**通用ViewPager框架**

问题:

现在使用的RecyclerView和ViewPager都是需要单独的使用适配器，而适配器的内容和很多都是重复的，每写一个adapter都会有许多的重复工作。除此之外许多的ViewPager适配都是不通用的，知识显示一张图片而已，如果变化了还得改适配的代码，和其他的框架并不兼容。为此需要主动权从adapter那里夺回来，写一个通用的ViewPager框架，要设计通用的框架就需要使用到万能的接口。

1.定义一个接口来创建视图和绑定数据

public interface ViewPagerHolder<T> {  
 /\*\*  
 \* 创建View  
 \* @param context  
 \* @return  
 \*/  
 View createView(Context context);  
 /\*\*  
 \* 绑定数据  
 \* @param context  
 \* @param position  
 \* @param data  
 \*/  
 void onBind(Context context,int position,T data);  
}

1. 创建一个接口专门生成各种ViewHolder(该类必须接收一个泛型而且必须是ViewHolder的子类，createViewHolder()返回一个ViewHolder的子类)

public interface ViewPagerHolderCreator<VH extends ViewPagerHolder> {  
 /\*\*  
 \* 创建ViewHolder  
 \* @return  
 \*/  
 public VH createViewHolder();  
}

1. 重写ViewPager的Adapter

这个类比较重要，因为以前adapter是提供布局和数据的显示的，但是，现在这个功能是不需要adapter来做了，他只需要提供一个view，view里面的内容就需要ViewHolder来做了。CommentAdapter的构造方法需要提供数据集合和ViewPagerHolderCreator生成器。

public class CommonViewPagerAdapter<T> extends PagerAdapter {  
 private List<T> mDatas;  
 private ViewPagerHolderCreator mCreator;//ViewHolder生成器  
 public CommonViewPagerAdapter(List<T> datas, ViewPagerHolderCreator creator) {  
 mDatas = datas;  
 mCreator = creator;  
 }  
 @Override  
 public int getCount() {  
 return mDatas == null ? 0:mDatas.size();  
 }  
 @Override  
 public boolean isViewFromObject(View view, Object object) {  
 return view == object;  
 }  
 @Override  
 public Object instantiateItem(ViewGroup container, int position) {  
 //重点就在这儿了，不再是把布局写死，而是用接口提供的布局  
 // 也不在这里绑定数据，数据绑定交给Api调用者。  
 View view = getView(position,null,container);  
 container.addView(view);  
 return view;  
 }  
 @Override  
 public void destroyItem(ViewGroup container, int position, Object object) {  
 container.removeView((View) object);  
 }  
 /\*\*  
 \* 获取viewPager 页面展示View  
 \* @param position  
 \* @param view  
 \* @param container  
 \* @return  
 \*/  
 private View getView(int position,View view ,ViewGroup container){  
 ViewPagerHolder holder =null;  
 if(view == null){  
 //创建Holder  
 holder = mCreator.createViewHolder();  
 view = holder.createView(container.getContext());  
 view.setTag(R.id.common\_view\_pager\_item\_tag,holder);  
 }else{  
 holder = (ViewPagerHolder) view.getTag(R.id.common\_view\_pager\_item\_tag);  
 }  
 if(holder!=null && mDatas!=null && mDatas.size()>0){  
 // 数据绑定  
 holder.onBind(container.getContext(),position,mDatas.get(position));  
 }  
 return view;  
 }  
}

1. 包装ViewPager

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <!-- ViewPager-->  
 <android.support.v4.view.ViewPager  
 android:id="@+id/common\_view\_pager"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"/>  
 <!-- 指示器 indicatorView-->  
 <com.viewpager.indicator.CircleIndicatorView  
 android:id="@+id/common\_view\_pager\_indicator\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:layout\_marginBottom="10dp"  
 app:indicatorSelectColor="@android:color/white"  
 app:indicatorColor="@android:color/darker\_gray"  
 app:fill\_mode="none"  
 app:indicatorSpace="5dp"  
 android:layout\_centerHorizontal="true"/>  
</RelativeLayout>

布局用到了ViewPager和CircleIndicatorView两个部分组成。

public class CommonViewPager<T> extends RelativeLayout {  
 private ViewPager mViewPager;  
 private CommonViewPagerAdapter mAdapter;  
 private CircleIndicatorView mCircleIndicatorView;  
 public CommonViewPager(@NonNull Context context) {  
 super(context);  
 init();  
 }  
 public CommonViewPager(@NonNull Context context, @Nullable AttributeSet attrs) {  
 super(context, attrs);  
 init();  
 }  
 public CommonViewPager(@NonNull Context context, @Nullable AttributeSet attrs, @AttrRes int defStyleAttr) {  
 super(context, attrs, defStyleAttr);  
 init();  
 }  
 @RequiresApi(api = Build.VERSION\_CODES.LOLLIPOP)  
 public CommonViewPager(@NonNull Context context, @Nullable AttributeSet attrs, @AttrRes int defStyleAttr, @StyleRes int defStyleRes) {  
 super(context, attrs, defStyleAttr, defStyleRes);  
 init();  
 }  
 private void init(){  
 View view = LayoutInflater.from(getContext()).inflate(R.layout.common\_view\_pager\_layout,this,true);  
 mViewPager = (ViewPager) view.findViewById(R.id.common\_view\_pager);  
 mCircleIndicatorView = (CircleIndicatorView) view.findViewById(R.id.common\_view\_pager\_indicator\_view);  
 }  
 /\*\*  
 \* 设置数据  
 \* @param data  
 \* @param creator  
 \*/  
 public void setPages(List<T> data, ViewPagerHolderCreator creator){  
 mAdapter = new CommonViewPagerAdapter(data,creator);  
 mViewPager.setAdapter(mAdapter);  
 mAdapter.notifyDataSetChanged();  
 mCircleIndicatorView.setUpWithViewPager(mViewPager);  
 }  
 public void setCurrentItem(int currentItem){  
 mViewPager.setCurrentItem(currentItem);  
 }  
 public int getCurrentItem(){  
 return mViewPager.getCurrentItem();  
 }  
 public void setOffscreenPageLimit(int limit){  
 mViewPager.setOffscreenPageLimit(limit);  
 }  
 /\*\*  
 \* 设置切换动画，see {@link ViewPager#setPageTransformer(boolean, ViewPager.PageTransformer)}  
 \* @param reverseDrawingOrder  
 \* @param transformer  
 \*/  
 public void setPageTransformer(boolean reverseDrawingOrder, ViewPager.PageTransformer transformer){  
 mViewPager.setPageTransformer(reverseDrawingOrder,transformer);  
 }  
 /\*\*  
 \* see {@link ViewPager#setPageTransformer(boolean, ViewPager.PageTransformer)}  
 \* @param reverseDrawingOrder  
 \* @param transformer  
 \* @param pageLayerType  
 \*/  
 public void setPageTransformer(boolean reverseDrawingOrder, ViewPager.PageTransformer transformer, int pageLayerType) {  
 mViewPager.setPageTransformer(reverseDrawingOrder,transformer,pageLayerType);  
 }  
 /\*\*  
 \* see {@link ViewPager#addOnPageChangeListener(ViewPager.OnPageChangeListener)}  
 \* @param listener  
 \*/  
 public void addOnPageChangeListener(ViewPager.OnPageChangeListener listener){  
 mViewPager.addOnPageChangeListener(listener);  
 }  
 /\*\*  
 \* 设置是否显示Indicator  
 \* @param visible  
 \*/  
 public void setIndicatorVisible(boolean visible){  
 if(visible){  
 mCircleIndicatorView.setVisibility(View.VISIBLE);  
 }else{  
 mCircleIndicatorView.setVisibility(View.GONE);  
 }  
 }  
 public ViewPager getViewPager() {  
 return mViewPager;  
 }  
}

CommonViewPager 是对ViewPager的包装，提供了一些ViewPager的常用方法。 其中有一个非常重要的方法public void setPages(List<T> data, ViewPagerHolderCreator creator),提供数据和ViewHolder。其他的基本上都是ViewPager的方法。也可以通过getViewPager 获取到ViewPager 再调用ViewPager的方法。

1. ViewPager Item的布局如下(这个布局可以自定义的，比以前的灵活多了):

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@+id/common\_view\_pager\_item\_tag"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <ImageView  
 android:id="@+id/viewPager\_item\_image"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:scaleType="centerCrop"/>  
 <TextView  
 android:id="@+id/item\_desc"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textSize="15sp"  
 android:gravity="center"  
 android:layout\_centerInParent="true"  
 android:textColor="@android:color/white"/>  
</RelativeLayout>

1. Activity的布局下正是的使用自己定义的Activity

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <com.viewpager.indicator.CommonViewPager  
 android:id="@+id/activity\_common\_view\_pager"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"/>  
</RelativeLayout>

1. Activity内部的代码如下:

public class MainActivity extends Activity {  
private CommonViewPager mCommonViewPager;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 requestWindowFeature(Window.FEATURE\_NO\_TITLE);  
 setContentView(R.layout.activity\_main);  
 initView();  
 }  
 private void initView() {  
 mCommonViewPager = (CommonViewPager) findViewById(R.id.activity\_common\_view\_pager);  
 //mCommonViewPager.setIndicatorVisible(false);  
 // 设置数据  
 mCommonViewPager.setPages(mockData(), new ViewPagerHolderCreator<ViewImageHolder>() {  
 @Override  
 public ViewImageHolder createViewHolder() {  
 // 返回ViewPagerHolder  
 return new ViewImageHolder();  
 }  
 });  
 }  
 /\*\*  
 \* 生成数据  
 \* @return  
 \*/  
 private List<DataEntry> mockData() {  
 List<DataEntry> list = new ArrayList<>();  
 for (int i=0;i<5;i++) {  
 DataEntry data = new DataEntry();  
 data.imageResId=R.mipmap.toast\_bg1;  
 data.desc="描述"+i;  
 list.add(data);  
 }  
 return list.size()>0?list:null;  
 }  
 /\*\*  
 \* 提供ViewPager展示的ViewHolder  
 \* <P>用于提供布局和绑定数据</P>  
 \*/  
 public static class ViewImageHolder implements ViewPagerHolder<DataEntry> {  
 private ImageView mImageView;  
 private TextView mTextView;  
 @Override  
 public View createView(Context context) {  
 // 返回ViewPager 页面展示的布局  
 View view = LayoutInflater.from(context).inflate(R.layout.view\_pager\_item,null);  
 mImageView = (ImageView) view.findViewById(R.id.viewPager\_item\_image);  
 mTextView = (TextView) view.findViewById(R.id.item\_desc);  
 return view;  
 }  
 @Override  
 public void onBind(Context context, int position, DataEntry data) {  
 // 数据绑定  
 // 自己绑定数据，灵活度很大  
 mImageView.setImageResource(data.imageResId);  
 mTextView.setText(data.desc);  
 }  
 }  
}