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

A10_b

介绍Android的Java层应用框架(b)

By 高煥堂

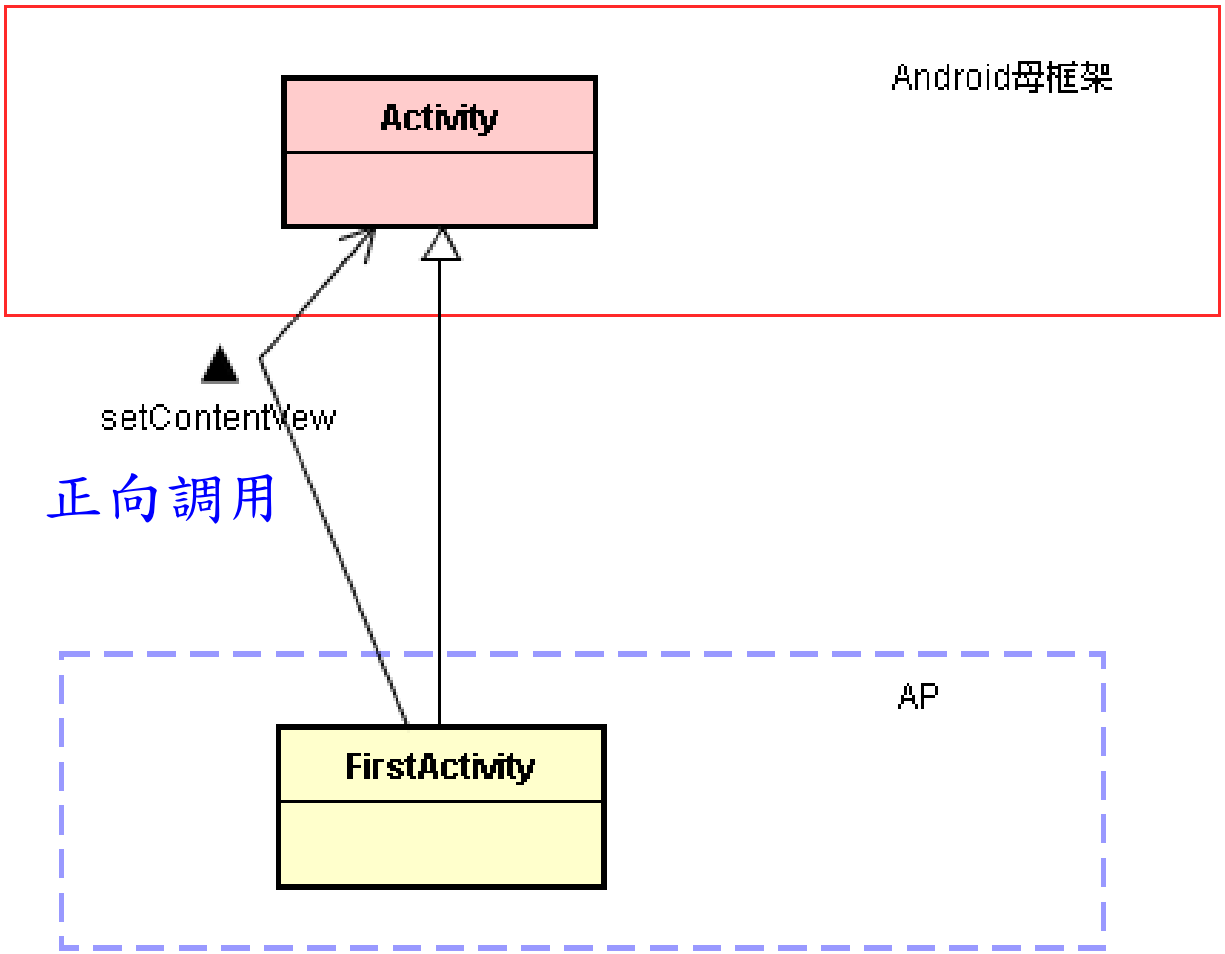
4、框架的控制机制

把握好莱坞大明星原则：

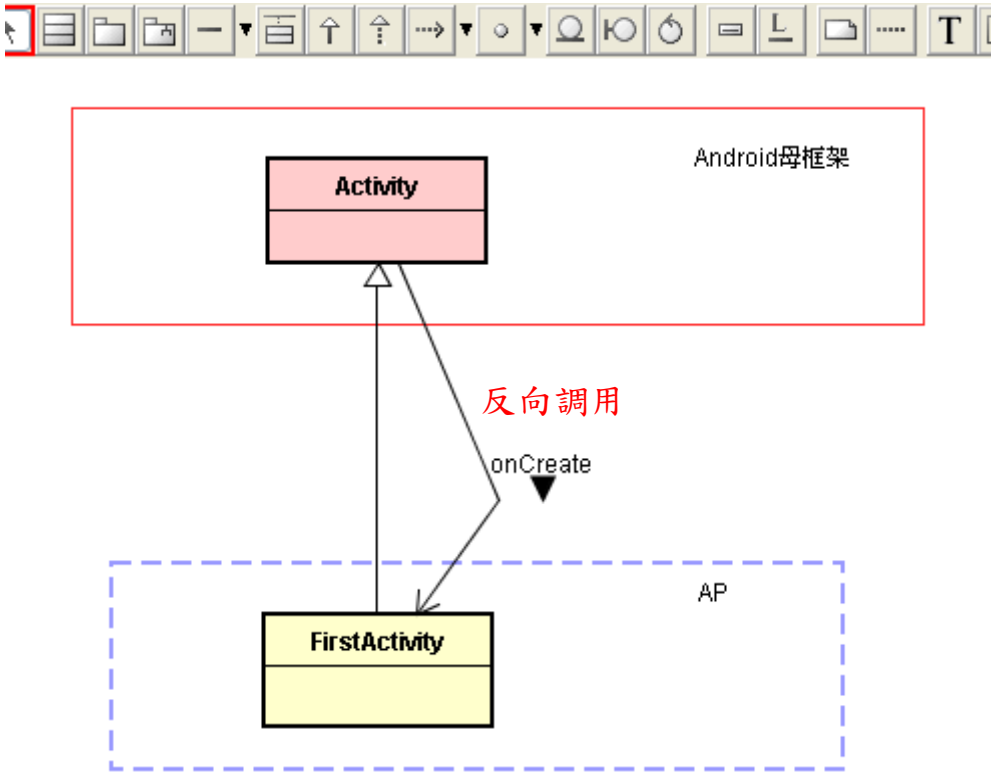
Don’t call me, I’ll call you back!

- 顾名思义，IoC(Inversion of Control)就是「反向控制」之意思。IoC观念和机制源自于OOP语言(如C++、Java等)的<基类/子类>结构。
- 例如Java语言中，基类(Super-class)的函数可以主动呼叫子类(Subclass)之函数，这就是一般所谓的IoC机制。后来，人们常将这些基类聚集起来，就称之为框架(Framework)。

- IoC又称为「反向呼叫」或「反向调用」。而反向调用的相反词就是：正向调用。正向调用就是App子类调用基类的函数。
- 例如，下图里的FirstActivity调用Activity基类的setContentView()函数。


Android母框架
Activity
setContentView
正向調用
AP
FirstActivity

- 反向调用恰好相反，表示由基类调用子类的函数。例如，下图里的Activity调用FirstActivity应用子类的onCreate()函数。

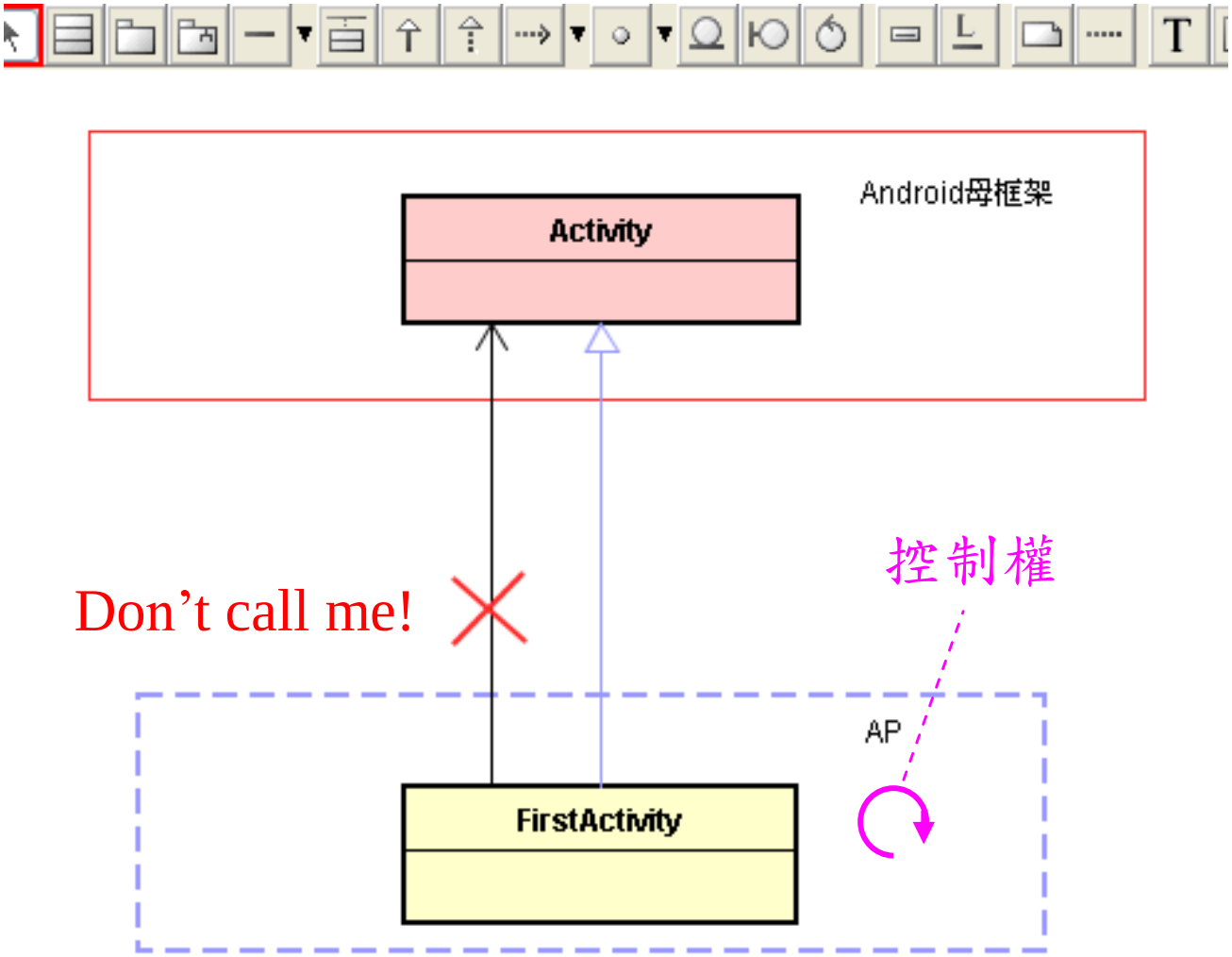


- 正向调用就意味着正向控制，也就是说，AP 子类控制了框架基类，这是违背框架设计原则的。这项设计原则就如同

  Hollywood(好莱坞)大明星的名言：

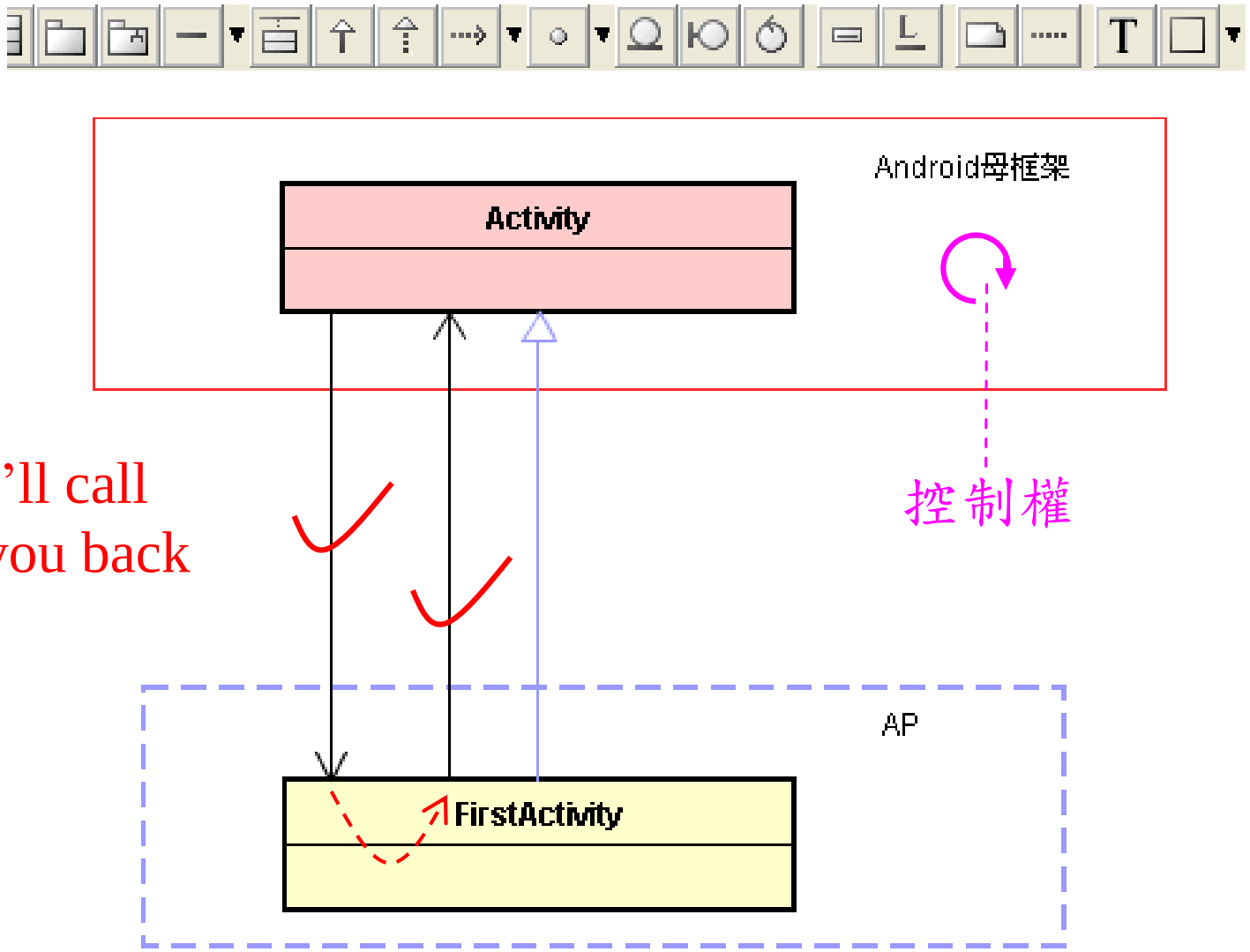
  “Don’t call me, I’ll call you back.”

如下图：


Android母框架
Activity
Don’t call me!
控制權
AP
FirstActivity

- 上图所示的正向控制，既违背了好莱坞大明星的原则，也违背框架设计的原则。
- 那么，该如何修正呢？改为反向控制就行了。如下图：

反向调用：
框架掌握控制权


Android母框架
Activity
I'll call
you back
控制權
AP
FirstActivity

- 反向调用就意味着反向控制，也就是说，框架基类控制了AP 子类，这是符合框架设计原则的，也符合Hollywood(好莱坞)大明星的原则。请看看Java代码如何实现这项原则：

```
//  FirstActivity.java
// ……
public class FirstActivity extends Activity
                                    implements OnItemClickListener {
    ArrayList<Row> coll;
    @Override  public void onCreate(Bundle icicle) {
                //I (框架) call You
        super.onCreate(icicle);
        coll = new ArrayList<Row>();
        coll.add(new Row("MP3", R.drawable.mp3_icon));
        coll.add(new Row("MP4",R.drawable.mp4_icon) );
        coll.add(new Row("Exit", R.drawable.icon2));

        ListView lv = new ListView(this);
        lv.setAdapter(new myAdapter(this, coll));
        lv.setOnItemClickListener(this);
       setContentView(lv);
            // You(App) call me
      }
```

```
@Override
protected void onActivityResult( int request,
                                 int r, Intent data) {
      // I (框架) call You
   Intent intent = new Intent( FirstActivity.this,
                                 VideoActivity.class);
   startActivity(intent);
      // You(App) call me
}
```

```
public void onItemClick( AdapterView<?> arg0, View arg1,
                                    int arg2, long arg3) {
            // I (框架) call You
    if(arg2 == 0) {
        Intent intent = new Intent(FirstActivity.this, mp3Activity.class);
        startActivity(intent);
            // You(App) call me
        }
    else if(arg2 == 1){
        Intent intent = new Intent(FirstActivity.this, LoadActivity.class);
        intent.putExtra("resid", R.raw.nice);
        startActivityForResult(intent, 0);
            // You(App) call me
        }
    else if(arg2 == 2)
        finish();  // You(App) call me
    }
}
```

- 无论是.NET、iOS或Android框架的设计，都依循着这项基本原则，让框架掌握控制权，也让掌握框架者能成为强龙。

~ Continued ~