bigapple之asynctask部分

1、简介

asynctask部分，主要使用了模版方法，对asynctask类的封装，主要是简化了耗时操作的写法，原本的耗时操作需要在线程中执行，然后如果在

```
private Handler handler = new Handler();
......
new Thread() {
    @Override
    public void run() {
        String resultStr = null;
        try {
            resultStr = doHttpOperation();//Http
        } catch (Exception e) {
        }

        //UITextView
        handler.post(new Runnable(){
            @Override
            public void run(){
                 textView.setText(resultStr);//UIUI
            }
        };
    );}
}.start();
```

2、 asynctask部分的使用详解

（1）首先，定义一个类，继承AbstractTask。如下面的代码：

```
public class GetUserTask extends AbstractTask<String> {
    public GetUserTask(Context context,boolean isShow) {
        super(context,isShow);
    }

    @Override
    protected Result<String> doHttpRequest(Object... objects) {
        String resultStr = doHttpOperation();//Http
        Result<String> result = null;
        if(TextUtils.isEmpty(resultStr)){
            result = new Result(false,"");
        }else{
            result = new Result(false,"",resultStr);
        }

        return result;
    }
}
2ActivityTextView
//truefalse
GetUserTask task = new GetUserTask(this,true);

//TextView
task.setAsyncTaskSuccessCallback(new AsyncTaskSuccessCallback<String>() {
    @Override
    public void successCallback(Result<String> result) {
        String resultStr = result.getValue();
        textView.setText(resultStr);
    }
}

//,
task.AsyncTaskSuccessCallback(new AsyncTaskSuccessCallback<String>() {
    @Override
    public void successCallback(Result<String> result) {
    }
}

//
task.execute(new Object[] { accountId});
```

3、结尾

上面看上去，好像使用了框架代码更多了，其实不是，使用框架之后，可以使耗时操作（例如Http操作）和UI更新操作分离。使代码更加简洁