szu-exp 安卓开发实验3我的校园

原创

Ant_ony 于 2021-12-01 19:28:10 发布 2678 收藏 2
分类专栏： szu-exp 文章标签： 安卓

本文链接：https://blog.csdn.net/weixin_52030069/article/details/121661955

> 发扬开源精神...
> 给个赞吧giegiejiejie们

实验目的与要求：

目的： 掌握安卓中活动的编写、自定义用户界面的开发、碎片开发、广播机制以及数据持久化技术等；并能通过对课堂知识进行扩展来完善该界面，并使界面尽量美观。

内容要求：

1. 请尽量模拟如下深大校园主页的功能，参考：

https://www1.szu.edu.cn/

2. 具体要求：

1) 该实现的界面在某些地方应体现出如下功能：

a. 界面能对平板与手机平台进行自适应（参考第4章碎片）；

b. 能对用户身份有强制下线的功能，比如网络中断，登录界面强行退出并显示提示错误的界面（错误dialog无法显示）

c. 界面某些地方体现数据持久化的技术，如文件数据的读取、存储的多种实现方式，并简单阐述几种实现方式具体的适用场景；

d. 界面要比较工整，没必要实现参考界面上的所有子项，能保证自己的界面实现能有扩展到参考界面的能力即可。

2) 功能并不局限于上面的要求，可以根据自己的理解设计一些新的功能，并在报告文档中进行详细的阐述，作为报告的亮点；

3）APP的布局尽快模仿参考界面，如果有较大的困难，可以只实现出右半边部分的界面，并尽量按上面要求进行完善；

4) 对于某一种功能，可以在不同的子项处采用多种实现方式，并比较这些实现方式的不同及优劣势。

3. 参考：尽量多的应用参考书《第一行代码 Android》第二版第2章活动、第3章UI开发第4章碎片、第5章广播机制与第6章数据持久化技术的各个知识点。

注意：

1. 实验报告中需要有功能的描述、实验结果的截屏图像及详细说明；重点要突出；

2. 也欢迎采用其它章节的知识点完成本次实验报告，如果实现的功能言之合理，会考虑酌情加分；

3. 尽量删除空白页。

方法、步骤：

实验需求分析：

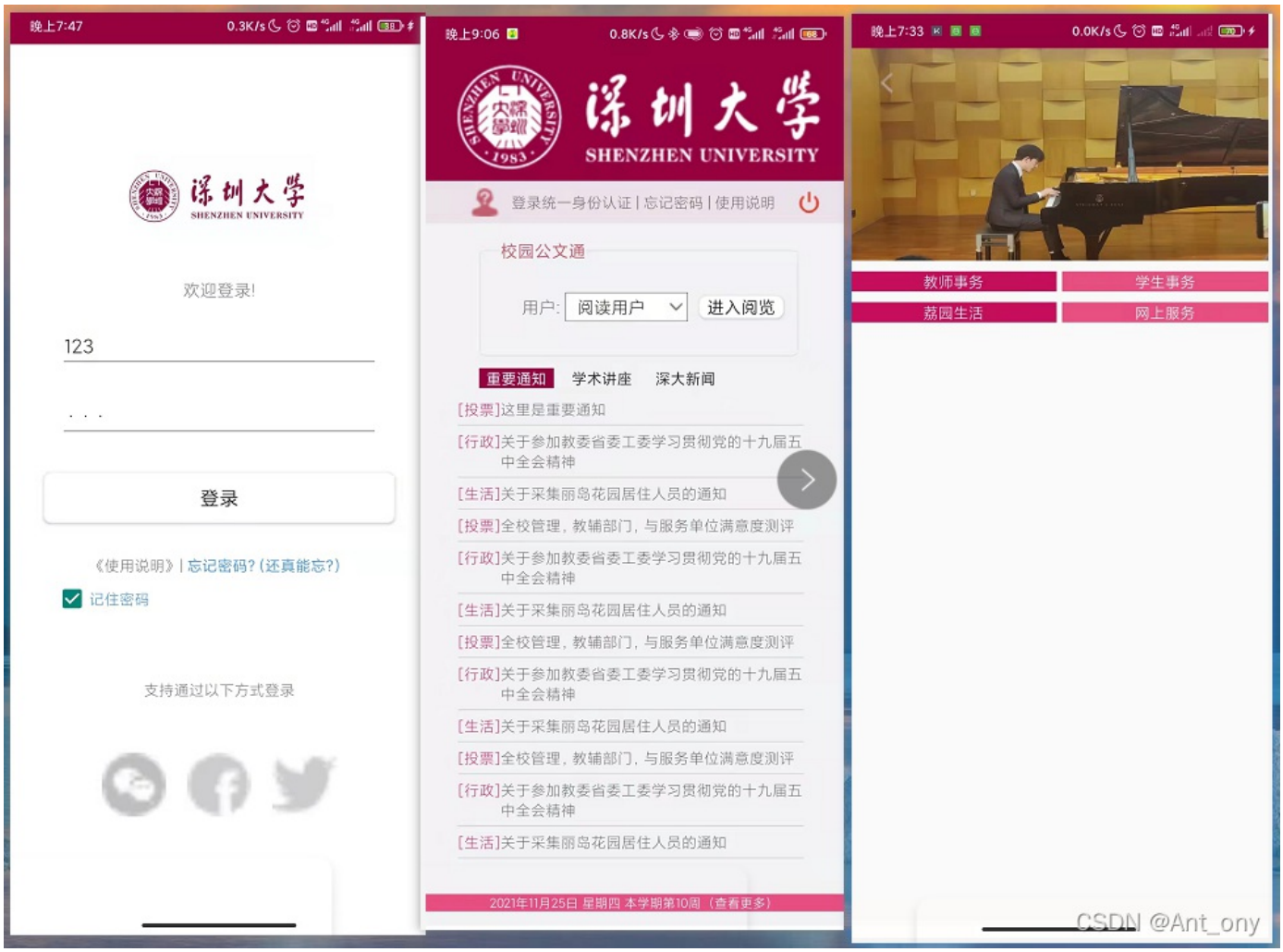
简要分析之后，我们发现，需**要做到手机和平板的自适应**，需要设置两个activity_main，其中一个我们使用large关键字进行限制，在手机布局中，我们的左布局和右布局不会一起展示，在平板布局中，左布局和右布局将会按1：3的比例展示。

强制下线的功能我们将通过广播来实现

数据持久化的技术将在实现记住密码功能中体现

样式模仿我们看效果就好了hh

先给我们所有页面来个大合照！



实验过程及内容：

两个activity的创建：

首先我们创建activity_main.xml以及activity_mian(large).xml

通过关键字large 我们就实现了碎片的自适应

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/layout"
    >
```


Activity_mian(large).xml 里面装有两个fragment

```
<fragment
    android:id="@+id/left_fragment"
    android:name="com.example.mycampus.LeftFragment"
    android:layout_width="0dp"
    android:layout_weight="2"
    android:layout_height="match_parent"
     />

<fragment
    android:id="@+id/tablet_r_fragment"
    android:name="com.example.mycampus.Tablet_right"
    android:layout_width="0dp"
    android:layout_weight="3"
    android:layout_height="match_parent"
     />
```


事实上，我们打算复用**left_fragment，平板中则新建tablet_r_fragment**接下来就先从left_fragment开始看吧

left_fragment的布局

布局的代码都是很普通的啦，就是linerLayout 各种嵌套，比较突出的点是使用了一下**Textclock** 以及使用了边框以及按钮的各种自定义布局

自定义的边框：（颜色和圆角）

代码：

设置背景为自定义的drawable

```
<LinearLayout
    android:layout_width="300dp"
    android:layout_height="100dp"
    android:layout_marginTop="25dp"
    android:layout_marginLeft="50dp"
    android:orientation="horizontal"
    android:background="@drawable/edge">
```


Drawable/edge:

```
<corners android:topLeftRadius="5dp"
    android:topRightRadius="5dp"
    android:bottomRightRadius="5dp"
    android:bottomLeftRadius="5dp"/>
<!-- 这里设置边框 -->
<stroke android:width="1dp" android:color="#e3e3e5" />
```


底部时钟的实现：

很简单，把textClock定义一下就好了

```
<TextClock
    android:id="@+id/show_more"
    android:clickable="true"
    android:layout_gravity="center"
    android:layout_marginLeft="60dp"
    android:textSize="12dp"
    android:textColor="@color/white"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:format12Hour="yyyy年MM月dd日 EEEE 本学期第10周（查看更多）" />
```


我们在left_fragment中还嵌套了一个ListView，关于怎么配置这一个ListView是非常重要的，在后文会有详细的解释。

首先我们在这里先放置一个**子framLayout吧,等下会展示三个List**

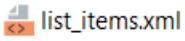
```
<FrameLayout
    android:id="@+id/three_tab_fragment"
    android:layout_width="325dp"
    android:layout_height="match_parent" >
</FrameLayout>
```


这就是left_fragment的基本布局了（还没有说里面嵌套的ListView）

子framLayout的配置（ListView）：

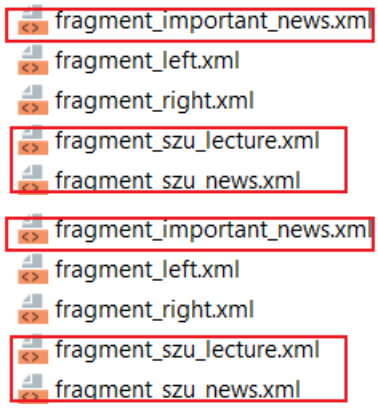
关于怎么向ListView添加元素已经在实验2中有过体现了，这里我也不在赘述了，大体就是：设置共有的item_list.xml—创建若干（所需要个数）的fragment以及对应的activity，fragment中放置listView—在各自的fragment的activity中配置数据。

共有list_item



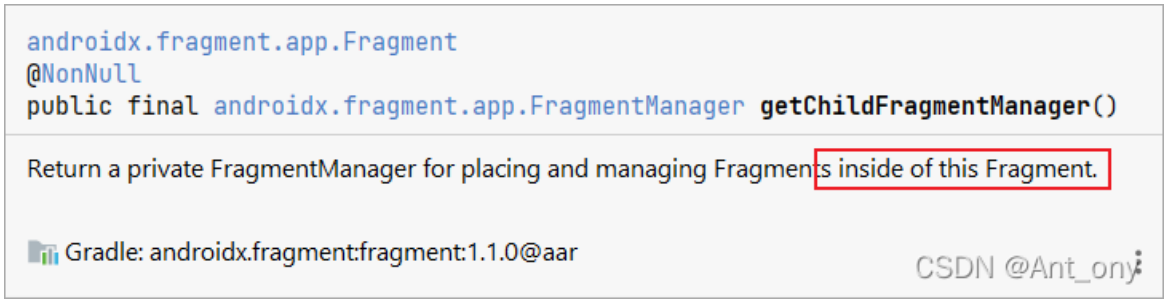
```
<LinearLayout
    android:layout_marginTop="5dp"
    android:layout_marginBottom="5dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/category"
        android:textColor="#cb3b7f"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```


三个所需fragment



重点来了！重点来了！重点来了！

重点就是因为这里是在FramLayout中使用FramLayout，（别忘了，我们的左子fragment也是FramLayout），所以们在切换布局的时候应该使用的是**getChildFragmentManager方法，**为什么呢，我们可以直接看到安卓对于这个方法的解释



详细说明一下，我们已经事先准备好了三个按钮，他们将可以点击以展示不同的列表

在LeftFragment中：

添加事务

```
getChildFragmentManager().beginTransaction()
        .add(R.id.three_tab_fragment,m_importantNews)
        .add(R.id.three_tab_fragment,m_szuLecture)
        .add(R.id.three_tab_fragment,m_szuNews)
        .commit() ;
return view ;
```


切换事务，这里代码放多了，顺便说了一下按钮背景的切换和字体颜色的切换

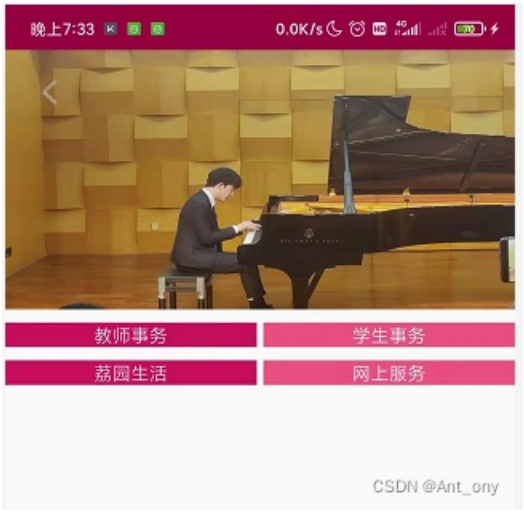
```
m_important.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //显示F3
        getChildFragmentManager().beginTransaction()
                .hide(m_szuLecture)
                .hide(m_szuNews)
                .show(m_importantNews)
                .commit();
        m_important.setSelected(true);
        m_news.setSelected(false);
        m_lecture.setSelected(false);
        m_important.setTextColor(getResources().getColor(android.R.color.white));
        m_news.setTextColor(getResources().getColor(android.R.color.black));
        m_lecture.setTextColor(getResources().getColor(android.R.color.black));
```


至此，我们的左布局已经完成啦！

右布局：

由于手机和平板的右边布局实在是复用不了，我们决定分别给他们建立两个不同的布局

手机：



在手机的右布局中，我们的顶部放置了VideoView,下面是四个ListView

平板：

在平板布局中，左布局和右布局将以2：3 的比例呈现

布局代码也不不需要过多的表述了，顶上一个轮播图，下面四个listView，然后是四个imageView

登录功能+强制下线—体现数据持久化以及广播

首先我们来看一下登录功能

基本的布局代码就没有什么好看的了

就是一个深大的标志，一个欢迎登陆的标语，一个输入账户的框，一个输入密码的框，一个记住密码的checkbox，然后是一些别的修饰

我们直接看一下效果吧

默认的登录账号和密码都是123

数据持久化的体现就在小小的记住密码上，

实现方法：

首先获取账号框，密码框，记住密码框的实例，

```
loginEtAccount = findViewById(R.id.loginEtAccount);
//loginEtAccount.setOnClickListener();
loginEtPassword = findViewById(R.id.loginEtPassword);
//loginEtPassword.setOnClickListener(this);
checkBox = (CheckBox) findViewById(R.id.checkbox) ;
```


我们使用**SHAREPREFERANCE—MODE_PRIVATE**来实现数据持久化

```
sharedPreferences = this.getSharedPreferences( name: "userinfo", Context.MODE_PRIVATE);
```


接下来调用自己写的**submit()方法，**实现检测密码以及存入数据的逻辑

```
private void submit() {
    // validate
    String loginEtAccountString = loginEtAccount.getText().toString().trim();
    if (TextUtils.isEmpty(loginEtAccountString)) {
       Toast.makeText( context: this,  text: "请输入账号", Toast.LENGTH_SHORT).show();
        return;
    }

    String loginEtPasswordString = loginEtPassword.getText().toString().trim();
    if (TextUtils.isEmpty(loginEtPasswordString)) {
        Toast.makeText( context: this,  text: "请输入密码", Toast.LENGTH_SHORT).show();
        return;
    }
    if ("123".equals(loginEtAccountString)&&"123".equals(loginEtPasswordString)){
        boolean isChecked = checkBox.isChecked() ;
        SharedPreferences.Editor editor = sharedPreferences.edit() ;
        if(isChecked){

            editor.putString("username",loginEtAccountString) ;
            editor.putString("password",loginEtPasswordString) ;
            editor.putBoolean("checkboxBoolean", true);

        } else
        {

            editor.putString("username", null);
            editor.putString("password", null);
            editor.putBoolean("checkboxBoolean", false);

        }editor.commit();
    }
```

检测输入是否正确的逻辑

如果选择了记住密码，就存入数据

然后回到初始化的界面，

如果发现当前checkbox 是选中状态的我们就要填入密码，反正不填入密码

```
if (sharedPreferences.getBoolean( key: "checkboxBoolean",  defValue: false))
{
    loginEtAccount.setText(sharedPreferences.getString( key: "username",  defValue: null));
    loginEtPassword.setText(sharedPreferences.getString( key: "password",  defValue: null));
    checkBox.setChecked(true);

}
```


至此，我们的记住密码功能已经得以实现，但我们还需要加入强制下线的功能才能看到记住密码的效果

强制下线：

强制下线的实现可能稍微有一点复杂，整体的逻辑是：在需要的活动中注册广播—设立广播接收器实现相应的逻辑—添加ActicityController类辅助实现逻辑，给强制下线按钮添加发送广播功能。

先看到ActivityController

```
@SuppressLint("Registered")
public class ActivityController extends Application {


    public static List<Activity> activities = new ArrayList<~>();
    public static void addActivity(Activity activity) { activities.add(activity); }
    public static void removeActivity(Activity activity) { activities.remove(activity); }
    public static void finishAll() { for (Activity activity : activities) {
        if (!activity.isFinishing()) { activity.finish(); }
    } }
}
```


提供了增加活动，移除活动，结束所有活动的功能

我们在左布局中定义了一个按钮，当点击按钮的时候就会发送强制下线的广播。

我们在LeftFragment中注册广播：

```
if (forceOfflineBroadcast == null) {
    forceOfflineBroadcast = new ForceOfflineBroadcast() ;
      forceOfflineBroadcast = new ForceOfflineBroadcast() {
          @Override
          public void onReceive(Context context, Intent intent) {
              Log.d("receive", "onReceive: yes");
          }
      };
}
localBroadcastManager = LocalBroadcastManager.getInstance(getActivity());
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction("com.example.mycampus.FORCE_OFFLINE");
localBroadcastManager.registerReceiver(forceOfflineBroadcast,intentFilter);
```


记得如果forceOfflineBroadcast 空，则我们要新建，因为每次结束活动都会把东西清掉

forceOfflineBroadcast实现接受广播后的代码逻辑：

```
@Override
public void onReceive(Context context, Intent intent) {
    Log.d( tag: "receive",  msg: "onReceive: you have received");
    Toast toast = Toast.makeText(context.getApplicationContext(),  text: "你完蛋啦，你下线啦！",Toast.LENGTH_SHORT) ;
    toast.show();
    ActivityController.finishAll();//销毁所有Activity
    Intent intent1 = new Intent(context, LoginActivity.class);
    intent1.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    context.startActivity(intent1);//重新启动LoginActivity
}
```


结束所有活动，重新进入login

点击按钮发送广播

```
forceoffline.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Log.d( tag: "Hello,I am DEBUG", msg: "Hello,I am DEBUG");
        Intent intent = new Intent( action: "com.example.mycampus.FORCE_OFFLINE") ;
        //sendBroadcast(intent);
        intent.putExtra( name: "resource",  value: "1");
        LocalBroadcastManager.getInstance(getActivity()).sendBroadcast(intent) ;
    }

});
```


最后我们还必须要记得在destory中解除注册

```
@Override
public void onDestroy() {
    super.onDestroy();
    localBroadcastManager.unregisterReceiver(forceOfflineBroadcast);
}
```

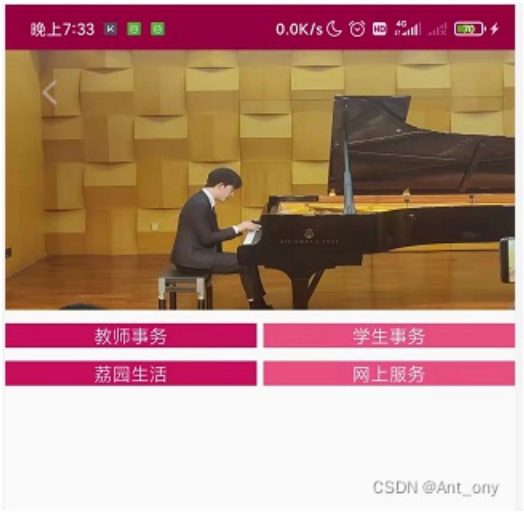

到这里，我们强制下线的功能就完成啦，来看一下效果

可以发现，点击按钮之后，我们接受到了强制下线的通知，并且回到了login页面

同时，如果我没有选择记住密码，那么强制下线之后密码就不会显示

其他功能——多媒体VideoView的使用

在右边布局中，我们在布局头部展示了一个视频，展示了深大的风采。



那么我们怎么实现呢

首先在fragment_right中放入VideoView

```
<VideoView

    android:layout_width="400dp"
    android:layout_height="wrap_content"
    android:id="@+id/videoView"/>
```


然后在RightFragment中

```
videoView = (VideoView)view.findViewById(R.id.videoView) ;
File videoFile = new File( pathname: "/storage/emulated/0/DCIM/Camera/VID_20211124_203107.mp4");
if (videoFile.exists()) {
    videoView.setVideoPath(videoFile.getAbsolutePath());
    videoView.setMediaController(mediaController);
    videoView.start();

    videoView.requestFocus();
}
```


Requestfocus保证了画面在中间。

获取实例，并且使用手机中已有的视频，（记得给手机开权限）并且使用了mediaCotroller从而无需手动实现进度条。

其他功能——轮播图的使用

我们在平板的右边布局的顶部增加了轮播图

首先创建好放置轮播图的viewPager

```
<androidx.viewpager.widget.ViewPager
    android:id="@+id/scroll_view"
    android:layout_width="match_parent"
    android:layout_height="120dp" />
```


然后新建MyPageeAdapter类继承PagerAdapter

```
// 1. 返回要显示的条目内容, 创建条目
@NonNull
@Override
public Object instantiateItem(@NonNull ViewGroup container, int position) {
    // container: 容器: ViewPager
    // position: 当前要显示条目的位置 0 -> 4
    //newPosition = position % 5
    int newPosition = position % imageViewList.size();
    ImageView img = imageViewList.get(newPosition);
    // a. 把View对象添加到container中
    container.addView(img);
    // b. 把View对象返回给框架, 适配器
    return img;
}

@Override
public void destroyItem(@NonNull ViewGroup container, int position, @NonNull Object object) {
    container.removeView((View)object);
}

@Override
public int getCount() { return Integer.MAX_VALUE;    //返回一个无限大的值，可以 无限循环!!!!! }

/**
 * 判断是否使用缓存, 如果返回的是true, 使用缓存. 不去调用instantiateItem方法创建一个新的对象
 */
@Override
public boolean isViewFromObject(@NonNull View view, @NonNull Object o) { return view == o ; }
```


回到Tablet_right初始化轮播图

```
viewPager = view.findViewById(R.id.scroll_view) ;
mimg = new int[]{
        R.drawable.right_header,
        R.drawable.header_2,
        R.drawable.header_3,
        R.drawable.header_4,
        R.drawable.header_5,
} ;
arrayList = new ArrayList<ImageView>() ;
ImageView imageView ;
for(int i = 0 ; i < mimg.length ;  i++){
    imageView = new ImageView(getContext()) ;
    imageView.setBackgroundResource(mimg[i]);
    arrayList.add(imageView) ;
}
viewPager.setAdapter(new MyPagerAdapter(arrayList));
int m = (Integer.MAX_VALUE / 2 ) % arrayList.size() ;
int currentPosition = Integer.MAX_VALUE / 2 - m ;
viewPager.setCurrentItem(currentPosition);
//轮询
new Thread(){
    public void run(){
        isRunning = true;
        while(isRunning){
            try{
                Thread.sleep( millis: 5000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            //下一条
            if(getActivity()!=null){
                getActivity().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        viewPager.setCurrentItem(viewPager.getCurrentItem()+1);
                    }
                });
            }
```

图片

装载

轮播逻辑

至此我们已经完成了轮播图的配置，我们来看看效果

还有几张就不展示了，附在后序链接上