[SCA] 多进程框架使用介绍

1.概述

当项目使用多进程运行时，可以考虑使用该框架，将很方便的解决进程之间的通信问题。

demo工程：corelib工程中的:lightlyipc:lightlyipc-sample

2.功能介绍

支持进程间的同步和异步调用

跨模块API调用，内部通过动态代理实现组件解耦

支持指定服务运行的进程

同一个接口的不同实现类支持运行在相同/不同的进程中

不同接口的实现类支持运行在相同/不同的进程中

支持接口的多级继承

3.使用方式

3.1 添加依赖和配置

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

//moduleappbuild.gradle
dependencies {
    //x.x.xapicompiler
    implementation 'com.shopee.sc.lightlyipc:lightlyipc-api:x.x.x'
    annotationProcessor 'com.shopee.sc.lightlyipc:lightlyipc-compiler:x.x.x'
    ...
}
```

3.2 暴露跨进程服务

声明接口(即跨进程的服务)，需继承接口ILightlyIPCProvider.java

```
public interface INetMonitor extends ILightlyIPCProvider {
        //
    String sayHello(String name);

        //
        void sayHello(String name, IPCCallback ipcCallback);
}
```

3.3 实现该接口，并加上相应注解

```
@LightlyIPC(authority = "com.shopee.sc.provider1", process = "remote1")
public class NetMonitorImpl implements INetMonitor {
    private Context mContext;

    @Override
    public void init(Context context) {
        mContext = context;
    }

        @Override
    public String sayHello(String name) {
            return "hello, " + name;
    }

        @Override
    public void sayHello(String name, IPCCallback ipcCallback) {
        Bundle bundle = new Bundle();
        bundle.putString("name", "hello, " + name);
        try {
            ipcCallback.callback(bundle);
        } catch (RemoteException e) {
            LogUtil.e(Const.TAG_REMOTE, e);
        }
    }
}
```

init(Context context)方法是ILightlyIPCProvider接口中声明的方法，将在ContentProvider的生命周期onCreate()中调用；

一个authority将对应一个接口实现类，发现服务时也是通过authority来区分不同服务的；

一个process将会自动生成一个ContentProvider()与之对应，且这个ContentProvider需要在manifest.xml中注册，如何注册可参考3.5小节；

自动生成的ContentProvider的命名规则如下：

process名称中有“.”：“ContentProvider”+process名称（process名称首字母大写）

process名称中没有“.”：“ContentProvider”+截取process最后一个“.”之后的字符串（截取的字符串首字母转为大写）

3.4 使用该服务

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener{
    private ActivityMainBinding mViewBinding;

    @LightlyIPCWired(authority = “com.shopee.sc.provider1”)
    public INetMonitor mNetMonitor1;

    public INetMonitor mNetMonitor2;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mViewBinding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(mViewBinding.getRoot());

                // 1.@LightlyIPCWiredMainActivityLightlyIPC
        MainActivityLightlyIPC.inject(this, this);

                // 2.api
                mNetMonitor2 = LightIPC.getProvider(this,INetMonitor.class,“com.shopee.sc.provider1”);

        mViewBinding.btn1.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.btn1) {
            clickedBtn1();
        }
    }

    private void clickedBtn1() {
                //
        String result1 = mNetMonitor1.sayHello("test1");
        Log.i(TAG, " result1:" + result1);


        String result2 = mNetMonitor2.sayHello("test2");
        Log.i(TAG, " result2:" + result2);

                //
                mNetMonitor2.sayHello("test", new IPCCallback.Stub() {
            @Override
            public void callback(Bundle bundle) throws RemoteException {
                String result = bundle.getString("name");
                Log.i(TAG, " result:" + result);
            }
        });
        }
}
```

以上分别介绍了以同步和异步调用的方式使用该框架进行通信。

受跨进程的限制，方法参数和返回值支持的类型为：

- 同步调用时方法参数和返回值只支持基本数据类型、Serializable、Parcelable类型；
- 异步调用时方法返回值、方法参数和同步调用支持的类型一致(异步调用也支持有返回值)，异步回调IPCCallback.callback(Bundle bundle)参数只支持一个参数，其类型为Bundle；

3.5 清单文件注册

自动生成的ContentProvider需要在manifest.xml中注册

```
<application>
        <provider
                android:name="com.shopee.sc.lightlyipc.cp.ContentProviderRemote1"
        android:authorities="com.shopee.sc.provider1"
                android:exported="false"
        android:process=":remote1" />
</application>
```

3.6 其他用法

同一个接口的不同实现类、不同接口的不同实现类：

- 将对应不同的authority，process可以相同，也可以不同；
- 一个process就是一个进程，会有一个自动生成的ContentProvider与之对应；
- 如果一个process中有多个服务，即多个authority，在manifest.xml中注册时代码类似如下：

```
<application>
        <provider
                android:name="com.shopee.sc.lightlyipc.cp.ContentProviderRemote1"
        android:authorities="com.shopee.sc.provider1;com.shopee.sc.provider2"
                android:exported="false"
        android:process=":remote1" />
</application>
```