Android 项目重构之路：实现篇（下）

2015-11-22 安卓应用频道

(点击上方公众号，可快速关注)

> 来源：Keegan小钢
> 链接：http://keeganlee.me/post/android/20150629

（接上文）

本系列：
Android 项目重构之路：架构篇
Android 项目重构之路：界面篇

核心层的逻辑

核心层处于接口层和界面层之间，向下调用Api，向上提供Action，它的核心任务就是处理复杂的业务逻辑。先看看我对Action的定义：

```
public interface AppAction {
  // 发送手机验证码
  public void sendSmsCode(String phoneNum, ActionCallbackListenerVoid> listener);
  // 注册
  public void register(String phoneNum, String code, String password, ActionCallbackListenerVoid> listener);
  // 登录
  public void login(String loginName, String password, ActionCallbackListenerVoid> listener);
  // 按分页获取券列表
  public void listCoupon(int currentPage, ActionCallbackListenerListCouponBO>> listener);
}
```

首先，和Api接口对比就会发现，参数并不一致。登录并没有iemi和loginOS的参数，获取券列表的参数里也少了pageSize。这是因为，这几个参数，跟界面其实并没有直接关系。

Action只要定义好跟界面相关的就可以了，其他需要的参数，在具体实现时再去获取。

另外，大部分action的处理都是异步的，因此，添加了回调监听器ActionCallbackListener，回调监听器的泛型则是返回的对象数据类型，例如获取券列表，返回的数据类型就是List，没有对象数据时则为Void。回调监听器只定义了成功和失败的方法，如下：

```
public interface ActionCallbackListenerT> {
  /**
   * 成功时调用
   *
   * @param data 返回的数据
   */
  public void onSuccess(T data);

 /**
   * 失败时调用
   *
   * @param errorEvemt 错误码
   * @param message    错误信息
   */
  public void onFailure(String errorEvent, String message);
}
```

接下来再看看Action的实现。首先，要获取imei，那就需要传入一个Context；另外，还需要loginOS和pageSize，这定义为常量就可以了；还有，要调用接口层，所以还需要Api实例。而接口的实现分为两步，第一步做参数检查，第二步用异步任务调用Api。具体实现如下：

```
public class AppActionImpl implements AppAction {
  private final static int LOGIN_OS = 1; // 表示Android
  private final static int PAGE_SIZE = 20; // 默认每页20条

  private Context context;
  private Api api;

  public AppActionImpl(Context context) {
```

```
    this.context = context;
    this.api = new ApiImpl();
}

@Override
public void sendSmsCode(final String phoneNum, final ActionCallbackListenerVoid> listener) {
    // 参数为空检查
    if (TextUtils.isEmpty(phoneNum)) {
        if (listener != null) {
            listener.onFailure(ErrorEvent.PARAM_NULL, "手机号为空");
        }
        return;
    }
    // 参数合法性检查
    Pattern pattern = Pattern.compile("1\d{10}");
    Matcher matcher = pattern.matcher(phoneNum);
    if (!matcher.matches()) {
        if (listener != null) {
            listener.onFailure(ErrorEvent.PARAM_ILLEGAL, "手机号不正确");
        }
        return;
    }

    // 请求Api
    new AsyncTaskVoid, Void, ApiResponseVoid>>() {
        @Override
        protected ApiResponseVoid> doInBackground(Void... voids) {
            return api.sendSmsCode4Register(phoneNum);
        }

        @Override
        protected void onPostExecute(ApiResponseVoid> response) {
            if (listener != null & response != null) {
                if (response.isSuccess()) {
                    listener.onSuccess(null);
```

```
                } else {
                    listener.onFailure(response.getEvent(), response.getMsg());
                }
            }
        }
    }.execute();
}

@Override
public void register(final String phoneNum, final String code, final String password, final
ActionCallbackListenerVoid> listener) {
    // 参数为空检查
    if (TextUtils.isEmpty(phoneNum)) {
        if (listener != null) {
            listener.onFailure(ErrorEvent.PARAM_NULL, "手机号为空");
        }
        return;
    }
    if (TextUtils.isEmpty(code)) {
        if (listener != null) {
            listener.onFailure(ErrorEvent.PARAM_NULL, "验证码为空");
        }
        return;
    }
    if (TextUtils.isEmpty(password)) {
        if (listener != null) {
            listener.onFailure(ErrorEvent.PARAM_NULL, "密码为空");
        }
        return;
    }

    // 参数合法性检查
    Pattern pattern = Pattern.compile("1\d{10}");
    Matcher matcher = pattern.matcher(phoneNum);
```

```
    if (!matcher.matches()) {
        if (listener != null) {
            listener.onFailure(ErrorEvent.PARAM_ILLEGAL, "手机号不正确");
        }
        return;
    }

    // TODO 长度检查，密码有效性检查等

    // 请求Api
    new AsyncTaskVoid, Void, ApiResponseVoid>>() {
        @Override
        protected ApiResponseVoid> doInBackground(Void... voids) {
            return api.registerByPhone(phoneNum, code, password);
        }

        @Override
        protected void onPostExecute(ApiResponseVoid> response) {
            if (listener != null & response != null) {
                if (response.isSuccess()) {
                    listener.onSuccess(null);
                } else {
                    listener.onFailure(response.getEvent(), response.getMsg());
                }
            }
        }
    }.execute();
}

@Override
public void login(final String loginName, final String password, final ActionCallbackListenerVoid> listener) {
    // 参数为空检查
    if (TextUtils.isEmpty(loginName)) {
        if (listener != null) {
            listener.onFailure(ErrorEvent.PARAM_NULL, "登录名为空");
```

```
            }
            return;
        }
        if (TextUtils.isEmpty(password)) {
            if (listener != null) {
                listener.onFailure(ErrorEvent.PARAM_NULL, "密码为空");
            }
            return;
        }

        // TODO 长度检查，密码有效性检查等

        // 请求Api
        new AsyncTaskVoid, Void, ApiResponseVoid>>() {
            @Override
            protected ApiResponseVoid> doInBackground(Void... voids) {
                TelephonyManager telephonyManager = (TelephonyManager)
context.getSystemService(Context.TELEPHONY_SERVICE);
                String imei = telephonyManager.getDeviceId();
                return api.loginByApp(loginName, password, imei, LOGIN_OS);
            }

            @Override
            protected void onPostExecute(ApiResponseVoid> response) {
                if (listener != null & response != null) {
                    if (response.isSuccess()) {
                        listener.onSuccess(null);
                    } else {
                        listener.onFailure(response.getEvent(), response.getMsg());
                    }
                }
            }
        }.execute();
    }
```

```
    @Override
    public void listCoupon(final int currentPage, final ActionCallbackListener<ListCouponBO>> listener) {
        // 参数检查
        if (currentPage  0) {
            if (listener != null) {
                listener.onFailure(ErrorEvent.PARAM_ILLEGAL, "当前页数小于零");
            }
        }

        // TODO 添加缓存

        // 请求Api
        new AsyncTask<Void, Void, ApiResponse<ListCouponBO>>>() {
            @Override
            protected ApiResponse<ListCouponBO>> doInBackground(Void... voids) {
                return api.listNewCoupon(currentPage, PAGE_SIZE);
            }

            @Override
            protected void onPostExecute(ApiResponse<ListCouponBO>> response) {
                if (listener != null & response != null) {
                    if (response.isSuccess()) {
                        listener.onSuccess(response.getObjList());
                    } else {
                        listener.onFailure(response.getEvent(), response.getMsg());
                    }
                }
            }
        }.execute();
    }
}
```

简单的实现代码就是这样，其实，这还有很多地方可以优化，比如，将参数为空的检查、手机号有效性的检查、数字型范围的检查等等，都可以抽成独立的方法，从而减少重复代码的编写。异步任务里的代码也一样，都是可以通过重构优化的。另外，需要扩展时，比如添加

缓存，那就在调用Api之前处理。
核心层的逻辑就是这样了。最后就到界面层了。

界面层

在这个Demo里，只有三个页面：登录页、注册页、券列表页。在这里，也会遵循界面篇提到的三个基本原则：规范性、单一性、简洁性。
首先，界面层需要调用核心层的Action，而这会在整个应用级别都用到，因此，Action的实例最好放在Application里。代码如下：

```
public class KApplication extends Application {

  private AppAction appAction;

  @Override
  public void onCreate() {
    super.onCreate();
    appAction = new AppActionImpl(this);
  }

  public AppAction getAppAction() {
    return appAction;
  }
}
```

另外，一个Activity的基类也是很有必要的，可以减少很多重复的工作。基类的代码如下：

```
public abstract class KBaseActivity extends FragmentActivity {
  // 上下文实例
  public Context context;
  // 应用全局的实例
  public KApplication application;
  // 核心层的Action实例
  public AppAction appAction;

  @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        context = getApplicationContext();
        application = (KApplication) this.getApplication();
        appAction = application.getAppAction();
    }
}
```

再看看登录的Activity：

```
public class LoginActivity extends KBaseActivity {

    private EditText phoneEdit;
    private EditText passwordEdit;
    private Button loginBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        // 初始化View
        initViews();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_login, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        // 如果是注册按钮
        if (id == R.id.action_register) {
```

```
        Intent intent = new Intent(this, RegisterActivity.class);
        startActivity(intent);
        return true;
    }

    return super.onOptionsItemSelected(item);
}

// 初始化View
private void initViews() {
    phoneEdit = (EditText) findViewById(R.id.edit_phone);
    passwordEdit = (EditText) findViewById(R.id.edit_password);
    loginBtn = (Button) findViewById(R.id.btn_login);
}

// 准备登录
public void toLogin(View view) {
    String loginName = phoneEdit.getText().toString();
    String password = passwordEdit.getText().toString();
    loginBtn.setEnabled(false);
    this.appAction.login(loginName, password, new ActionCallbackListener<Void>() {
        @Override
        public void onSuccess(Void data) {
            Toast.makeText(context, R.string.toast_login_success, Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(context, CouponListActivity.class);
            startActivity(intent);
            finish();
        }

        @Override
        public void onFailure(String errorEvent, String message) {
            Toast.makeText(context, message, Toast.LENGTH_SHORT).show();
            loginBtn.setEnabled(true);
        }
    });
```

```
    }
}
```

登录页的布局文件则如下：

```
LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.keegan.kandroid.activity.LoginActivity">

    EditText
        android:id="@+id/edit_phone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/edit_vertical_margin"
        android:layout_marginBottom="@dimen/edit_vertical_margin"
        android:hint="@string/hint_phone"
        android:inputType="phone"
        android:singleLine="true" />

    EditText
        android:id="@+id/edit_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/edit_vertical_margin"
        android:layout_marginBottom="@dimen/edit_vertical_margin"
        android:hint="@string/hint_password"
        android:inputType="textPassword"
```

```
    android:singleLine="true" />

  Button
    android:id="@+id/btn_login"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/btn_vertical_margin"
    android:layout_marginBottom="@dimen/btn_vertical_margin"
    android:onClick="toLogin"
    android:text="@string/btn_login" />
LinearLayout>
```

可以看到，EditText的id命名统一以edit开头，而在Activity里的控件变量名则以Edit结尾。按钮的onClick也统一用toXXX的方式命名，明确表明这是一个将要做的动作。还有，string，dimen也都统一在相应的资源文件里按照相应的规范去定义。

注册页和登陆页差不多，这里就不展示代码了。主要再看看券列表页，因为用到了ListView，ListView需要添加适配器。实际上，适配器很多代码都是可以复用的，因此，我抽象了一个适配器的基类，代码如下：

```
public abstract class KBaseAdapterT> extends BaseAdapter {

  protected Context context;
  protected LayoutInflater inflater;
  protected ListT> itemList = new ArrayListT>();

  public KBaseAdapter(Context context) {
    this.context = context;
    inflater = LayoutInflater.from(context);
  }

  /**
   * 判断数据是否为空
   *
   * @return 为空返回true，不为空返回false
```

```
     */
    public boolean isEmpty() {
        return itemList.isEmpty();
    }

    /**
     * 在原有的数据上添加新数据
     *
     * @param itemList
     */
    public void addItems(ListT> itemList) {
        this.itemList.addAll(itemList);
        notifyDataSetChanged();
    }

    /**
     * 设置为新的数据，旧数据会被清空
     *
     * @param itemList
     */
    public void setItems(ListT> itemList) {
        this.itemList.clear();
        this.itemList = itemList;
        notifyDataSetChanged();
    }

    /**
     * 清空数据
     */
    public void clearItems() {
        itemList.clear();
        notifyDataSetChanged();
    }

    @Override
```

```
    public int getCount() {
        return itemList.size();
    }

    @Override
    public Object getItem(int i) {
        return itemList.get(i);
    }

    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    abstract public View getView(int i, View view, ViewGroup viewGroup);
}
```

这个抽象基类集成了设置数据的方法，每个具体的适配器类只要再实现各自的getView方法就可以了。本Demo的券列表的适配器如下：

```
public class CouponListAdapter extends KBaseAdapter<CouponBO> {

    public CouponListAdapter(Context context) {
        super(context);
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        ViewHolder holder;
        if (view == null) {
            view = inflater.inflate(R.layout.item_list_coupon, viewGroup, false);
            holder = new ViewHolder();
            holder.titleText = (TextView) view.findViewById(R.id.text_item_title);
            holder.infoText = (TextView) view.findViewById(R.id.text_item_info);
            holder.priceText = (TextView) view.findViewById(R.id.text_item_price);
```

```
            view.setTag(holder);
        } else {
            holder = (ViewHolder) view.getTag();
        }

        CouponBO coupon = itemList.get(i);
        holder.titleText.setText(coupon.getName());
        holder.infoText.setText(coupon.getIntroduce());
        SpannableString priceString;
        // 根据不同的券类型展示不同的价格显示方式
        switch (coupon.getModelType()) {
            default:
            case CouponBO.TYPE_CASH:
                priceString = CouponPriceUtil.getCashPrice(context, coupon.getFaceValue(),
coupon.getEstimateAmount());
                break;
            case CouponBO.TYPE_DEBIT:
                priceString = CouponPriceUtil.getVoucherPrice(context, coupon.getDebitAmount(),
coupon.getMiniAmount());
                break;
            case CouponBO.TYPE_DISCOUNT:
                priceString = CouponPriceUtil.getDiscountPrice(context, coupon.getDiscount(), coupon.getMiniAmount());
                break;
        }
        holder.priceText.setText(priceString);

        return view;
    }

    static class ViewHolder {
        TextView titleText;
        TextView infoText;
        TextView priceText;
    }
```

```
}
```

而券列表的Activity简单实现如下：

```
public class CouponListActivity extends KBaseActivity implements SwipeRefreshLayout.OnRefreshListener {
  private SwipeRefreshLayout swipeRefreshLayout;
  private ListView listView;
  private CouponListAdapter listAdapter;
  private int currentPage = 1;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_coupon_list);

    initViews();
    getData();

    // TODO 添加上拉加载更多的功能
  }

  private void initViews() {
    swipeRefreshLayout = (SwipeRefreshLayout) findViewById(R.id.swipe_refresh_layout);
    swipeRefreshLayout.setOnRefreshListener(this);
    listView = (ListView) findViewById(R.id.list_view);
    listAdapter = new CouponListAdapter(this);
    listView.setAdapter(listAdapter);
  }

  private void getData() {
    this.appAction.listCoupon(currentPage, new ActionCallbackListenerListCouponBO>>() {
      @Override
      public void onSuccess(ListCouponBO> data) {
        if (!data.isEmpty()) {
          if (currentPage == 1) { // 第一页
            listAdapter.setItems(data);
```

```
            } else { // 分页数据
                listAdapter.addItems(data);
            }
        }
        swipeRefreshLayout.setRefreshing(false);
    }

    @Override
    public void onFailure(String errorEvent, String message) {
        Toast.makeText(context, message, Toast.LENGTH_SHORT).show();
        swipeRefreshLayout.setRefreshing(false);
    }
  });
 }

 @Override
 public void onRefresh() {
  // 需要重置当前页为第一页，并且清掉数据
  currentPage = 1;
  listAdapter.clearItems();
  getData();
 }
}
```

完结

终于写完了，代码也终于放上了github，为了让人更容易理解，因此很多都比较简单，没有再进行扩展。

> github地址：https://github.com/keeganlee/kandroid

安卓应用频道

微信号：**AndroidPD**

打造东半球最好的 安卓技术 微信号

商务合作QQ：2302462408

投稿网址：top.jobbole.com

微信扫一扫
关注该公众号