MICROOH 麦可网

Android-从程序员到架构师之路

出品人：Sundy
讲师：高焕堂（台湾）

http://www.microoh.com

B05_d

IPC的Proxy-Stub设计模式(d)

By 高煥堂

5、谁来写Proxy及Stub类呢?

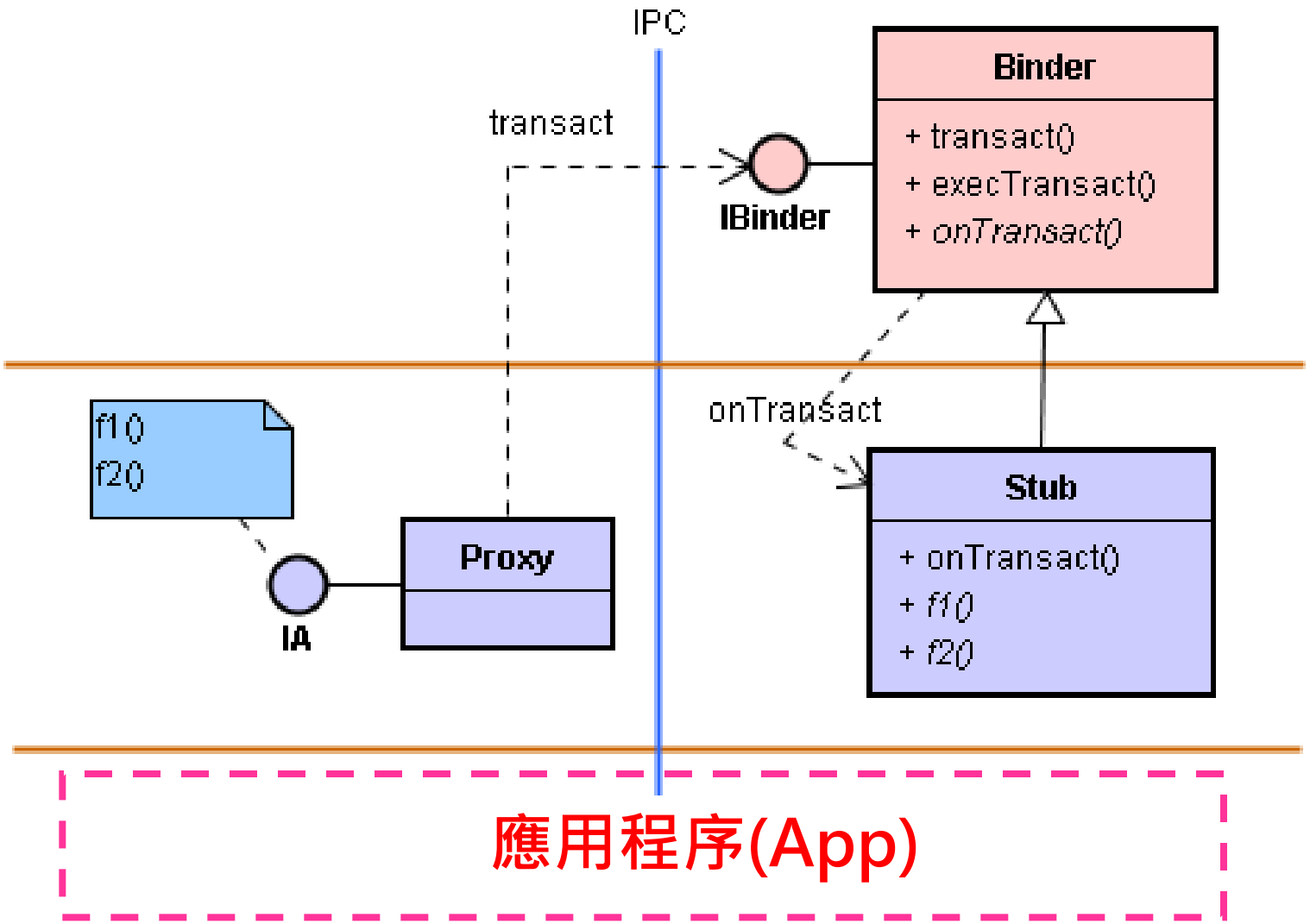
-- 强龙提供AIDL工具，
给地头蛇产出Proxy和Stub类

如何考虑<人>的<分工>

- 由框架开发者来撰写Proxy-Stub类，才能减轻开发者的负担。
- 框架分为：<天子框架>和<曹操框架>。
- 因此，**应该由两者(天子或曹操)之一来撰写Proxy-Stub类。**

IA接口知识取得的难题

- 但是，有个难题：IA接口(如下图所示)的内容必须等到<买主>来了才会知道。
- 在框架开发阶段，买主还没来，IA接口的知识无法取得，又如何定义IA接口呢? 没有IA接口定义，又如何撰写Stub和Proxy类呢?


IPC
transact
IBinder
Binder
+ transact()
+ execTransact()
+ onTransact()
onTransact
Stub
+ onTransact()
+ f1()
+ f2()
f1()
f2()
IA
Proxy
應用程序(App)

- 好办法是：

“强龙(天子或曹操)撰写代码(在先)；然后，地头蛇(App开发者)定义接口(在后)。”

在编程(Programming)上，有什么技术可以实现这个办法呢?

- 技术之一是：類別模板(class template)
  例如，强龙撰写模板：

```
template< class T >
class SomeClass
{
      private:
            T data;
      public:
            SomeClass() { }
            void set(T da)
                  { data = da; }
 };
```

- 地头蛇利用模板来生成一个类：

  SomeClass<Integer> x;

- 由于接口(interface)是一种特殊的类(class)，所以也可以定义模板如下：

```
template<interface I>
class BinderProxy
    {
        // ……
    };
```

- 地头蛇利用模板来生成一个类：

  BinderProxy<IPlayer> proxy;

- 除了模板之外，还有其它编程技术可以实现<强龙写代码，地头蛇定义接口>的方案吗?

- 答案是：

  程序生成器(program generator)

  例如：Android的aidl.exe


AIDL

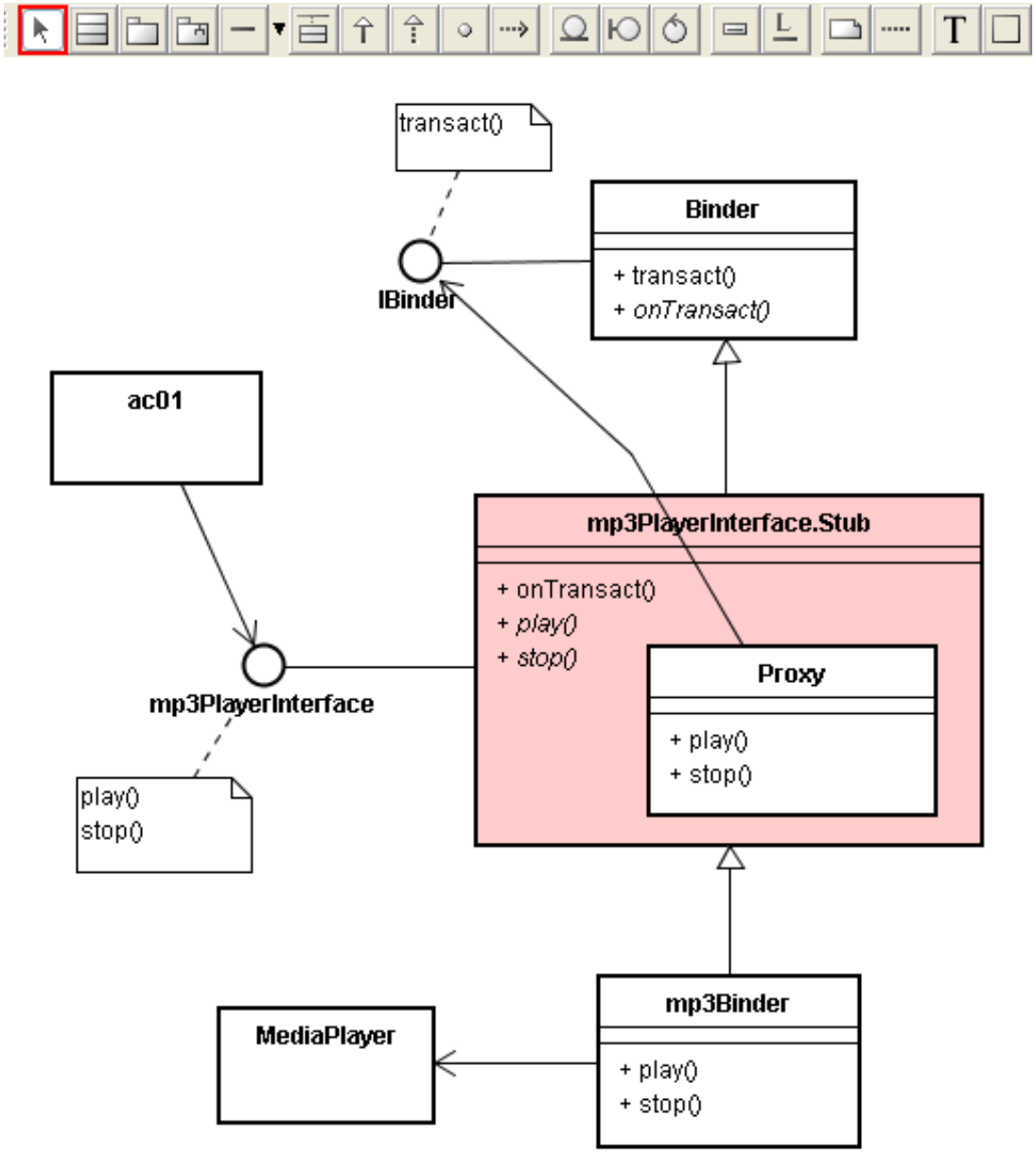
- AIDL的目的是定义Proxy/Stub来封装IBinder接口，以便产生更亲切贴心的新接口。
- 所以，在应用程序里，可以选择使用IBinder接口，也可以使用AIDL来定义出新接口。

- 由于IBinder接口只提供单一函数(即transact()函数)来进行远距通信，呼叫起来比较不方便。
- 所以Android提供aidl.exe工具来协助产出Proxy和Stub类别，以化解这个困难。

- 只要你善于使用开发环境的工具(如Android的aidl.exe软件工具)自动产生Proxy和Stub类别的程序代码；那就很方便了。

范例

- 此范例使用Android-SDK的/tools/里的aidl.exe工具程序，根据接口定义档(如下述的mp3PlayerInterface.aidl)而自动产出Proxy及Stub类别，其结构如下：


transact()
Binder
+ transact()
+ onTransact()
IBinder
ac01
mp3PlayerInterface.Stub
+ onTransact()
+ play()
+ stop()
Proxy
+ play()
+ stop()
mp3PlayerInterface
play()
stop()
mp3Binder
+ play()
+ stop()
MediaPlayer

- 藉由开发工具自动产出Proxy及Stub类的代码，再分别转交给ac01和mp3Binder开发者。此范例程序执行时，出现画面如下：

- 依据UI画面的两项功能：<Play>和< Stop>，以Java定义接口，如下的代码：

```
// mp3PlayerInterface.aidl
interface mp3PlayerInterface mp3PlayerInterface{
        void play();
        void stop();
}
```

- 使用Android-SDK所含的aidl.exe工具，將上述的mp3PlayerInterface.aidl档翻译成为下述的mp3PlayerInterface.java档案。

```
// mp3PlayerInterface.java
/*
 * This file is auto-generated.  DO NOT MODIFY.
 * Original file: mp3PlayerInterface.aidl
 */
// ……
public interface mp3PlayerInterface extends android.os.IInterface
{
/** Local-side IPC implementation stub class. */
public static abstract class Stub extends android.os.Binder
implements com.misoo.pkgx.mp3PlayerInterface
{
// …….
public boolean onTransact(int code, android.os.Parcel data,
android.os.Parcel reply, int flags) throws android.os.RemoteException
{
switch (code){
case INTERFACE_TRANSACTION:{
reply.writeString(DESCRIPTOR);
return true;
}
```

```
case TRANSACTION_play:{
data.enforceInterface(DESCRIPTOR);
this.play();
reply.writeNoException();
return true;
}
case TRANSACTION_stop:{
data.enforceInterface(DESCRIPTOR);
this.stop();
reply.writeNoException();
return true;
}}
return super.onTransact(code, data, reply, flags);
}
```

```
private static class Proxy implements
com.misoo.pkgx.mp3PlayerInterface
{
private android.os.IBinder mRemote;
//……
public void play() throws android.os.RemoteException
{
android.os.Parcel _data = android.os.Parcel.obtain();
android.os.Parcel _reply = android.os.Parcel.obtain();
try {
_data.writeInterfaceToken(DESCRIPTOR);
mRemote.transact(Stub.TRANSACTION_play, _data, _reply, 0);
_reply.readException();
}
finally {
_reply.recycle();
_data.recycle();
}
}
```

```
public void stop() throws android.os.RemoteException
{
android.os.Parcel _data = android.os.Parcel.obtain();
android.os.Parcel _reply = android.os.Parcel.obtain();
try {
_data.writeInterfaceToken(DESCRIPTOR);
mRemote.transact(Stub.TRANSACTION_stop, _data, _reply, 0);
_reply.readException();
}
finally {
_reply.recycle();
_data.recycle();
}}}
static final int TRANSACTION_play =
(IBinder.FIRST_CALL_TRANSACTION + 0);
static final int TRANSACTION_stop =
(IBinder.FIRST_CALL_TRANSACTION + 1);
}
public void play() throws android.os.RemoteException;
public void stop() throws android.os.RemoteException;
}
```

- 表面上，此mp3PlayerInterface.java是蛮复杂的，其实它的结构是清晰又简单的，只要对于类继承、反向調用和接口等面向对象观念有足够的认识，就很容易理解了。

```
// mp3Binder.java
package com.misoo.pkgx;
import android.content.Context;
import android.media.MediaPlayer;
import android.util.Log;
public class mp3Binder extends mp3PlayerInterface.Stub{
        private MediaPlayer mPlayer = null;
        private Context ctx;
        public mp3Binder(Context cx){    ctx= cx;       }
        public void play(){
            if(mPlayer != null) return;
            mPlayer = MediaPlayer.create(ctx, R.raw.test_cbr);
            try {  mPlayer.start();
            } catch (Exception e)
                    {   Log.e("StartPlay", "error: " + e.getMessage(), e);  }
          }
        public void stop(){
           if (mPlayer != null)
                  { mPlayer.stop();  mPlayer.release();  mPlayer = null; }
        }
}
```

撰写mp3RemoteService类

```
// mp3Service.java
package com.misoo.pkgx;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

public class mp3Service extends Service {
    IBinder ib = null;
    @Override public void onCreate() {
            super.onCreate();
            ib = new mp3Binder(this.getApplicationContext());
            }
    @Override      public void onDestroy() { }
    @Override      public IBinder onBind(Intent intent) {return ib;}
}
```

```
// ac01.java
// ……
public class ac01 extends Activity implements OnClickListener {
    //…….
    private PlayerProxy pProxy = null;
    public void onCreate(Bundle icicle) {
    // ……
          startService(new Intent("com.misoo.pkgx.REMOTE_SERVICE"));
          bindService(new Intent("com.misoo.pkgx.REMOTE_SERVICE"),
                                  mConnection, Context.BIND_AUTO_CREATE);
            }
    private ServiceConnection mConnection = new ServiceConnection() {
    public void onServiceConnected(ComponentName className,
                        IBinder ibinder) {
            pProxy = mp3PlayerInterface.Stub.asInterface(ibinder);
          }
       public void onServiceDisconnected(ComponentName className) {}
     };
```

```
public void onClick(View v) {
    switch (v.getId()) {
      case 101:  pProxy.play();  tv.setText(pProxy.getStatus());  break;
      case 102:  pProxy.stop();  tv.setText(pProxy.getStatus());  break;
      case 103:
            unbindService(mConnection);
            stopService(new Intent(
                        "com.misoo.pkgx.REMOTE_SERVICE"));
            finish();  break;
        }
    }
}
```

- 对于Anrdoid的初学者而言，Android的AIDL机制可说是最难弄懂的。

Thanks...

高煥堂