ViewPager官方提供V4包中，它是google SDk中自带的一个附加包的一个类,一个可以左右滑动的容器
布局声明：

```
<android.support.v4.view.ViewPager
    android:id="@+id/vp_content"
    android:layout_weight="1"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    ></android.support.v4.view.ViewPager>
```

代码实现：
获取控件对象

```
ViewPager mViewPager = findView(R.id.vp_content);
```

获取数据适配器对象

```
MainVpAdapter mMainVpAdapter = new MainVpAdapter(getSupportFragmentManager(),mBasePagers);
```

设置数据

```
mViewPager.setAdapter(mMainVpAdapter);
```

mViewPager的选择事件

```
mViewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
    @Override   //
    public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {
    }
    @Override  //
    public void onPageSelected(int position) {
        mTabHost.setCurrentTab(position);
    }
    @Override  //
    public void onPageScrollStateChanged(int state) {
    }
});
```

MainVpAdapter适配器对象：

自定义类继承FragmentPagerAdapter或者继承PagerAdapter的区别：

FragmentPagerAdapter
FragmentPagerAdapter 继承自 PagerAdapter。相比通用的 PagerAdapter，
该类更专注于每一页均为 Fragment 的情况。如文档所述，**该类内的每一个生成的 Fragment 都将保存在内存之中**，
因此适用于那些相对静态的页，数量也比较少的那种；如果需要处理有很多页，并且数据动态性较大、占用内存较多的情况，
应该使用**FragmentStatePagerAdapter**。

前者加载数据返回加载的是Fragment对象，后者加载并返回的是View对象

前者适用于与FragmentTaost组合，后者适用于设置加载轮播图

方法一、自定义类继承 extends FragmentPagerAdapter 以下是重写的方法：

```
private List<BaseFragment> mBasePagers = new ArrayList<>();   //数据必须是Fragment的对象
public MainVpAdapter(FragmentManager fm,List<BaseFragment> mBasePagers) {
    super(fm);
    this.mBasePagers = mBasePagers;
}
@Override   //需要返回Fragment对象
public Fragment getItem(int position) {
```

```
    BaseFragment fragment = mBasePagers.get(position);
    return fragment;
/*如果没有数据集合传入可以用以下方法：
     Fragment fragment = null;
     switch (position){
         case 0:
             fragment = new Fragments1();
         break;
         case 1:
             fragment = new Fragments2();
             break;
     }
     return fragment;
*/

}
@Override   //返回数据的条目
public int getCount() {
    return mBasePagers.size();
}
```

方法二、自定义类继承 extends PagerAdapter 以下是具体实现：

```
private List<ImageView> mImageViewList = new ArrayList<>();
//获取数据
public HomeAdapter(List<ImageView> imageViewList) {
    mImageViewList = imageViewList;
}
@Override   //设置要显示的数据一共有多少？
public int getCount() {
    return mImageViewList.size();
}
//ViewPager里面用了一个mItems(ArrayList)来存储每个page的信息(ItemInfo)，当界面要展示或者发生变化时，需要依据page的
//当前信息来调整，但此时只能通过view来查找，所以只能遍历mItems通过比较view和object来找到对应的ItemInfo
@Override
public boolean isViewFromObject(View view, Object object) {
    return view==object;
}
@Override   //这个方法返回的值必须一定要是一个View，面且必须要向ViewGroup里添加
public Object instantiateItem(ViewGroup container, int position) {
    ImageView imageView = mImageViewList.get(position);
    container.addView(imageView);
    return imageView;
}
@Override   //当View销毁的时候 要做的事
public void destroyItem(ViewGroup container, int position, Object object) {
    container.removeView((View) object);
}
```

自动轮播图的实现过程：

1、一，设置要显示的数据数量 以确定可以进行轮播的核心条件

```
/**
 * 一，设置要显示的数据数量 以确定可以进行轮播的核心条件
 * @return    返回数量为 要显示的数量
 */
@Override
public int getCount() {
    return Integer.MAX_VALUE/2;
}
```

2、通过*position*的值 来确定当前显示的是哪一个，这里的值决定了可以向右滑动

```
/**
 * 二、通过position的值 来确定当前显示的是哪一个，这里的值决定了可以向右滑动
 */
@Override
public Object instantiateItem(ViewGroup container, int position) {
    /*当position的值大于mViews集合时  通过求余那到正确的当前位置*/

    position=position%mViews.size();
    ImageView imageView = mViews.get(position);
    container.addView(imageView);
```

3、通过设置当前*position*的位置来确保可以向左滑动 并且通过求余来确保显示位置的正确性

```
/**
 * 三、通过设置当前position的位置来确保可以向左滑动  并且通过求余来确保显示位置的正确性
 */
int index=Integer.MAX_VALUE/4;
index=index-index%mViews.size();
mVp.setCurrentItem(index);
```

4、通过*Handler.postDelayed*方法创建自动无限轮播

```
/**
 * 四、通过Handler.postDelayed方法创建自动无限轮播
 */
private void startvp() {
    mHandler.postDelayed(new Runnable() {
        @Override
        public void run() {
            int currentItem = mVp.getCurrentItem();
            currentItem++;
            mVp.setCurrentItem(currentItem);
            mHandler.postDelayed(this,4000);
        }
    },4000);
}
```

5、设置停止轮播的方法

```
/**
 * 五、设置停止轮播的方法
 */
private void stopBanner() {
    mHandler.removeCallbacksAndMessages(null);
}
```

6、处理手动滑动时的事件 停止自动

```
/**
 * 六、处理手动滑动时的事件  停止自动
 */
mVp.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        switch (event.getAction()){
            case MotionEvent.ACTION_DOWN:
                stopBanner();
                break;
            case MotionEvent.ACTION_MOVE:
                break;
            case MotionEvent.ACTION_UP:
                startvp();
                break;
        }
        return false;
    }
});
```

7、实现小红点的切换及文字的切换 通过当前的位置来确定要显示的数据

```
/**
 * @param position  七、实现小红点的切换及文字的切换  通过当前的位置来确定要显示的数据
 */
@Override
public void onPageSelected(int position) {
    position=position%mViews.size();
    mTvBannerTitle.setText(TITLES[position]);
    //设置所有小点为没有选择的白点
    for (int i = 0; i < mLlDot.getChildCount(); i++) {
        mLlDot.getChildAt(i).setSelected(false);
    }
    //设置当前选择的为红色
    mLlDot.getChildAt(position).setSelected(true);
}
```

8、在instantiateItem(ViewGroup **container**, int **position**)设置点击某一个图片后的跳转事件

```java
/**
 * 八、设置点击某一个图片后的跳转事件
 */
View viewClick=mViews.get(position);
final int finalPosition = position;
viewClick.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        switch (finalPosition){
            case 0:
                Intent intentA = new Intent(MainActivity.this,A.class);
                startActivity(intentA);
                break;
            case 1:
                Intent intentB = new Intent(MainActivity.this,B.class);
                startActivity(intentB);
                break;
            case 2:
                Intent intentC = new Intent(MainActivity.this,C.class);
                startActivity(intentC);
                break;
```

布局核心控件

```
<RelativeLayout
android:layout_width="match_parent"
android:layout_height="150dp">
<android.support.v4.view.ViewPager
android:id="@+id/vp"
android:layout_width="match_parent"
android:layout_height="match_parent">
</android.support.v4.view.ViewPager>
<RelativeLayout
android:layout_width="match_parent"
android:layout_height="50dp"
android:background="#2287758e"
android:layout_alignParentBottom="true">
<TextView
android:id="@+id/tv_banner_title"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="今天天气好冷"
android:layout_centerHorizontal="true"
android:textSize="14sp"
android:textColor="#000"
android:layout_marginTop="5dp"/>
<LinearLayout
android:id="@+id/ll_dot"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="horizontal"
android:layout_below="@id/tv_banner_title"
android:layout_marginTop="10dp"
android:layout_alignParentRight="true"
```

```
android:layout_marginRight="15dp"></LinearLayout>
</RelativeLayout>
</RelativeLayout>
```

DrawerLayout官方提供V4包中　SlidingMenu第三方控件

navigationView 侧滑单菜单控件（包含在DrawerLayout里面）

SmartTabLayout