[SCA] Application生命周期分发SDK使用介绍

1.概述

目前组内大部分项目都是基于组件化编程，部分业务子Module需要在Application的生命周期中执行一些初始化、反初始化的操作，以前的做法是：在壳【app】module中定义一个顶层的Application类，在这个类中编写业务子Module的初始化、反初始化等代码，这造成了壳Module与业务子Module之间的耦合。通过该组件可以解耦，将Application的生命周期分发给各个需要的子Module，子Module进行相应的业务处理。

demo工程：corelib工程中的:lifecycledispatch:sample

2.功能介绍

分发Application的生命周期至各个子Module中，并支持在编译期间检查清单文件中的注册信息是否正确，若不正确则终止编译流程提前发现错误。

3.使用方式

3.1 添加依赖和配置

3.1.1 添加对插件的依赖

```
//build.gradle
buildscript {
    repositories {
        maven { url "https://maven.garenanow.com/nexus/content/groups/public/" }
    }
    dependencies {
        classpath 'com.shopee.sc:lifecycle-dispatch-plugin:x.x.x' //x.x.x,
    }
}


//modulebuild.gradleapp modulebuild.gradle
apply plugin: 'com.shopee.sc.lifecycledispatch'
```

说明：只需要在主module级别的build.gradle文件中使用“apply plugin”应用该插件即可。

3.1.2 添加对组件sdk的依赖

```
//build.gradle
allprojects {
    repositories {
        maven { url "https://maven.garenanow.com/nexus/content/groups/public/" }
        ...
    }
}

//Applicationmodulebuild.gradlesdk
dependencies {
    //x.x.x
    implementation 'com.shopee.sc:lifecycle-dispatch:x.x.x'
}
```

说明：第一步插件的版本和第二步sdk的版本需要保持一致。

3.2 代码实现部分

3.2.1 实现接口并添加注解

在需要监听Application生命周期的业务子module中实现IApplicationLifecycle接口并添加注解@ApplicationLifecycle。

```
@ApplicationLifecycle
public class Module1ClassA implements IApplicationLifecycle {

    //onCreate()
    @Override
    public void onCreate(Context applicationContext, boolean debug) {
        Log.i("LifecycleDispatch","Module1ClassA onCreate");
    }


    //
    @Override
    public void onConfigurationChanged(@NonNull Configuration newConfig) {

    }

    @Override
    public void onTerminate() {

    }

    @Override
    public void onTrimMemory(int level) {

    }

    @Override
    public void onLowMemory() {

    }

        //sdk
    @Override
    public int priority() {
        return 0;
    }
}
```

3.2.2 修改Application的onCreate()

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
                ApplicationLifecycleManager.getInstance(this).onCreate(this, BuildConfig.DEBUG);
    }
}
```

说明：接入者在接入时只需要在Application的onCreate()中分发onCreate()生命周期即可。原因如下：

1.Application涉及五个生命周期，分别是：onCreate()、onTerminate()、onLowMemory()、onConfigurationChanged()、onTrimMemory()

2.SDK内部会自动分发后三个生命周期，即onLowMemory()、onConfigurationChanged()、onTrimMemory()会被自动分发，接入者不用关系

3.由官方注释可以看出，onTerminate()生命周期方法只会在模拟器中调用，真机不会调用，所以该方法实际可不用分发，这里只是保留。若接入者还是想分发，重写Application的onTerminate()分发即可

3.2.3 在清单文件中注册

打开子module的Manifest.xml清单文件，在<applicaiton>标签中通过<meta-data>添加配置信息，name为类的全限定名，value固定写为“com.shopee.sc.LifecycleDispatch”

```
<application>
        <meta-data
            android:name="com.shopee.sc.lifecycledispatch.module1.Module1ClassA"
            android:value="com.shopee.sc.LifecycleDispatch" />
</application>
```

4 其他

- 插件目前只扫描“com.shopee”包下的代码（主要基于以下考虑：1.目前组内所有项目的业务代码基本都在该包下，基本可以实现全覆盖，不存在遗漏的class文件 2.限制包名可以减少扫描的类数量，节约编译时间）
- 如果不想在编译期间应用该插件，可在主module级别的**gradle.properties文件中通过enableLifecycleDispatchPlugin=false关闭，关闭后由开发者自己保证清单文件中注册信息的正确性。**