复习练习---请同步视频自己操作

（步骤视频有超链接直接 ctrl+点击可以观看）

（至少跟着练习 2 遍，再自己独立完成 1 遍，3 遍以上更佳）

完成一个图书管理 App 的功能，包括以下四大功能（第四个功能不做要求，视频也不提供）
（1）欢迎双页面滑动进入效果
（2）登录（远程访问 post 接口验证管理员账号，正确可以登录成功，否则登录失败，管理员账号为 book，密码为 5）
（3）登录成功后主界面以 ListView 清单显示所有的图书的名称，单价和数量（远程访问 get 接口得到 json 数组解析通过 BaseAdapter 显示在 ListView 清单上）
（4）最后点击清单页中的某个图书这一行将跳转到显示这个图书的信息的独立详情页，新的详情页显示图书名称，单价和数量信息。

1. **步骤 1：**创建一个工程，并且设置默认生成的首启动 Activity 为 BookWelActivity
（同步看视频-1.Android 工程的创建与默认 Activity 的命名）

2. **步骤 2：**创建工程所需的模拟器，版本设置为 6P 的 5.7 英寸的版本。
（同步看视频-2.模拟器的创建与启动）

3. **步骤 3：**为上面描述的功能创建 BookApp 工程的所有工程文件。
（同步看视频-3.图书 App 所需的功能的所有文件的创建和 Fragment 布局关联）
【代码复习：Fragment 的 java 文件和对应的布局 xml 文件的关联需要在 Fragment 的 java 类中，自动生成 onCreateView 方法，并且在 onCreateView 方法中删除原来的代码：改写以下代码替代

```
View view = inflater.inflate(R.layout.book_frag1,container,false);   //book_frag1 为需要关联的布局文件名
return view;
```

】

4. **步骤 4：实现双页面滑动的欢迎界面效果，并在第二个欢迎界面设置“立即体验”圆角按钮，点击跳转到主界面 BookMainActivity。**
（4-1）导入欢迎界面的背景图片资源，修改两个欢迎界面的 book_frag1 和 book_frag2 的布局为背景全屏图片，并且创建 FragmentStatePagerAdapter 的子类 BookAdapter，实现滑动页面通用适配器类代码。
（同步看视频-4-1.欢迎界面的图片资源导入，布局和适配器类实现）
【代码复习：Fragment 适配器类代码编写如下：

```
public class BookAdapter extends FragmentStatePagerAdapter {
    private ArrayList<Fragment> fragments;
    public BookAdapter(FragmentManager fm,ArrayList<Fragment> fragments) {
        super(fm);
```

```
            this.fragments = fragments;
        }
        @Override
        public Fragment getItem(int position) {
            return this.fragments.get(position);
        }
        @Override
        public int getCount() {
            return this.fragments.size();
        }
}
```

】

（4-2）在欢迎界面的 BookWelActivity 的布局文件 acitivty_book_wel 上添加一个 ViewPager 控件为满屏，设置它的 id 为 view_paper，在对应 BookWelActivity.java 文件中映射 viewpager 为 java 变量，同步创建 BookAdapter 适配器变量 adapter，创建的时候调用构造函数传递两个参数，第二个参数为构造出来的含有 BookFrag1 和 BookFrag2 的 ArrayList 数组，最后适配器变量 adapter 和 ViewPager 绑定。

（同步看视频-4-2 ViewPager 应用 BookAdapter 实现双页滑动效果）

【代码复习：BookWebActivity 的代码中需要映射 ViewPager 和适配器对象创建和 ListView 关联

```
public class BookWelActivity extends AppCompatActivity {
    private ViewPager viewPager;    //ViewPager 变量定义
    private ArrayList<Fragment> fragments=new ArrayList<Fragment>();    //适配器所需的 Fragment 动态数组定义和 new 分配内存
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_book_wel);
        viewPager = (ViewPager) findViewById(R.id.view_pager); //ViewPager 变量映射 xml 文件的 id
        fragments.add(new BookFrag1() );    //动态数组添加欢迎界面第 1 个 Fragment 变量
        fragments.add(new BookFrag2());    //动态数组添加欢迎界面第 2 个 Fragment 变量
        BookAdapter adapter = new BookAdapter(getSupportFragmentManager(),fragments);    //利用 Fragment 管理器和动态数组数据生成 Fragment 适配器
        viewPager.setAdapter(adapter);    //将适配器绑定到 viewPager 变量上
    }
}
```

】

（4-3）实现欢迎界面第二个页面 BookFrag2 上添加“立即体验”圆角按钮，并且实现点击跳转到图书管理 App 登录界面 BookLoginActivity。

（同步看视频-4-3 欢迎界面的第二页立即体验圆角按钮和无参跳转到图书管理 App 登录界面 BookLoginActivity）

【代码复习：

圆角按钮需要新建一个图片资源文件代码如下

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">      <!--修改标签为 shape-->
    <corners android:radius="10dp"/>      <!--添加 corners 标签表示圆角，设置圆角半径为 10dp-->
    <solid android:color="#ffffff"/>     <!--添加 solid 标签表示按钮填充色，这里设置白色-->
</shape>
```

最后在 book_frag1.xml 布局中增加一个 Button 标签并且设置正确的位置，然后设置它的 background 属性为上述的@drawable/圆角白色背景的文件名。

按钮的映射由于在 Fragment 中需要基于 view 做 view.findViewById

需要添加按钮的映射和单击事件代码

按钮名称.setOn 之后选择 OnClick 事件，然后在括号内填写 new On 自动生成 onClick 事件

在事件中添加无参跳转的代码

```
public class BookFrag2 extends Fragment {
    private Button btnStart;   //定义按钮变量
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.book_frag2,container,false);
        btnStart = (Button) view.findViewById(R.id.btn_start);   //基于整个 Fragment 生成的 view 做
        btnStart.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //无参跳转
                Intent intent = new Intent( getActivity()    , BookLoginActivity.class);
                startActivity(intent);
            }
        });
        return view;
    }
}
```

】

5. **步骤 5：实现图书管理 App 登录主界面账号密码输入和登录按钮的布局，并且实现访问网络 json 的 POST 接口，发送登录用户名和密码，返回结果。具体的接口说明如下，只有输入账号为 book 和密码为 1 才返回 result 的值为 ok，否则返回 result 的值为 failed。**

接口描述	用于管理员登录	
url 地址	http://study.smartye.top/api/shop/login	
访问方式	POST	
发包格式	{“username”:“book”,“password”:“5”}	
收包格式	登录成功	只有 book 和 5 才匹配返回 {result:“ok”}
	登录失败	非 book 和 5 的组合表示不一致则返回 {result:“failed”}

（5-1）设置整个 App 的全部配置文件 AndroidManifest.xml，添加权限允许访问网络（切记

这个不要忘记不然网络访问结果始终出不来）

（同步看视频-5-1 设置 App 允许 Intenet 网络访问权限）

【代码复习：在全局配置文件 AndroidManifest.xml 中的根标签内部添加代码

```
<uses-permission android:name="android.permission.INTERNET"/>
```

】

（5-2）添加 http 网络访问的第三方框架 okhttp 的两个 jar 包。

（同步看视频-5-2 okhttp 第三方框架 jar 包导入方法）

（5-3）实现 BookLoginActivity 的布局文件 activity_book_login 的账号密码输入和按钮登录效果并设置账号密码编辑框和登录按钮的 id 号。

（同步看视频-5-3 图书管理 App 的登录布局 activity_book__login 的登录界面布局）

（5-4）通过 okhttp 框架实现登录 post 接口的访问并得到 JSONObject 对象准备发送给 Handler。

（同步看视频-5-4 登录界面访问 POST 接口的 okhttp 网络访问)

【代码复习：okhttp 的 post 访问代码

```
//引用 okhttp 类进行网络访问可以免除一些线程的操作
MediaType JSON = MediaType.parse("application/json;charset=utf-8");   //这句话必不可少，表示媒体流是 json
OkHttpClient client = new OkHttpClient(); //  创建一个 okhttp 客户端
String json =
"{\"username\":\""+edtUser.getText().toString()+"\",\"password\":\""+edtPass.getText().toString()+"\"}";
//{“username”:”book”, “password”:”5”}是对发包格式的拆分，其中 book 和 5 是替换成两个控件 edtUser 和
edtPass 读取 string 内容，其余内容由于双引号里面又有双引号，所以里面的所有双引号都加入转义字符\
RequestBody body = RequestBody.create(JSON,json);//利用 JSON 媒体流说明和具体的 json 字符串生成包体
Request request = new Request.Builder().url("http://study.smartye.top/api/shop/login").post(body).build();
//执行 post 请求
Call call = client.newCall(request);//创建请求
//这是一个回调函数代码，call 的队列请求，这里隐含了线程，所以不需要自己创建线程
call.enqueue(new Callback() {
    @Override
    public void onFailure(Call call, IOException e) {

    }

    @Override
    public void onResponse(Call call, Response response) throws IOException {
        //在回应中书写代码，读取回调函数参数 response 的 body 里面的 string 得到回复的字符串
        String strJson= response.body().string();
        try {
            //将回复的字符串构造成 JSONObject 对象，发送给 Handler 在主线程解析判断
            JSONObject jsonObject = new JSONObject(strJson);
            Handler.obtainMessage(0,jsonObject).sendToTarget(); //
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
});
```

】

（5-5）在主线程最前面的属性部分创建 Handler 对象 handler 并实现 handleMessage 方法，在方法内部接收到的数据包，并且解析出来 result 这个 key 对应的 value，判断这个 value 字符串是否为 ok,比较字符串是否相等用字符串 1.equals(字符串 2)的返回结果是否为 true，如果是 true 则建议提示框登录成功跳转到图书清单主界面 BookMainActivity，否则简易提示框登录失败不做跳转。

（同步看视频-5-5 Handler 接收 JSONObject 对象解析 result 值判断是否为 ok 实现登录)

【代码复习：okhttp 的 Handler 处理传递过来的 json 包并且解析 result 这个 key 进行判断

```
Handler handler = new Handler()
{
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        if(msg.what==0)   //编码为 0 触发
        {
            JSONObject jsonObject = (JSONObject)(msg.obj);   //把相关的 obj 取出来转化成 JSONObject
            try {
                String result = jsonObject.getString("result"); //  解析里面的 result 这个 key 的值
                if(result.equals("ok"))   //判断两个字符串是否相等用.equals，如果是 ok，登录成功跳转
                {
                    Toast.makeText(BookLoginActivity.this,"登录成功",Toast.LENGTH_LONG).show();
                    Intent intent = new Intent(BookLoginActivity.this,BookMainActivity.class);
                    startActivity(intent);
                }
                else   //  不是 ok 登录失败
                {
                    Toast.makeText(BookLoginActivity.this,"登录失败",Toast.LENGTH_LONG).show();
                }


            } catch (JSONException e) {
                e.printStackTrace();
            }
        }

    }
};
```

】

6. 步骤 6：实现图书管理 App 图书清单主界面显示所有图书信息，访问图书清单信息 GET 接口，通过 ListView 展示图书清单。

（6-1）创建图书清单的单行图书布局 book_item，设置按照 4,3,3 比例横向排列的 TextView。

（同步看视频-6-1 图书清单列表页的单行布局 book_item 实现）

（6-2）实现图书清单列表界面 BookMainActivity 的布局 activity_book_main.xml 添加一个全屏的 ListView 组件，并且 id 取名为 list_view，并且在 java 文件中映射为 listView，书写 findViewById 代码。

（同步看视频-6-2 ListView 满屏布局和 java 代码映射）

（6-3）复制上面的访问 POST 接口的网络代码到这里 GET 接口，并且简化发包内容部分的代码删除。

接口描述	显示所有图书信息的 json 数组接口
url 地址	http://study.smartye.top/api/shop/book
访问方式	GET
发包格式	无
收包格式	[{book_name:“呐喊”,book_price:"32",writer:"鲁迅"}, {book_name:“西游记”,book_price:40",writer:"吴承恩"}, {book_name:“红楼梦”,book_price:"56",writer:"曹雪芹"}, {book_name:“红高粱”,book_price:"72",writer:"莫言"}, {book_name:“海燕”,book_price:"38",writer:"高尔基"}]

（同步看视频-6-3 复制访问登录 post 接口的网络访问代码和 handler 外壳）

（6-4）编写利用 JSONArray 数据填充的 BaseAdapter 的子类 MyAdapter 在 BookMainActivity 的内部称为静态内部类，同时在 Handler 里面创建这个类的实例将网络读取出并传递到 handler 里面的 JSONArray 数据作为构造函数的参数，最后将这个适配器和 ListView 绑定）。

（同步看视频-6-4 JSONArray 数据驱动适配器，最后绑定 ListView 显示）

【代码复习：BaseAdapter 子类 MyAdapter 的创建

```
private static class MyAdapter extends BaseAdapter
{
    private Context context;
    private JSONArray jsonArray;

    public MyAdapter(Context context,JSONArray jsonArray)
    {
        this.context = context;
        this.jsonArray =jsonArray;
    }

    @Override
    public int getCount() {
        return jsonArray.length();
    }

    @Override
    public Object getItem(int position) {
        return position;
```

```
    }

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

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = LayoutInflater.from(this.context);
        convertView = inflater.inflate(R.layout.book_item,null);
        TextView tvName = (TextView) convertView.findViewById(R.id.name);
        TextView tvPrice = (TextView) convertView.findViewById(R.id.price);
        TextView tvWriter = (TextView) convertView.findViewById(R.id.writer);
        try {
            tvName.setText(jsonArray.getJSONObject(position).getString("book_name"));
            tvPrice.setText(jsonArray.getJSONObject(position).getString("book_price")+"元");
            tvWriter.setText(jsonArray.getJSONObject(position).getString("writer"));
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return convertView;
    }
}
    }
};
```