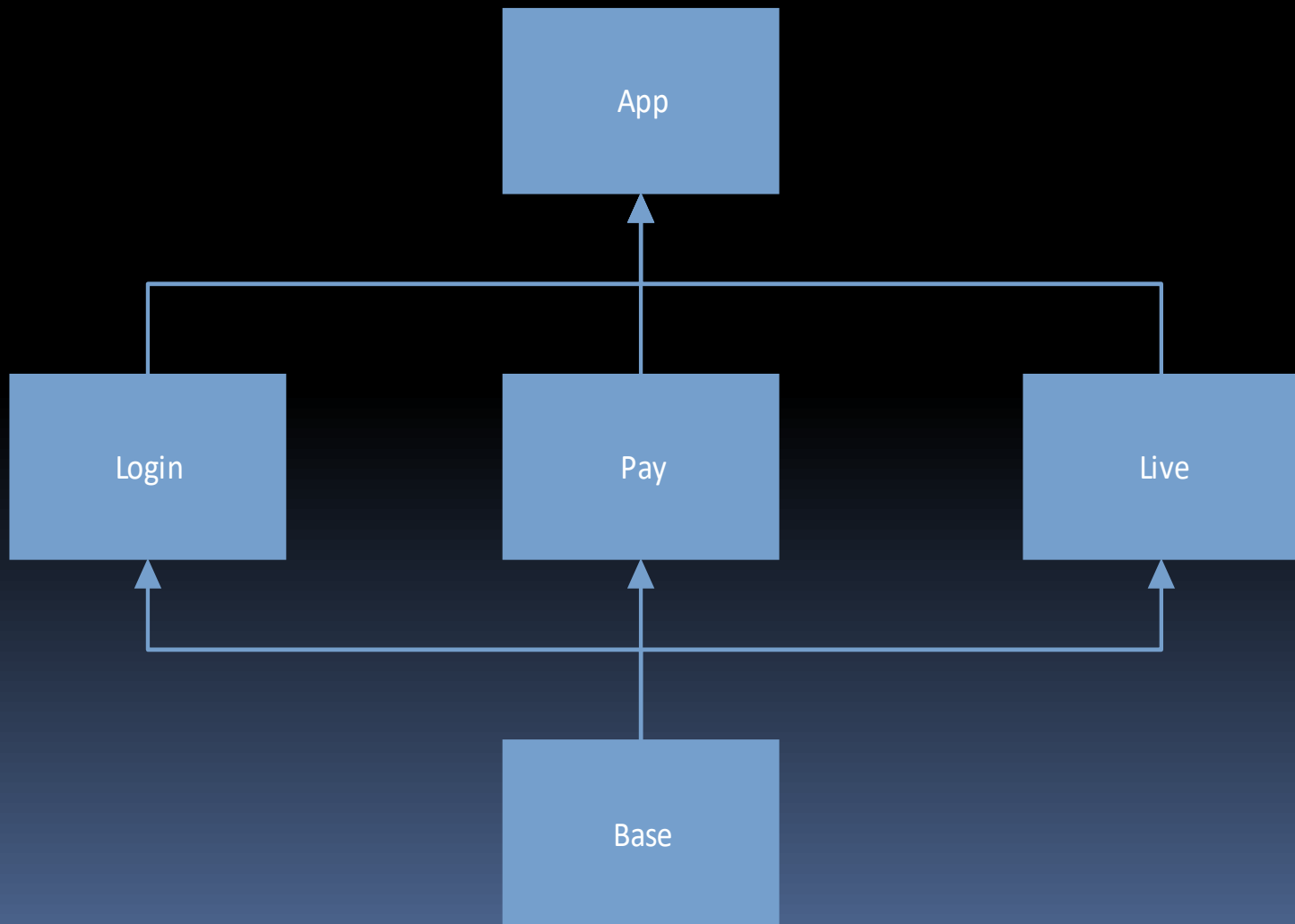
# 如何实现组件化

# 基础的组件化架构

应用层

组件层

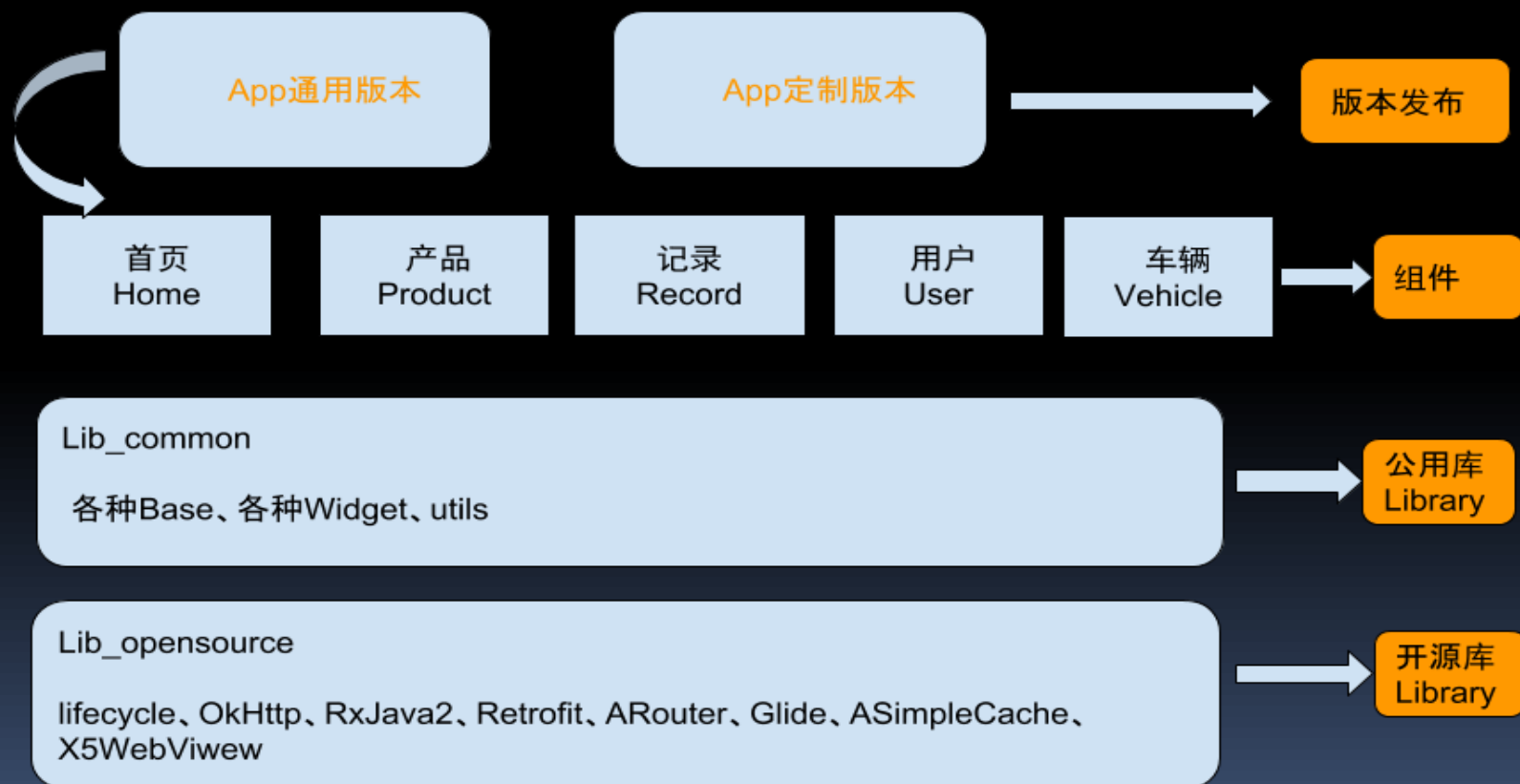
基础层



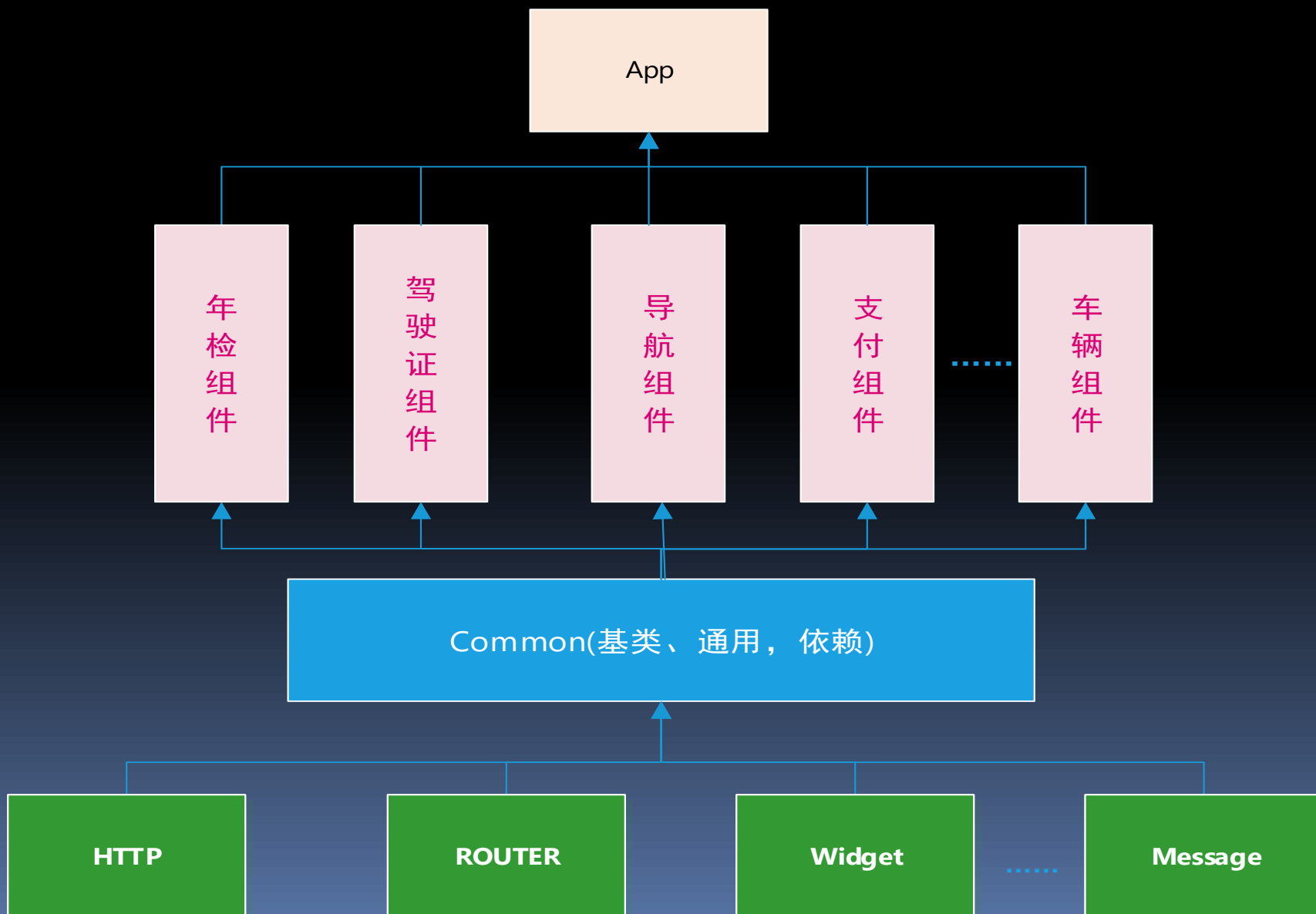


# 项目架构图

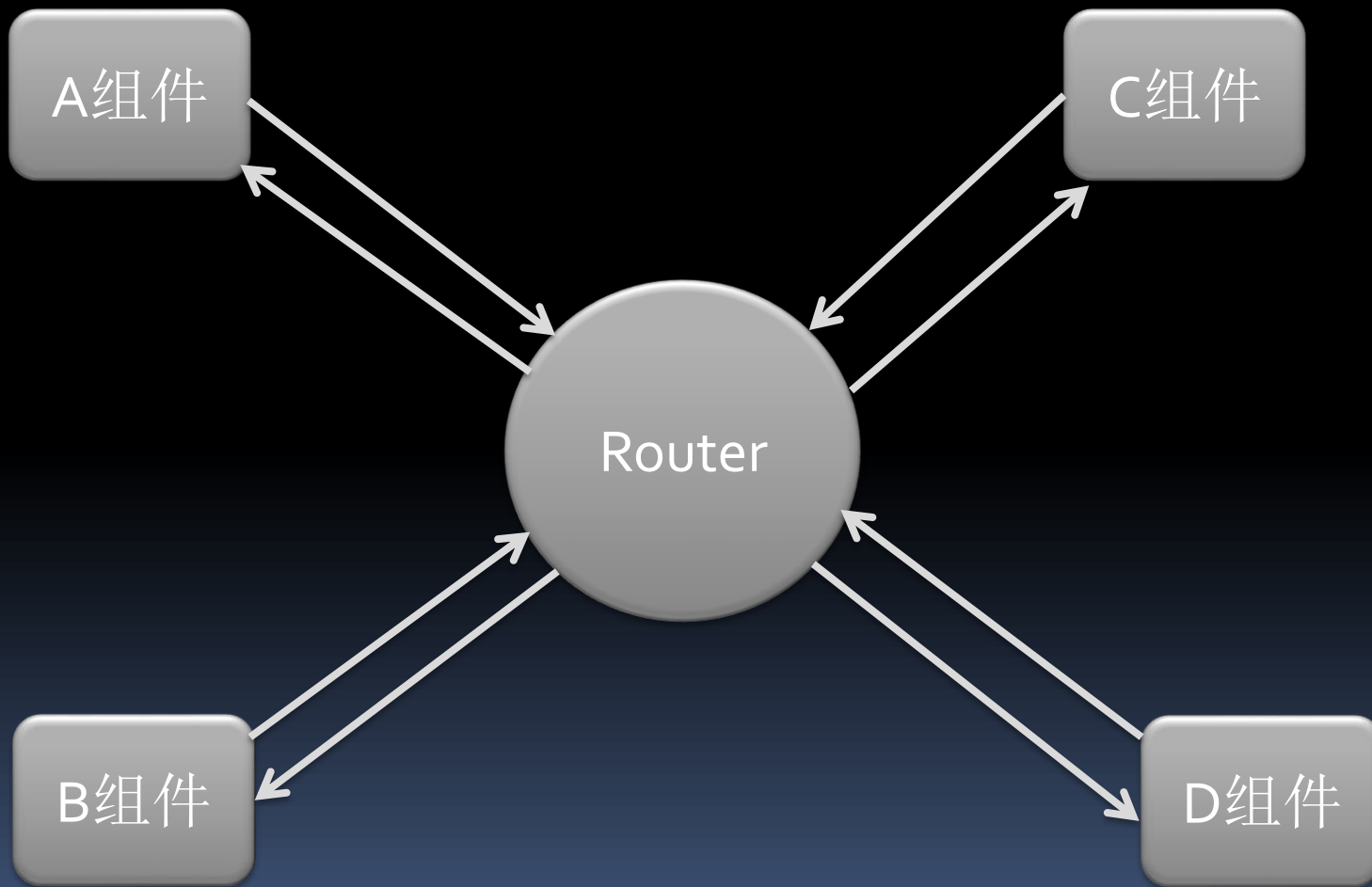
## 车金家项目结构图



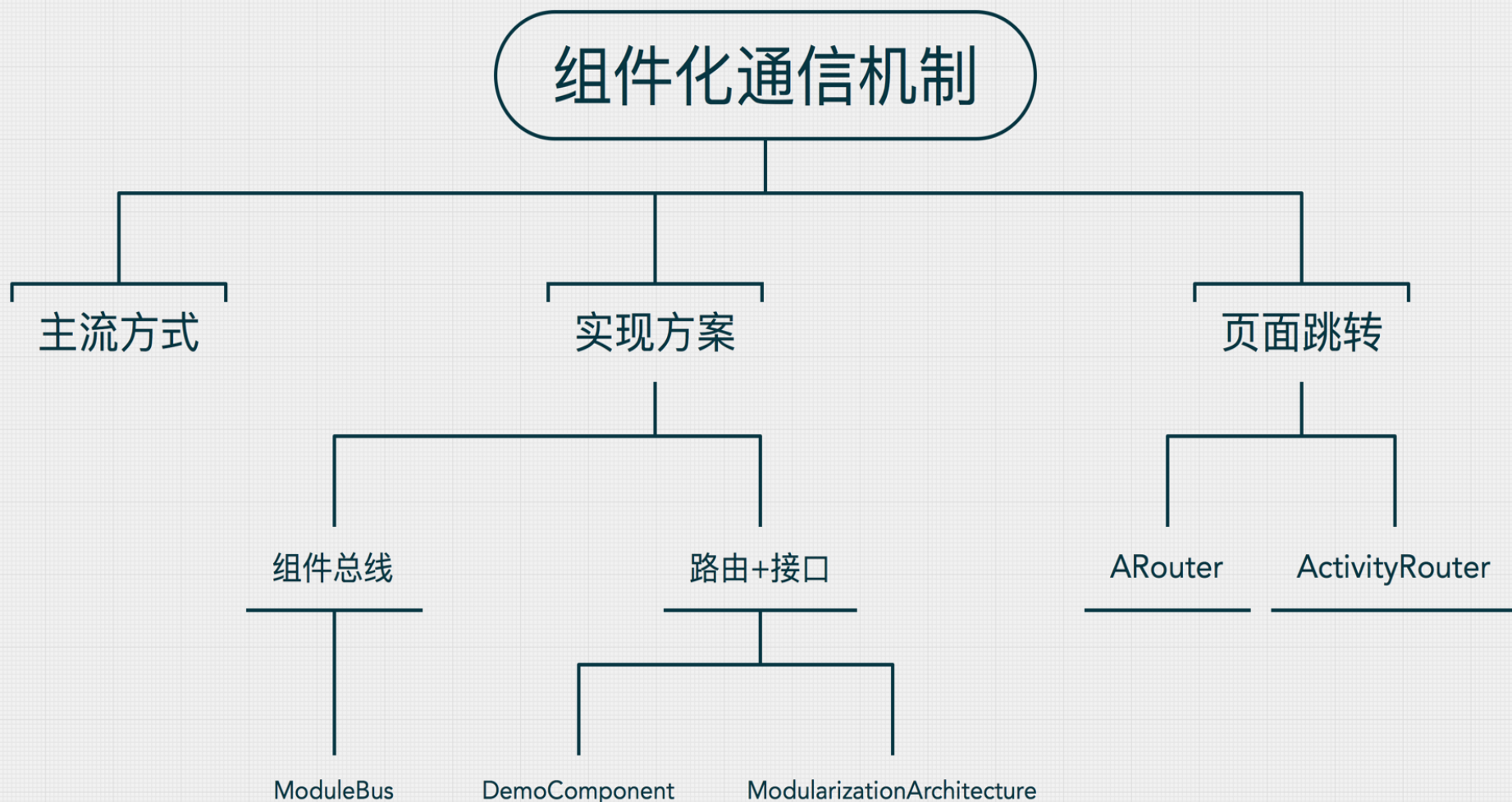
# Android客户端组件化拆分



# 组件与路由的关系

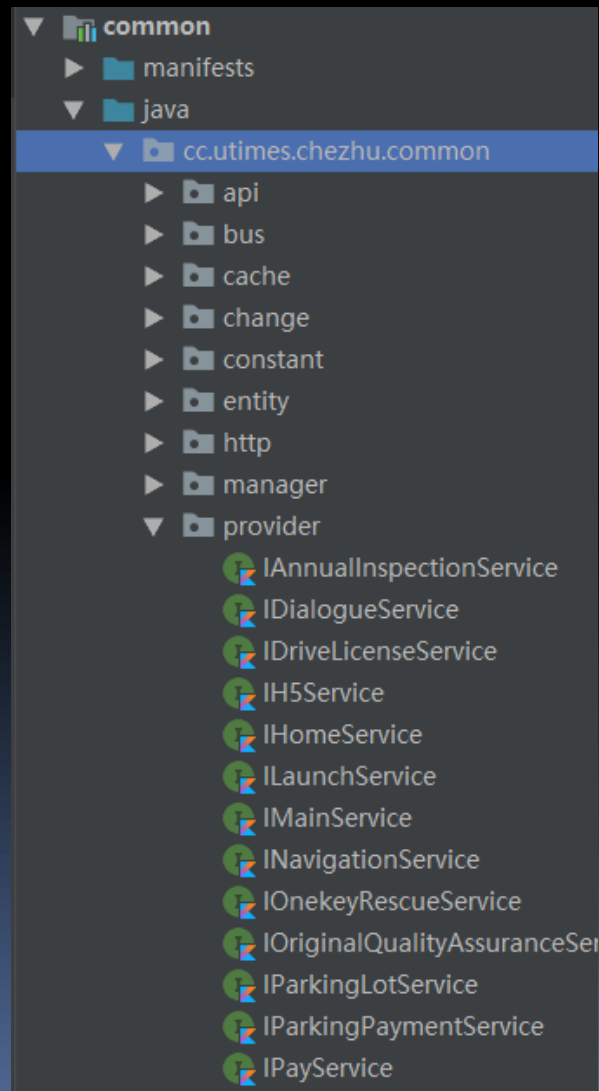


# 如何实现组件化之间的通信



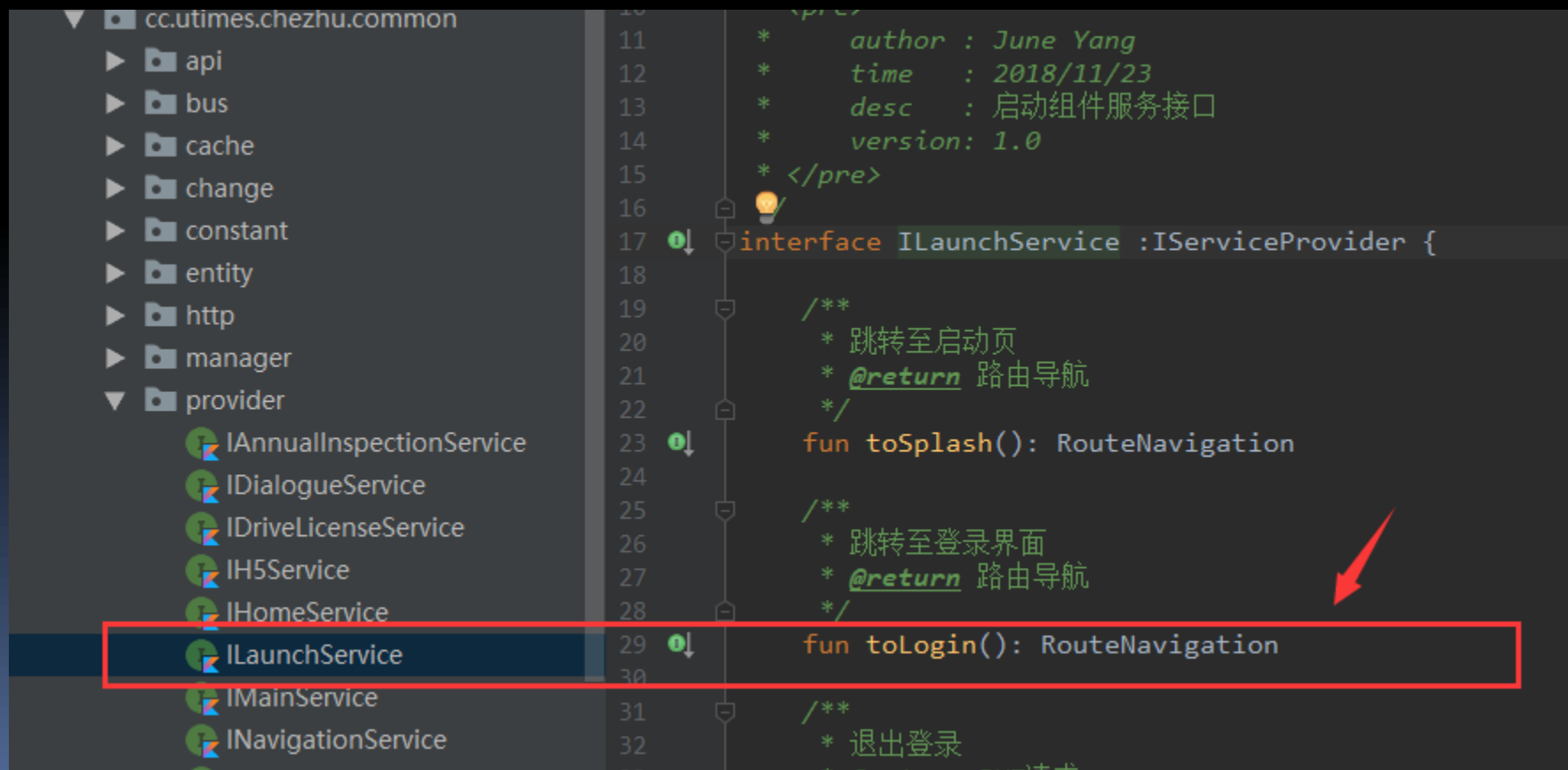
# ARouter路由服务实现

1.定义公共组件路由API和入口  
通过路由服务组件查找



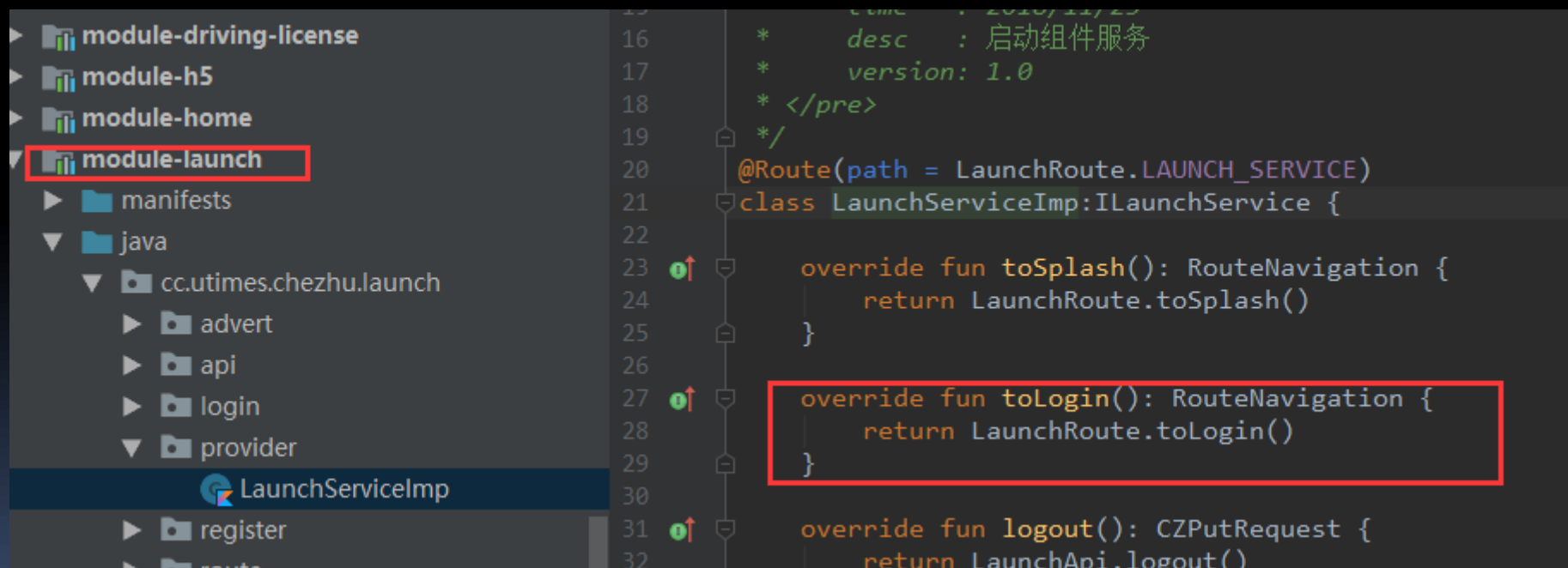
# ARouter路由服务实现

## 2.组件路由实现



# ARouter路由服务实现

## 3.具体实现



The screenshot displays an IDE interface with a project structure on the left and a code editor on the right. The project structure shows a hierarchy of modules: `module-driving-license`, `module-h5`, `module-home`, and `module-launch` (highlighted with a red box). Under `module-launch`, there are sub-directories `manifests` and `java`. The `java` directory contains a package `cc.utimes.chezhu.launch` with sub-packages `advert`, `api`, `login`, and `provider`. The `LaunchServiceImp` class is selected in the project structure.

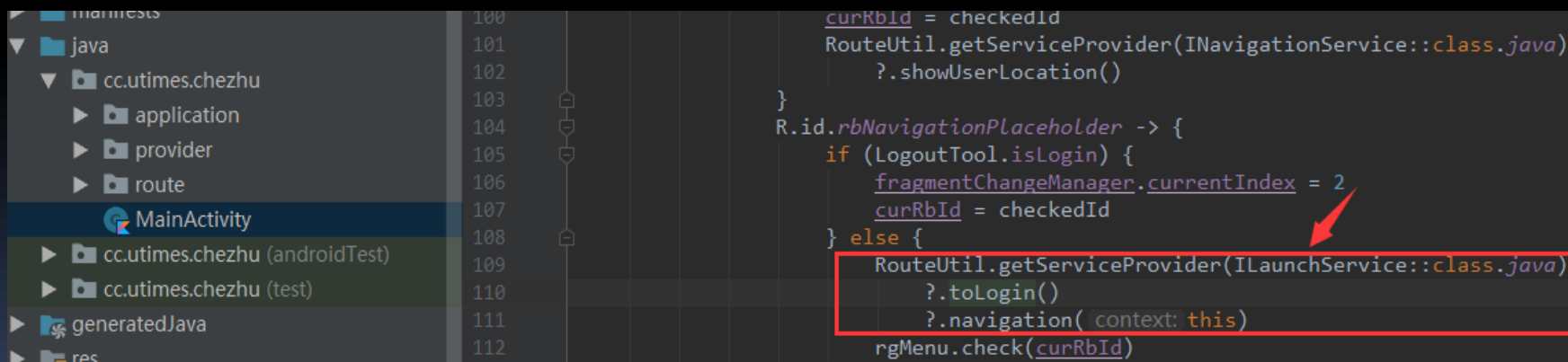
The code editor shows the implementation of the `LaunchServiceImp` class, which implements the `ILaunchService` interface. The code is as follows:

```
16  *      desc    : 启动组件服务
17  *      version: 1.0
18  *  </pre>
19  */
20  @Route(path = LaunchRoute.LAUNCH_SERVICE)
21  class LaunchServiceImp:ILaunchService {
22
23  override fun toSplash(): RouteNavigation {
24      return LaunchRoute.toSplash()
25  }
26
27  override fun toLogin(): RouteNavigation {
28      return LaunchRoute.toLogin()
29  }
30
31  override fun logout(): CZPutRequest {
32      return LaunchApi.logout()
33  }
```

The `toLogin()` method implementation is highlighted with a red box.

# 如何调用？

路由使用：比如我们启动App程序后跳转到登陆页面，使用Login 接口里的方法，使用如下：





下拉刷新

快速开发框架

组件化

Material Design

模块化

Volley、OKHttp

热修复

插件化

我们还能做什么

RN、WEEX、Flutter

MVP、MVVM

Hybird

# 提升研发效率

- 迭代效率
- 发布效率
- 研发成本

# 进一步组件化提升迭代效率

- Library aar → application apk
- Android App Bundle (gradle/dynamic-feature)
- .....

# Dynamic Feature Module



Legacy APK



Dynamic Delivery

# 通过技术升级降低研发成本

- Java语言 → Kotlin语言
- Android、iOS → RN、Weex、Flutter
- .....

Q & A

# Thanks