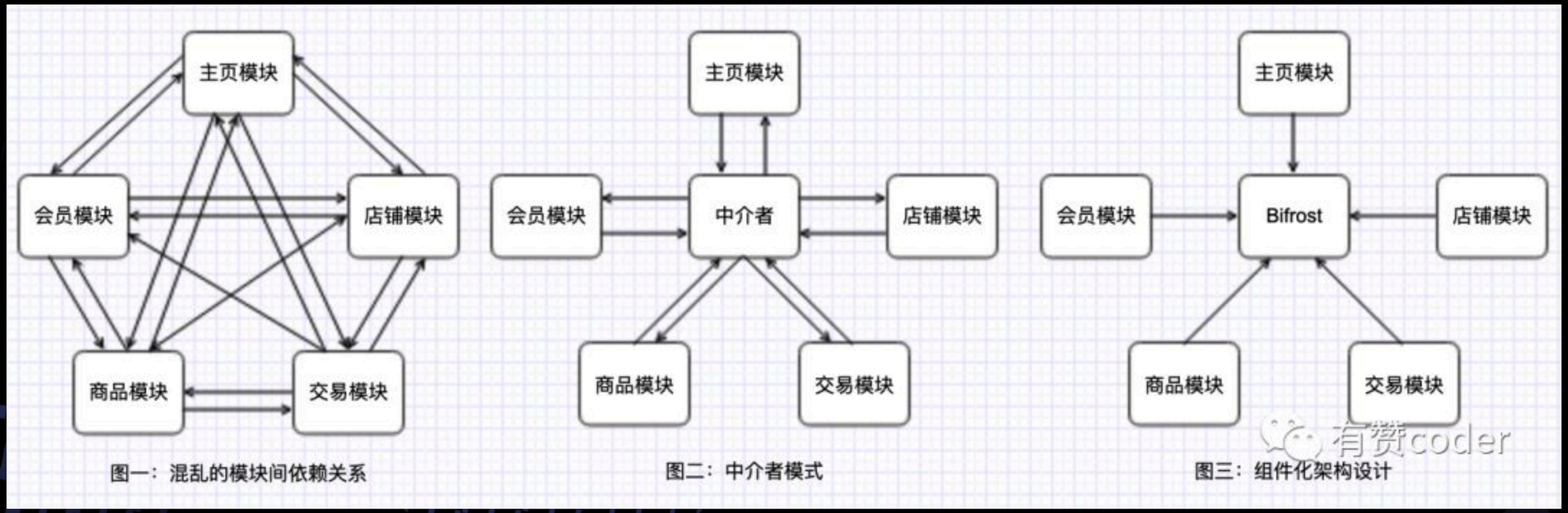
Flutter混编工程的模块化架构设计实践

——Jackie

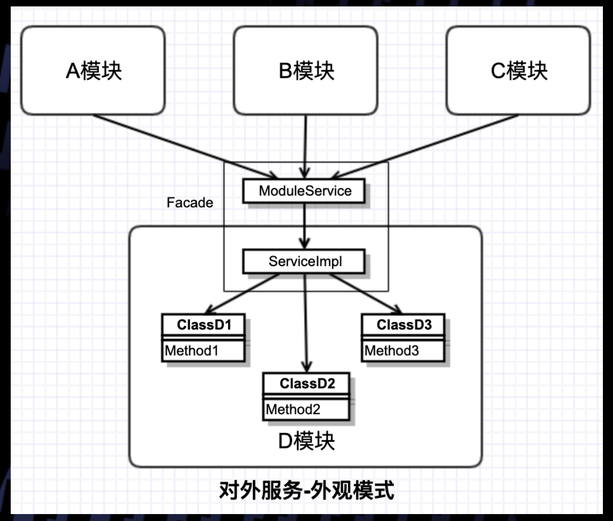
老话题，业务模块化

- 术语调频：业务模块 vs 功能组件
- 业务模块化：

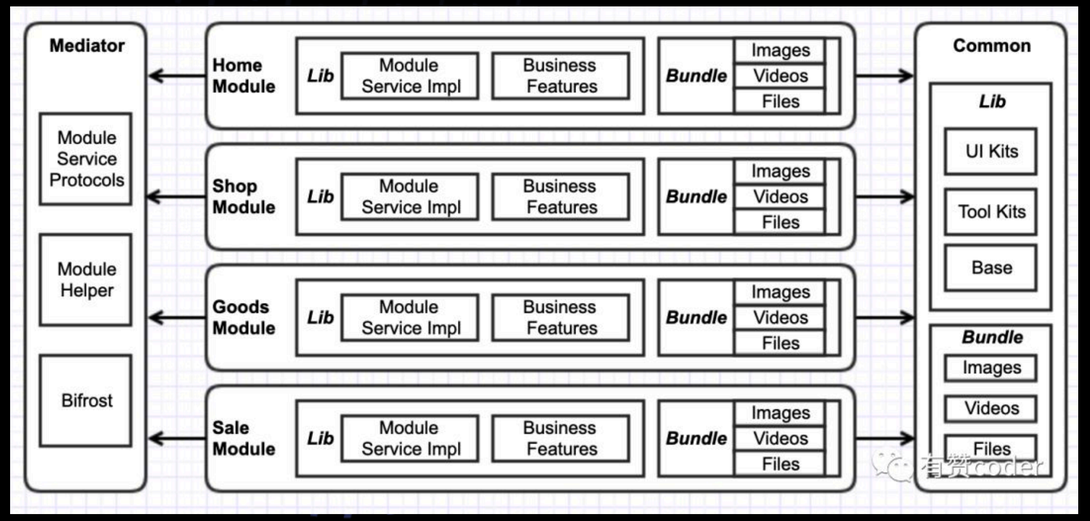


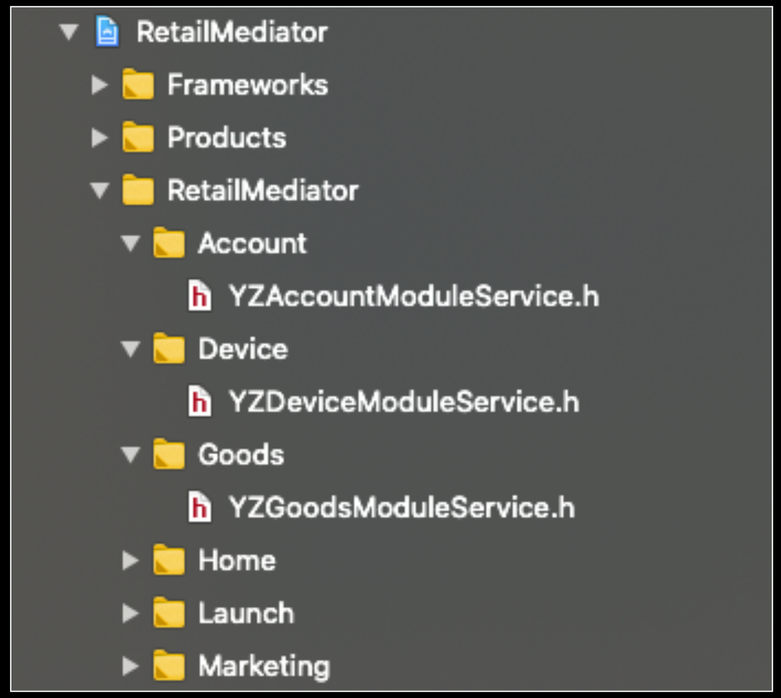
老话题，业务模块化

- 业界常见方案： 蘑菇街, CTMediator, BeeHive, ……
- 有赞的方案 - Bifrost（雷神里的彩虹桥）
  - 以外观模式设计模块架构，拆分ModuleService和ServiceImpl
  - 对外服务：服务接口，模型接口，路由声明，消息通知，……



对外服务-外观模式

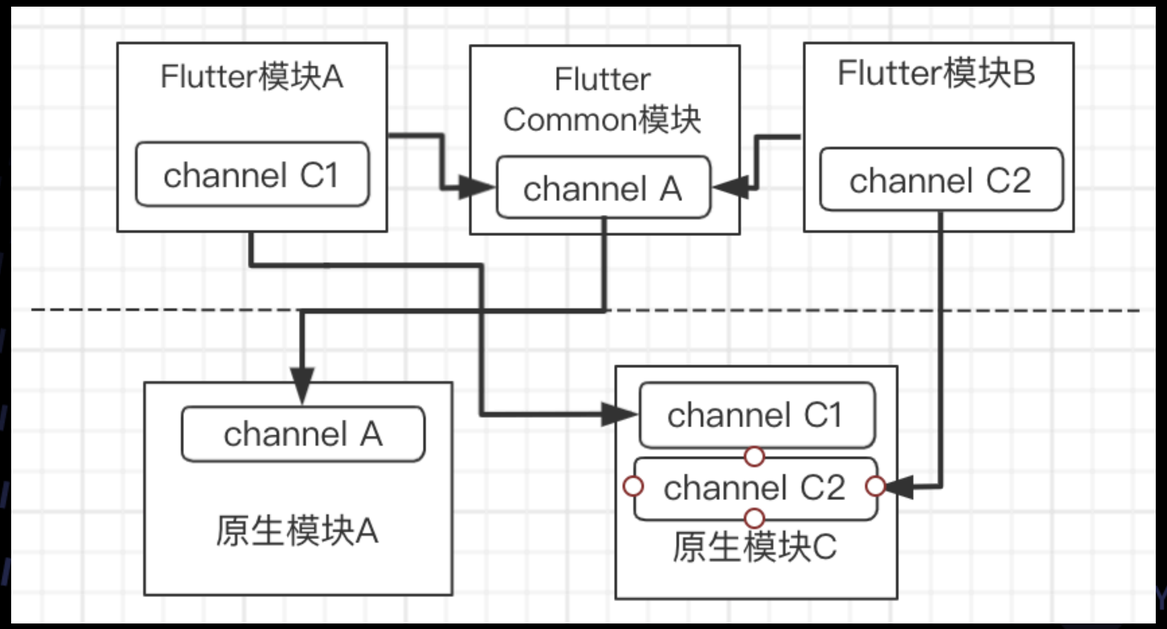





```
#define BFModule(service_protocol)
    ((id<service_protocol>)[Bifrost
    moduleByService:@protocol(service_protocol)])
...
YZRetailShopSystemType systemType =
    BFModule(YZShopModuleService).shopModel.shopType;
...
```

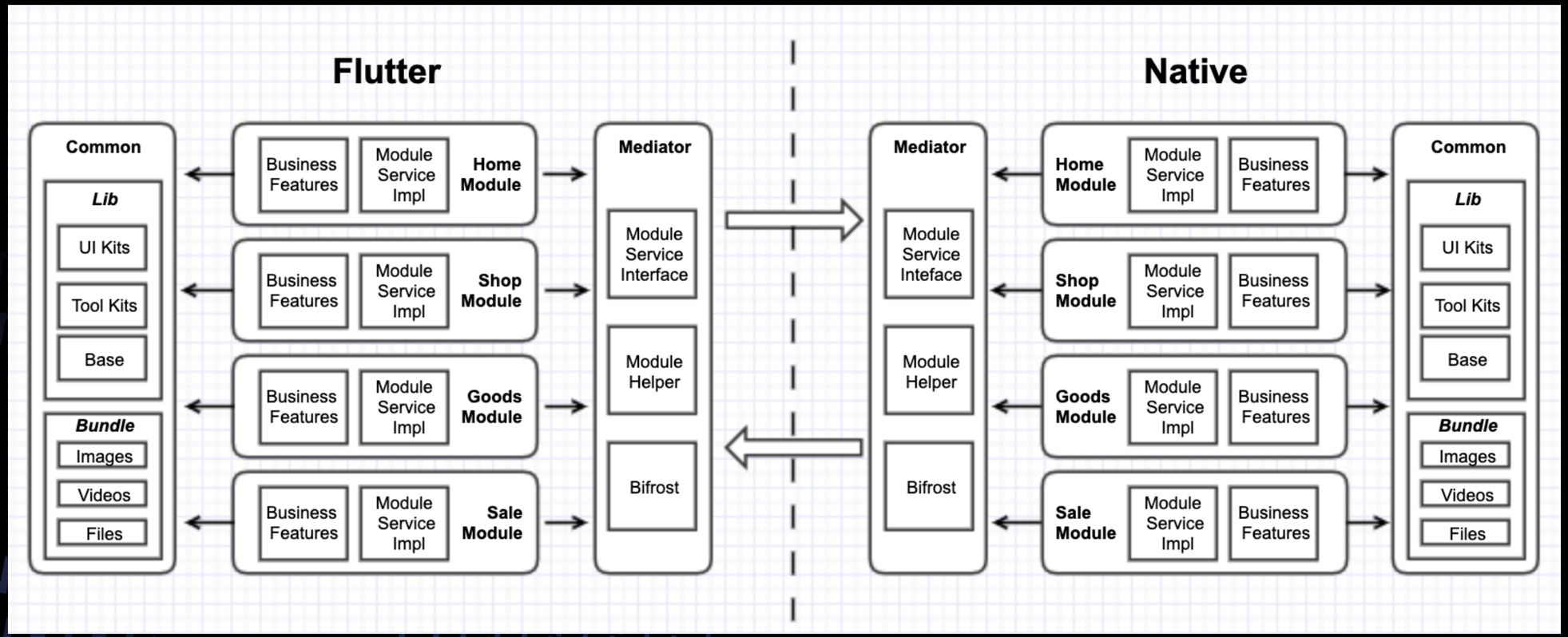
Flutter => 新问题

- 设计上的退步：Flutter侧未遵循模块化设计
  - Flutter模块之间的通讯，把代码下沉到common
  - 路由注册等业务代码直接写到App壳工程main.dart文件中
  - ……
- Flutter调用原生模块的服务
  - 随意写channel，散乱，重复



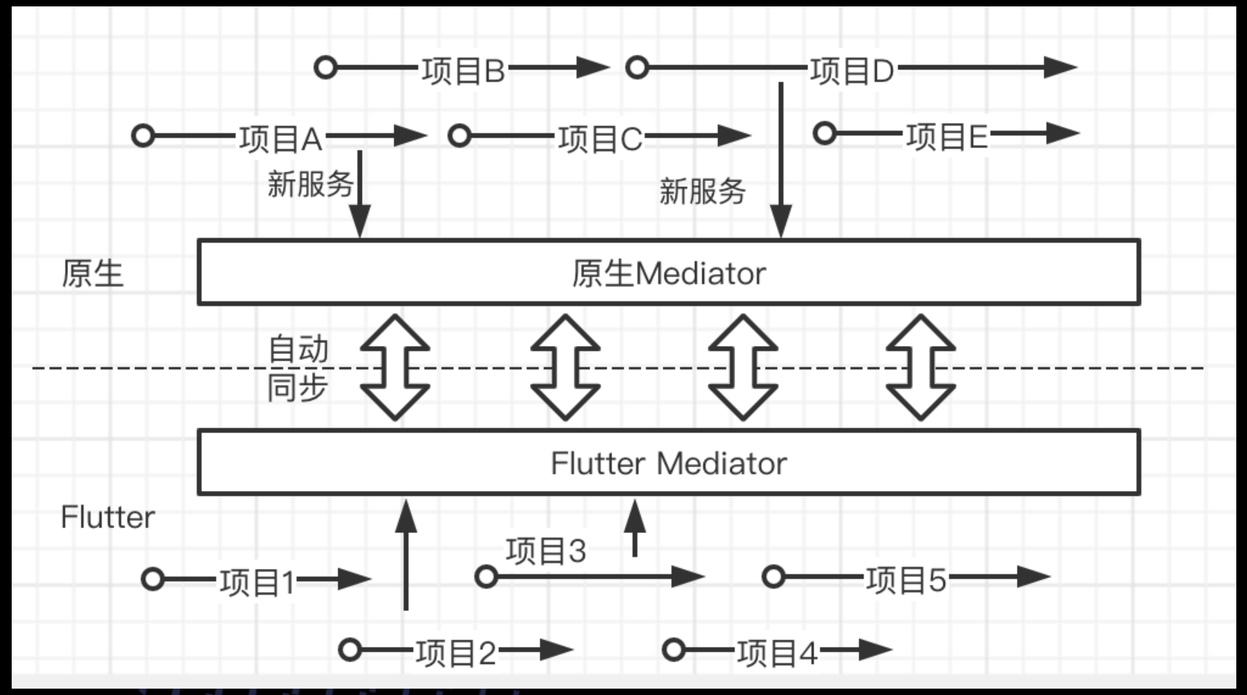
跨栈场景下的服务暴露与调用

- 怎么优化？很简单，镜像一下：



跨栈场景下的服务暴露与调用

- 理想很丰满，现实很骨感
- 关键挑战：用于跨端通讯的channel代码 **手写 vs 自动化**



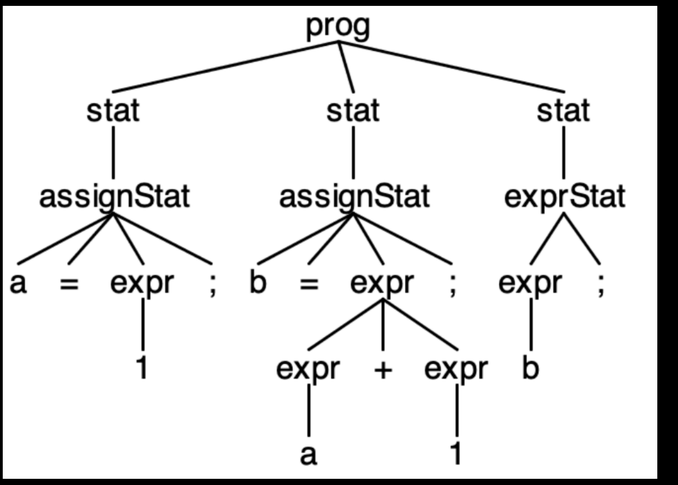
跨栈场景下的服务暴露与调用

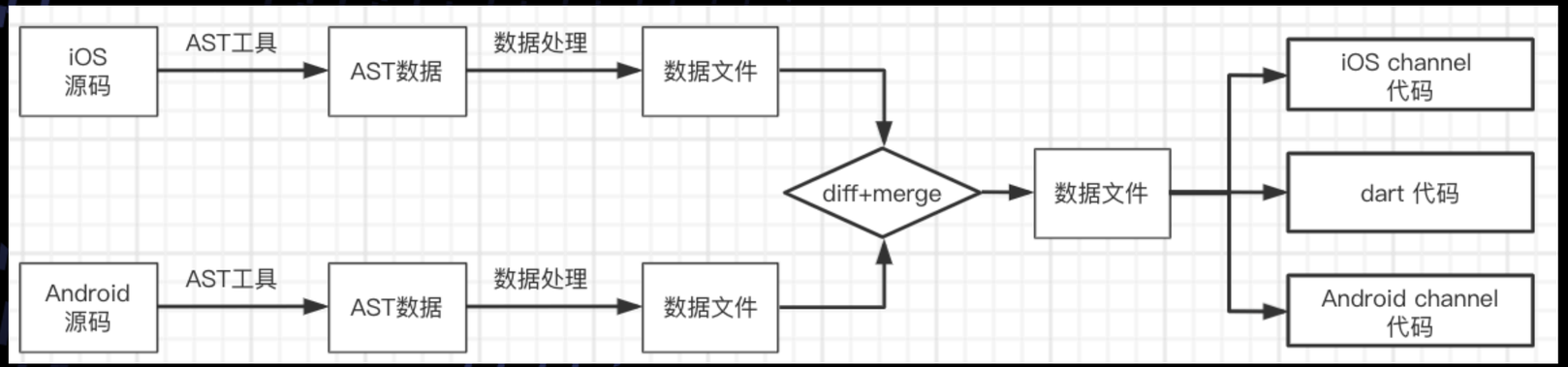
- 自动化的价值
  - **聚焦**：原生项目的同学聚焦原生部分，不需要分散精力额外去实现Flutter侧服务。
  - **质量**：避免某一侧有变动，另一侧没有及时同步引发线上问题
  - **效率**：跑在业务前面，需要用时有现成的
- 自动化的问题：实现复杂
  - **源码解析**：语言多，语法特性多
  - **多端差异**：iOS、Android、Dart有较多语法差异和设计差异
  - **规范落地**：新的设计规范的落地问题

服务自动同步-源码解析

- AST（抽象语法树，Abstract Syntax Code）：
  - 源代码的抽象语法结构的树状表示
  - 很方便做生成物(AST)的各种处理
- 基于AST的源码解析流程

```
a = 1;
b = a + 1;
b;
```





服务自动同步-源码解析

- AST常用工具：ClangAST，Lex+Yacc，Antlr，……
- Antlr

  - 全语法支持，也方便扩展自定义语法描述文件
  - 文档齐全，例子众多

```
topLevelDeclaration
    : importDeclaration
    | functionDeclaration
    | declaration
    | classInterface
    | classImplementation
    | categoryInterface
    | categoryImplementation
    | protocolDeclaration
    | protocolDeclarationList
    | classDeclarationList
    | functionDefinition
    ;

importDeclaration
    : '@import' identifier ';'
    ;

classInterface
    : IB_DESIGNABLE?
      '@interface'
       className=genericTypeSpecifier (':' supe
      '@end'
    ;

categoryInterface
    : '@interface'
       categoryName=genericTypeSpecifier LP cla
      '@end'
    ;

classImplementation
    : '@implementation'
```

服务自动同步-多端差异

- 基础类型

The following table shows how Dart values are received on the platform side and vice versa:

Dart	Java	Kotlin	Obj-C	Swift
null	null	null	nil (NSNull when nested)	nil
bool	java.lang.Boolean	Boolean	NSNumber numberWithBool:	NSNumber(value: Bool)
int	java.lang.Integer	Int	NSNumber numberWithInt:	NSNumber(value: Int32)
int, if 32 bits not enough	java.lang.Long	Long	NSNumber numberWithLong:	NSNumber(value: Int)
double	java.lang.Double	Double	NSNumber numberWithDouble:	NSNumber(value: Double)
String	java.lang.String	String	NSString	String
Uint8List	byte[]	ByteArray	FlutterStandardTypedData typedDataWithBytes:	FlutterStandardTypedData(bytes: Data)
Int32List	int[]	IntArray	FlutterStandardTypedData typedDataWithInt32:	FlutterStandardTypedData(int32: Data)
Int64List	long[]	LongArray	FlutterStandardTypedData typedDataWithInt64:	FlutterStandardTypedData(int64: Data)
Float64List	double[]	DoubleArray	FlutterStandardTypedData typedDataWithFloat64:	FlutterStandardTypedData(float64: Data)
List	java.util.ArrayList	List	NSArray	Array
Map	java.util.HashMap	HashMap	NSDictionary	Dictionary

服务自动同步-多端差异

- 复合类型 - 例：模型的传递
  - 问题：
    - 模型的实例是动态变动的，所以无法为模型接口实现channel方法
  - 方案：
    - 原生侧自动做序列化，然后传递数据
    - 在dart侧创建相应模型类，自动反序列化，但只提供get方法

```
//原生服务接口声明
@protocol UserModuleService
...
- (NSObject<UserModelProtocol> *)getUserById:(NSInteger)userId;

- (void)saveUser:(NSObject<UserModelProtocol>*)user;
...
@end
//原生侧模型声明
@protocol UserModelProtocol <NSObject>
@property (nonatomic, assign) NSInteger userId;
@property (nonatomic, strong) NSString *name;
@property (nonatomic, strong) NSString *telephone;
- (NSInteger)getCurrentPoints;
@end

//dart侧上层业务内调用
...
user = getUserById(userId);
print(user.name);
print(user.telephone);
...
```

```
//iOS侧channel代码实现
- (void)registerWithEngine:(FlutterEngine *)engine {
    ... //创建channel对象
    [channel setMethodCallHandler:^(FlutterMethodCall *call,
                                    FlutterResult result) {
        NSString *methodName = call.method; //方法名
        NSArray *arguments = call.arguments; //调用参数
        if ([methodName isEqualToString:@"name"]) {
            NSString *name = [instance name];
            result(name);
        } else if (...) {...}
        ...
    }];
}
```

服务自动同步-多端差异

- 特殊类型 - 例：Block对应Dart的什么？

```
例：原生侧login服务
//iOS侧接口声明
typedef void (^Handler)(ResultModel *result, NSError *error);
- (void)login:(NSString*)phone
     password:(NSString*)password
      handler:(Handler)handler;
```

```
//Android侧声明
fun login(phone: String, password: String): Observable<ResultModel>
```

```
//Dart侧使用
//1.Future方式
//方法声明
Future<ResultModel> login(String phone, String password) async{}
//业务方使用
...
ResultModel result = await login(phone, password);
...
```

```
//iOS侧channel代码实现
- (void)registerWithEngine:(FlutterEngine *)engine {
    ...
    [channel setMethodCallHandler:^(FlutterMethodCall *call,
                                    FlutterResult result) {
        NSString *methodName = call.method;
        NSArray *arguments = call.arguments;
        if ([methodName isEqualToString:@"login_name_password"]) {
            [BFModule(YZAccountModuleService) login:arguments[0]
                                           password:arguments[1]
                                            handler:^(ResultModel *model,
                                                      NSError *error) {
                if (error) {
                    return result([[FlutterError alloc] initWithError:error]);
                } else {
                    return result([model yy_modelToJSONObject]);
                }
            }];
        } else if ([methodName isEqualToString:@"xxxxx"]) {
            ...
        }
        ...
    }];
    ...
}
```

- block包含多个参数怎么办？
- 一个方法包含多个block怎么办？
- block没有被执行，await不就卡死了吗？
- ……

Youzan.com

服务自动同步-多端差异

- 特殊类型 - 例：Block对应Dart的什么？**还是得用Function！**

```
例：原生侧login服务
//iOS侧接口声明
typedef void (^Handler)(ResultModel *result, NSError *error);
- (void)login:(NSString*)phone
     password:(NSString*)password
      handler:(Handler)handler;
```

```
//Android侧声明
fun login(phone: String, password: String): Observable<ResultModel>
```

```
//Dart侧使用
//2.function方式
//channel方法
void login(String phone, String password,
        void Function(ResultModel result, FlutterError error) handler) {
    _channel.invokeMethod('login_name_password',
                          [phone, password]).then((value) {
      handler(ResultModel.fromJson(value), null);
    }, onError: (error) {
      handler(null, error);
    });
}
//业务方使用
...
login(phone, password, (ResultModel result, FlutterError error) {
   ...
});
...
```

```
//iOS侧channel代码实现
- (void)registerWithEngine:(FlutterEngine *)engine {
    ...
    [channel setMethodCallHandler:^(FlutterMethodCall *call,
                                    FlutterResult result) {
        NSString *methodName = call.method;
        NSArray *arguments = call.arguments;
        if ([methodName isEqualToString:@"login_name_password"]) {
            [BFModule(YZAccountModuleService) login:arguments[0]
                                           password:arguments[1]
                                            handler:^(ResultModel *model,
                                                      NSError *error) {
                if (error) {
                    return result([[FlutterError alloc] initWithError:error]);
                } else {
                    return result([model yy_modelToJSONObject]);
                }
            }];
        } else if ([methodName isEqualToString:@"xxxxx"]) {
            ...
        }
        ...
    }];
    ...
}
```

服务自动同步-多端差异

- 其它问题
  - 接口注释信息，AST拿不到
    - 通过方法关键字去源码文件中反查
  - 特殊对象，如UIView， UIButton等
    - 暂不支持。特别需求case by case处理
  - 技术栈独特接口或类型（就是只给自己用的，如Android的Context）
    - 自动同步工具忽略

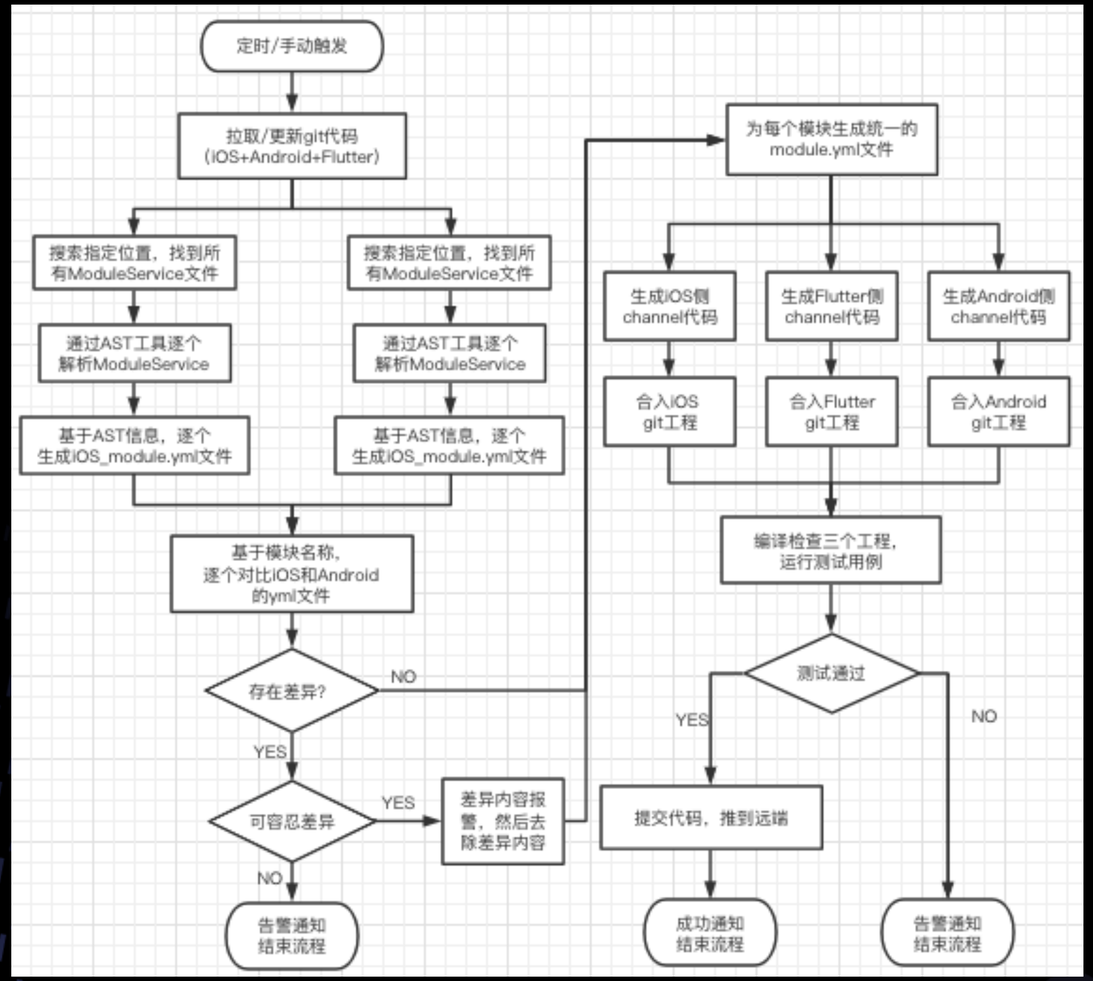
```
/**
 获取是否是类目白名单
 */
- (BOOL)[redacted];

/// 查询品牌列表
/// @param pageNo 页码
/// @param pageSize size
/// @param brandName 品牌名（模糊查询）
/// @param completion 查询结果
- (void)[redacted]:(NSInteger)pageNo
                 pageSize:(NSInteger)pageSize
                brandName:(NSString * __nullable)brandName
               completion:(void(^)(NSArray<NSObject<YZBrandModel
                  __nullable brandList, NSError * __nullable er
```

服务(接口)同步问题 - 规范落地

- 老代码怎么办?
  - 方法名问题：提供方法名注解，可以指定flutter侧方法名，原生侧方法名不需调整
  - 设计实现问题：提供Ignore注解，暂不自动同步，允许业务方按节奏推进
- 新代码怎么办?
  - 方法设计方式上是不是受到很大制约?
    - 不断迭代优化 + case by case看
  - 怎么确保iOS和Android在方法设计上的一致?
    - 参考路由和后端接口等跨栈信息的规范，技术方案时就约定好。

自动化同步任务

- 支持多种触发方式
  - 暂时没有加入CI环节
- 异常处理
  - 差异处理
  - 自动化测试
  - 告警通知



模块间通讯之外

- 明确Flutter侧模块设计规范
  - 自己的业务代码写自己模块，禁止写在main.dart/app.dart中
    - 模块注册：基于App的pubspec.yaml中的依赖解析，生成Module.dart文件
    - 路由注册： 通过注解进行注册，通过mediator对外提供。
  - Bifrost也会提供App关键生命周期信息的监听方法

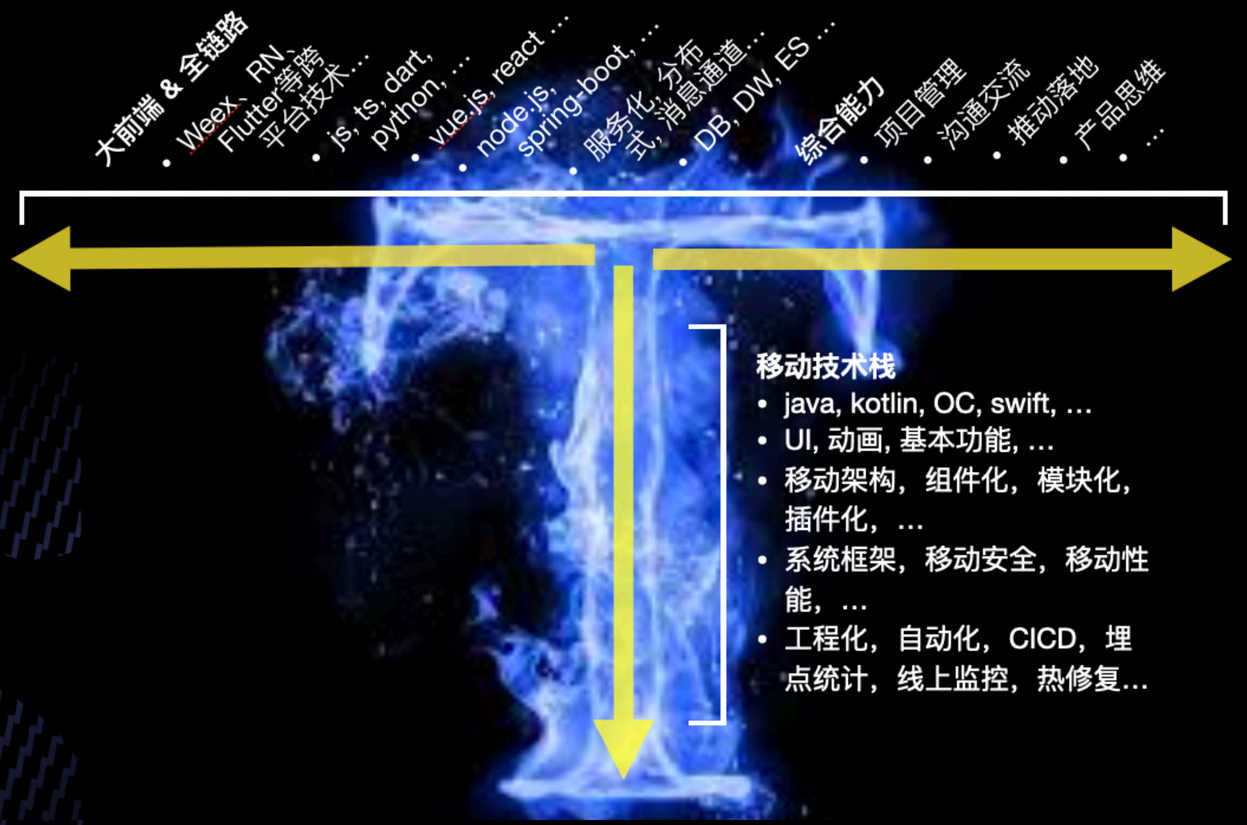
```
// 写在 mediator/account.dart 路由地址
static const String VerifyScaner = "//sample/verify_scaner";

//account模块内业务代码，路由和widget页面绑定
@BifrostRouter(VerifyScaner)
class VerifyScanPage extends StatefulWidget {
}

//基于App壳工程的pubspec.yaml中的module依赖解析， 自动生成Module.dart
class Module {
    static void load() {
        List<BifrostBaseModuleInterface> modules = [
          AccountModule(),
            ...
        ];
        // 注册模块
        Bifrost.register(modules);
    }
}

//App壳工程的app.dart中，仅有通用代码
class _AppState extends State<MyApp> {
  @override
  void initState() {
    super.initState();
    // 触发模块注册
    Module.load();
  }
}
```

总结与展望

- 更多优化:
  - 单向同步 -> 双相同步
  - 更多语法和类型的支持
  - 调用性能优化
  - ...
- Flutter提效: 开发 + 测试 双提效

有赞移动技术团队 & QA

- 文化：追求卓越，赋能业务
- 成长理念：T型人才
- 还有众多移动HC, 欢迎加入，一起玩转大前端技术~



Questions

Youzan.com