组件化解耦框架集成文档

框架简介：

项目在进行组件化改造中，不可避免的会遇到模块之间的解耦与组件模块之间的服务调用。针对以上问题，本框架提供了一套低侵入、轻便简洁的接入方式。框架通过注解来标识暴露的服务，以确保框架对原代码的低侵入。模块间的服务调用通过调用框架生成的中间类来完成调用，最大程度降低了模块间的耦合。

一、集成步骤:

框架的集成支持gradle自动集成。

1、集成plugin

在project的build.gradle文件中添加依赖和属性配置：

```
buildscript {

    repositories {
        google()
        jcenter()
        //...

    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.1.2'
        classpath 'com.beyondxia.modules:transform-plugin:1.0.4'
        //...

    }
}
```

classpath ‘com.beyondxia.modules:transform-plugin:1.0.4'

并在app module的build.gradle文件中引入该插件：

```
apply plugin: 'com.android.application'
apply plugin: 'com.beyondxia.modules.plugin'
```

apply plugin: 'com.beyondxia.modules.plugin'

2、新建module

在项目中新建一个名为modules-services-api（工程名一定要命名为modules-services-api）的android library性质的module（用来放置生成的公共源码），并在该module的build.gradle文件中添加如下依赖和属性配置。

```
dependencies {
    api 'com.beyondxia.modules:api:1.0.2'
    //...
}
```

api 'com.beyondxia.modules:api:1.0.2'
其他各个业务module均需要依赖此module。

3、添加annotationProcessor

在需要接口的业务module中添加如下annotationProcessor依赖

```
dependencies {
    annotationProcessor 'com.beyondxia.modules:compiler:1.0.3'
    //...
}
```

annotationProcessor ‘com.beyondxia.modules:compiler:1.0.3’

至此，已完成所有的配置工作。

二、使用步骤：

一、初始化

在Application类的onCreate中加入：
ServiceHelper.init(this);

二、提供服务：

以下提供一个典型的对外的服务提供类

```
@ExportService(moduleName = "business1", preLoad = true)
public class Login implements ILifeCycle{

    @ExportMethod
    public boolean doLogin(Context context, String userName, String password) {
        //...
    }

    @ExportMethod
    public void nav2LoginActivity(Context context) {
        //...
    }

    public void otherMethod(){

    }

    @Override
    public void onInstalled() {

    }

    @Override
    public void onUninstalled() {

    }

}
```

说明：

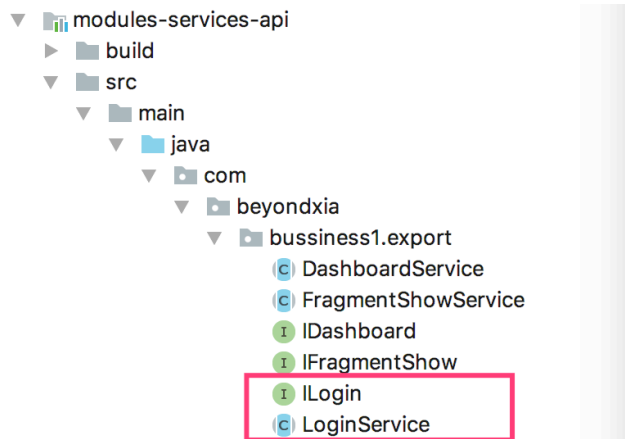
a、ExportService：标识对外提供服务的类

参数介绍：moduleName，该服务类所属的模块名，有默认值，可以不传，若传入，本模块内的所有服务类可相同（建议相同），但需保证与其他模块的互异性（建议不同，特殊情况下也可以相同）。

preLoad，是否预加载该服务类，默认为false。为true，则在启动app的时候则会实例化该服务类，否则在使用的时候再实例化，可根据当前的模块特性进行灵活配置。
b、ExportMethod：标识对外提供的方法，在对外提供的服务类中，只有被ExportMethod标记的方法才可以被其他模块所访问。注：该注解只能标识在为public的成员方法上，否则会抛出异常，终止构建。
c、ILifeCycle：服务类的生命周期，是否实现该接口为可选。若实现该接口，可在服务安装与卸载的回调中做一些初始化与资源释放等工作。
d、特别重要：该类不允许存在父类。

三、生成中间类：

完成以上服务类的配置工作，执行./gradlew :moduleName:compileDebugJavaWithJavac任务即可生成对应的服务中间类（对服务类进行修改变更的时候也需执行一下该任务以更新服务中间类），生成的中间类的目录为：rootProjectDir/modules-services-api/src/main/java



说明：服务中间类的生成规则：原服务类名为ClassName，则会生成对应的一个服务中间类与服务中间接口，明明规则为：ClassName+Service、I+ClassName。ClassNameService中有静态方法get()，通过调用该方法可以获取IClassName接口类型的对象，该对象即是原服务类的实例对象。

四、调用服务：

在其他模块中，以调用以上Login类中的方法为例，调用方式如下：

```
String username = LoginService.get().getUserName();
```

如上图对Login类的调用通过其中间服务类LoginService的调用即可，避免了对Login类的强依赖，达到了解耦的目的。