学学课堂 致力于培养卓越的程序员

Retrofit源码解析

Alvin老师

前三星/小米研发经理

12月27日 20:00

- Retrofit 创建和配置过程
- Retrofit Adapter适配方案
- Retrofit converter转化原理
- Retrofit 设计模式简介与分析

VIP课程

我是谁?

Alvin

华南理工大学 软件工程 工程硕士

三星中国研究院 5 years
项目经理

小米科技 2 years
技术总监

- 曾就业于三星中国研究院及小米旗下互联网公司担任android任软件工程师及项目经理
- 拥有扎实的C/Java 基础，深入研究android系统多年。
- 讲课形象生动，热情洋溢

目 录

CONTENTS

retrofit 创建过程

retrofit 配置

ServiceMethod 构建

Adpater & converter 添加

Retrofit Adapter

Call<T> 函数适配过程

retrofit converter

ResponseBody 转化为 Bean的过程

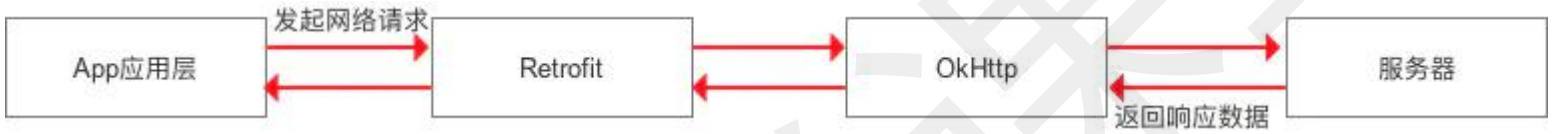
其他数据转变为 RequestBody过程

Retrofit 设计模式

9大设计模式分析

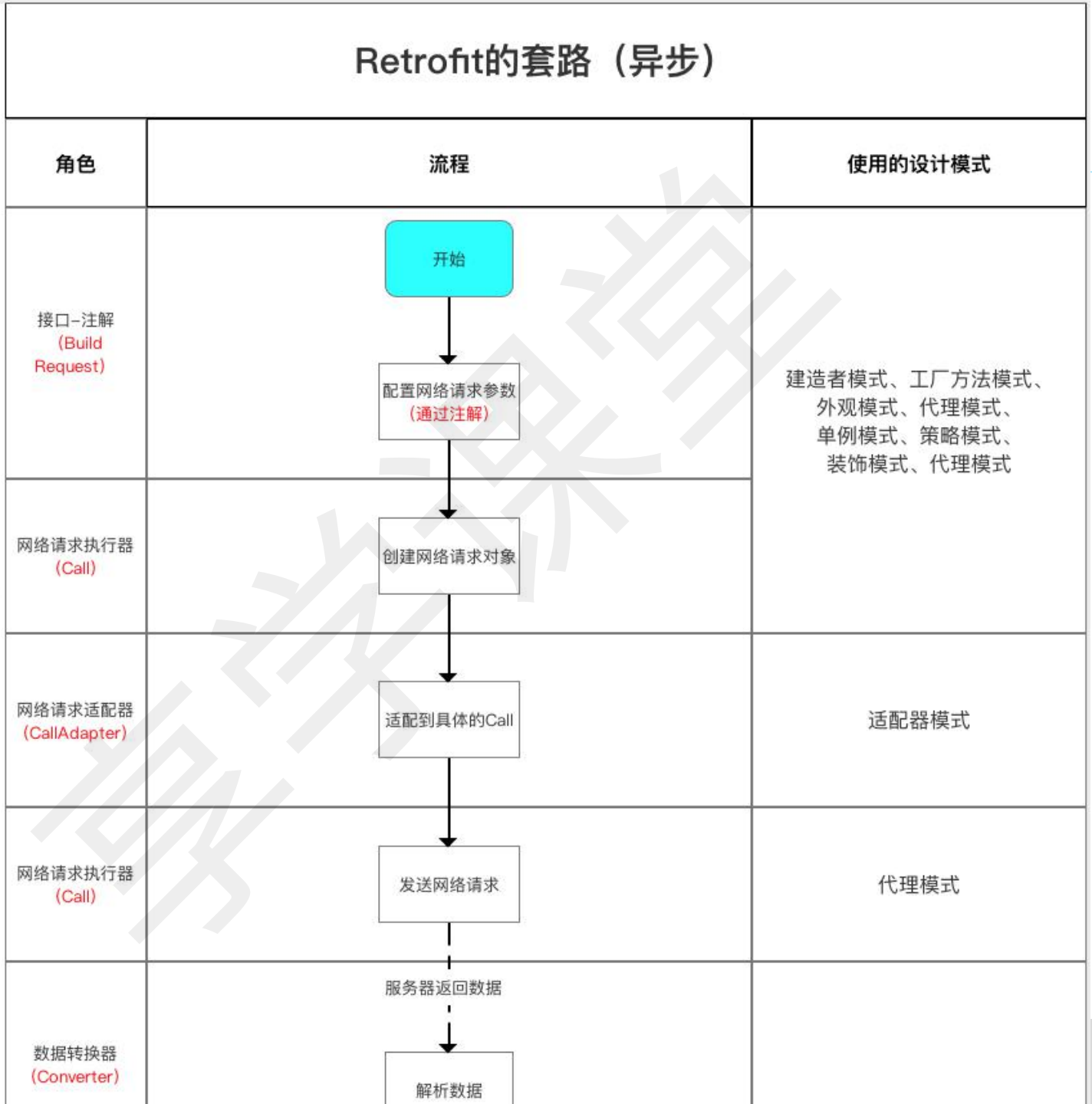
retrofit 是什么

准确来说，Retrofit 是一个 RESTful 的 HTTP 网络请求框架的封装。
原因：网络请求的工作本质上是 OkHttp 完成，而 Retrofit 仅负责 网络请求接口的封装



- App应用程序通过 Retrofit 请求网络，实际上是使用 Retrofit 接口层封装请求参数、Header、Url 等信息，之后由 OkHttp 完成后续的请求操作。
- 在服务端返回数据之后，OkHttp 将原始的结果交给 Retrofit，Retrofit根据用户的需求对结果进行解析。

retrofit 总流程



代理实例创建过程

```
retrofit = new Retrofit.Builder()
        .baseUrl("https://www.wanandroid.com/")
        .addConverterFactory(GsonConverterFactory.create(new Gson()))
        .build();
```

1

```
ISharedListService sharedListService =  retrofit.create(ISharedListService.class);
```

2

1: 成功建立一个Retrofit对象的标准：配置好Retrofit类里的成员变量

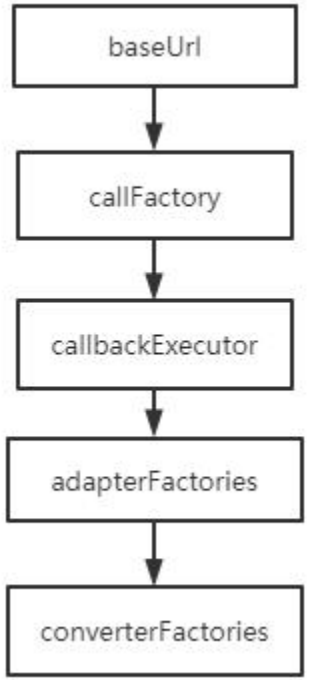
baseUrl：网络请求的url地址

callFactory：网络请求工厂

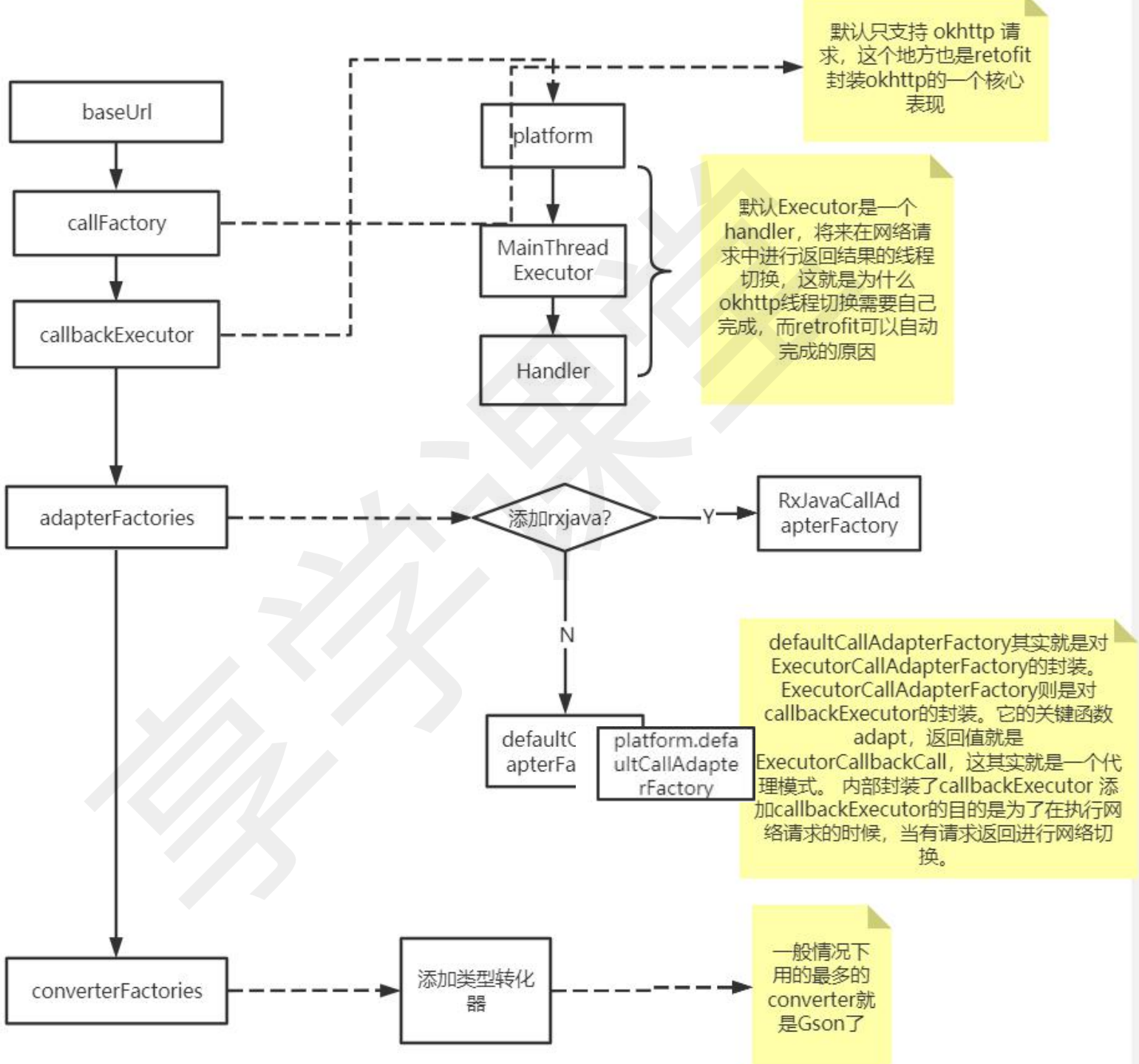
callbackExecutor：回调方法执行器

adapterFactories：网络请求适配器工厂的集合

converterFactories：数据转换器工厂的集合



代理实例创建详情


baseUrl
callFactory
callbackExecutor
adapterFactories
converterFactories
platform
MainThread Executor
Handler
默认只支持 okhttp 请求，这个地方也是retofit封装okhttp的一个核心表现
默认Executor是一个handler，将来在网络请求中进行返回结果的线程切换，这就是为什么okhttp线程切换需要自己完成，而retrofit可以自动完成的原因
添加rxjava?
Y
RxJavaCallAdapterFactory
N
defaultCapterFa
platform.defaultCallAdapterFactory
defaultCallAdapterFactory其实就是对ExecutorCallAdapterFactory的封装。ExecutorCallAdapterFactory则是对callbackExecutor的封装。它的关键函数adapt，返回值就是ExecutorCallbackCall，这其实就是一个代理模式。 内部封装了callbackExecutor 添加callbackExecutor的目的是为了在执行网络请求的时候，当有请求返回进行网络切换。
添加类型转化器
一般情况下用的最多的converter就是Gson了

代理实例创建过程

```
retrofit = new Retrofit.Builder()                                    1
        .baseUrl("https://www.wanandroid.com/")
        .addConverterFactory(GsonConverterFactory.create(new Gson()))
        .build();
ISharedListService sharedListService =  retrofit.create(ISharedListService.class);  2
```

2: 创建了一个ISharedListService 接口类的对象，create函数内部使用了动态代理来创建接口对象，这样的设计可以让所有的访问请求都被代理

Call<SharedListBean> sharedListCall = sharedListService.getSharedList(2,1);

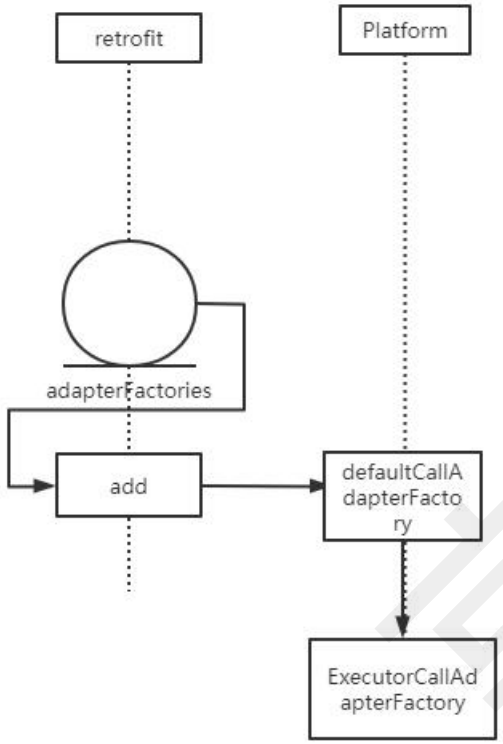
调用getSharedList的时候，在动态代理里面，会存在一个函数 getSharedList，这个函数里面会调用 invoke，这个invoke函数也就是retrofit里面 invoke函数。
所以，动态代理可以代理所有的接口，**让所有的接口都走 invoke函数**，这样就可以**拦截**调用函数的执行，从而**将网络接口的参数配置归一化**。

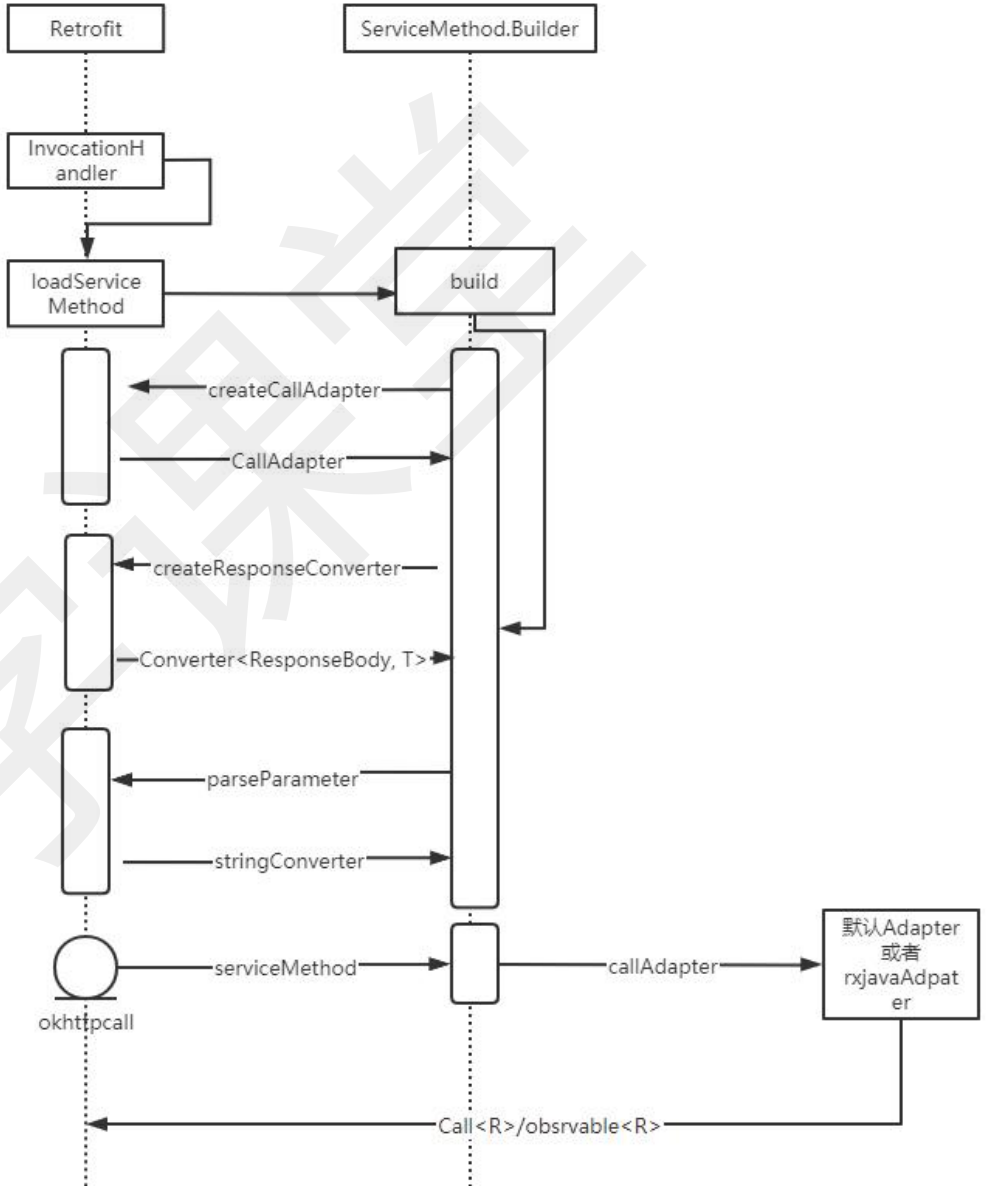
目 录

CONTENTS

访问接口创建过程

```
Call<SharedListBean> sharedListCall
      = sharedListService.getSharedList(2,1);
```



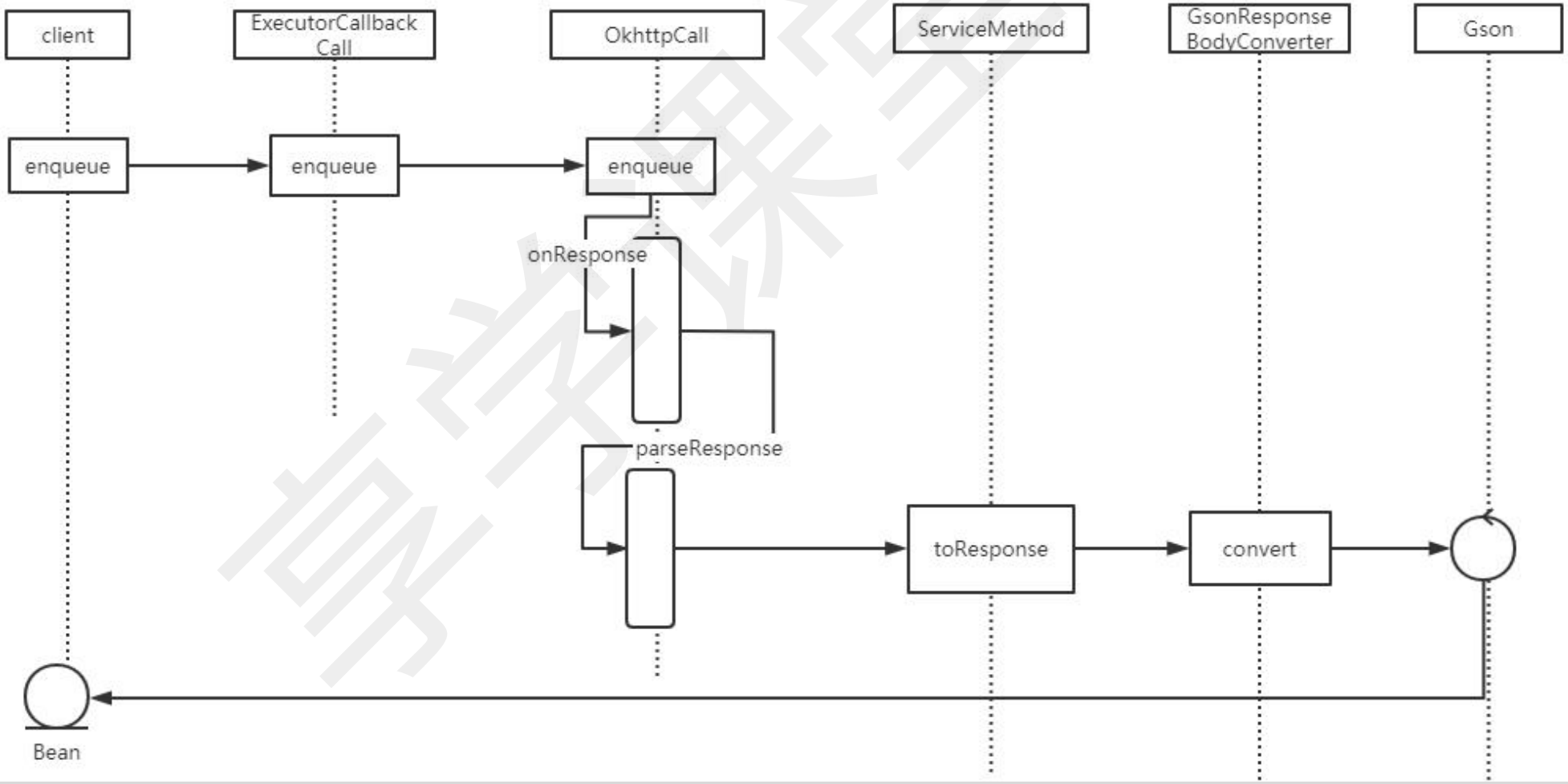


目 录

CONTENTS

网络请求过程 converter response2 javaBean

```
sharedListCall.enqueue(new Callback<SharedListBean>() {...});
```



目 录

CONTENTS

retrofit 创建过程

retrofit 配置

ServiceMethod 构建

Adpater & converter 添加

Retrofit Adapter

Call<T> 函数适配过程

retrofit converter

ResponseBody 转化为 Bean的过程

其他数据转变为 RequestBody过程

Retrofit 设计模式

9大设计模式分析

设计模式

1）Retrofit 实例使用建造者模式通过Builder类构建。

当构造函数的参数大于4个，且存在可选参数的时候既可以使用 建造者设计模式

2）Retrofit 创建时的callFactory，使用工厂方法设计模式，但是似乎并不打算支持其他的工厂。

3）整个retrofit 采用的时外观模式。统一的调用创建网络请求接口实例和网络请求参数配置的方法

4）retrofit里面使用了动态代理来创建网络请求接口实例，这个是retrofit对用户使用来说最大的复用，其它的代码都是为了支撑这个动态代理给用户带来便捷性的

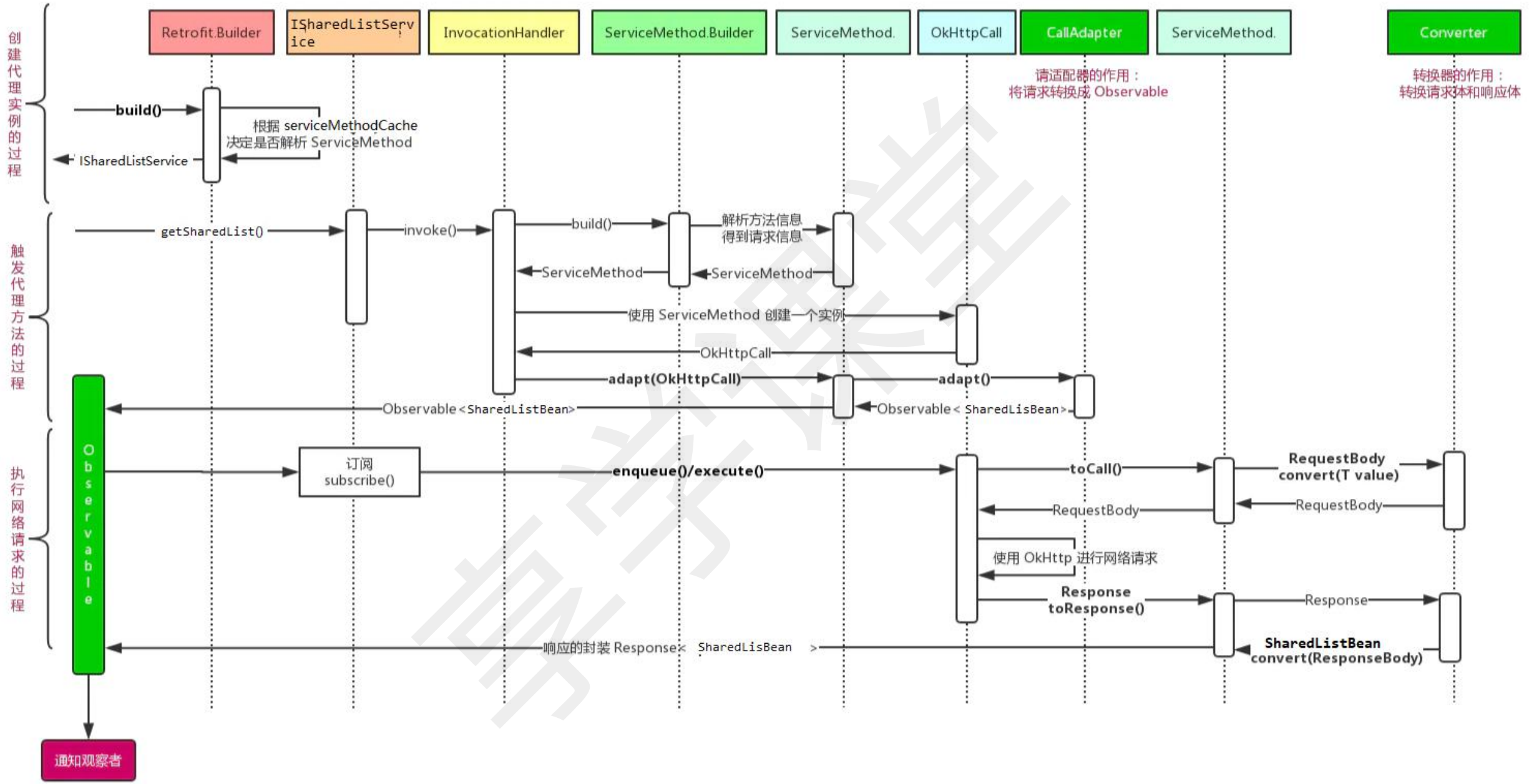
5）使用了策略模式对serviceMethod对象进行网络请求参数配置，即通过解析网络请求接口方法的参数、返回值和注解类型，从Retrofit对象中获取对应的网络的url地址、网络请求执行器、网络请求适配器和数据转换器

设计模式

6）ExecuteCallBack 使用装饰者模式来封装callbackExecutor，用于完成线程的切换

7）ExecutorCallbackCall 使用静态代理(委托) 代理了Call进行网络请求，真正的网络请求由okhttpCall执行，然而okHttpCall不是自己执行，它是okhttp 提供call给 外界（retrofit）使用的唯一门户，其实这个地方就是门面模式

8）ExecutorCallbackCall 的被初始化是在 ExecutorCallAdapterFactory里面通过适配器模式被创建的。CallAdapter采用了适配器模式 为创建访问Call接口提供服务。默认不添加Rxjava则使用默认的ExecutorCallAdapterFactory 将okhttp3.call转变成为 retroift中的call，如果有Rxjava则将okhttp3.call转化为abservable。


创建代理实例的过程
触发代理方法的过程
执行网络请求的过程
Retrofit.Builder
ISharedListService
InvocationHandler
ServiceMethod.Builder
ServiceMethod.
OkHttpCall
CallAdapter
ServiceMethod.
Converter
请适配器的作用：
将请求转换成 Observable
转换器的作用：
转换请求体和响应体
build()
根据 serviceMethodCache
决定是否解析 ServiceMethod
ISharedListService
getSharedList()
invoke()
build()
解析方法信息
得到请求信息
ServiceMethod
ServiceMethod
使用 ServiceMethod 创建一个实例
OkHttpCall
adapt(OkHttpCall)
adapt()
Observable<SharedListBean>
Observable< SharedLisBean>
Observable
订阅
subscribe()
enqueue()/execute()
toCall()
RequestBody
convert(T value)
RequestBody
RequestBody
使用 OkHttp 进行网络请求
Response
toResponse()
Response
SharedListBean
convert(ResponseBody)
响应的封装 Response< SharedLisBean >
通知观察者

retrofit 作业

绘制retrofit进行网络请求的总流程，从enqueue 到 得到 callback响应的总流程