목차

1. ViewPager 란

안드로이드에서도 하나의 Activity 에서 여러 개의 화면을 배치할 수 있는 UI 들이 생겨나고 있다.

수평으로 스크롤 되는 View 를 만들 경우가 있습니다. 많은 안드로이드 어플리케이션들이 이 기능을 활용하고 있는데 **ViewPager 는 수평 스크롤 기능을 구현하기 위한 표준 방법입니다**.

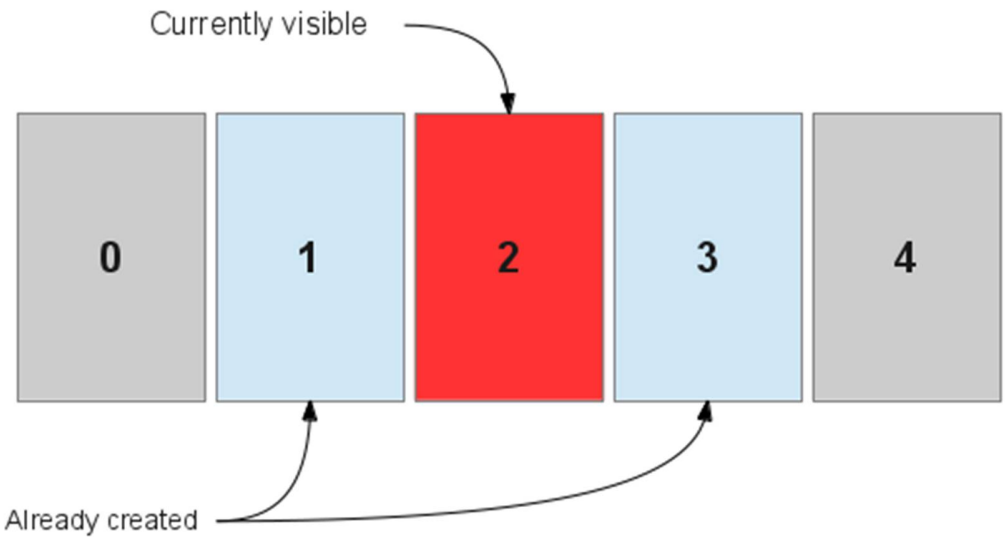
- ViewPager 는 View 또는 Fragment 를 페이지 단위로 관리 할 수 있는 커스텀 뷰입니다.
- ViewPager 은 화면 전환 없이 좌우 스크롤을 통해 효율적으로 페이지 전환을 할 수 있는 UI 이다.
- ViewPager 은 기본적으로 좌/우 화면을 미리 로딩하기 때문에 빠른 좌우 화면 전환을 보여준다.

ViewPager 는 수평으로 View 를 좌/우로 스크롤 할 때 사용하는 클래스입니다.

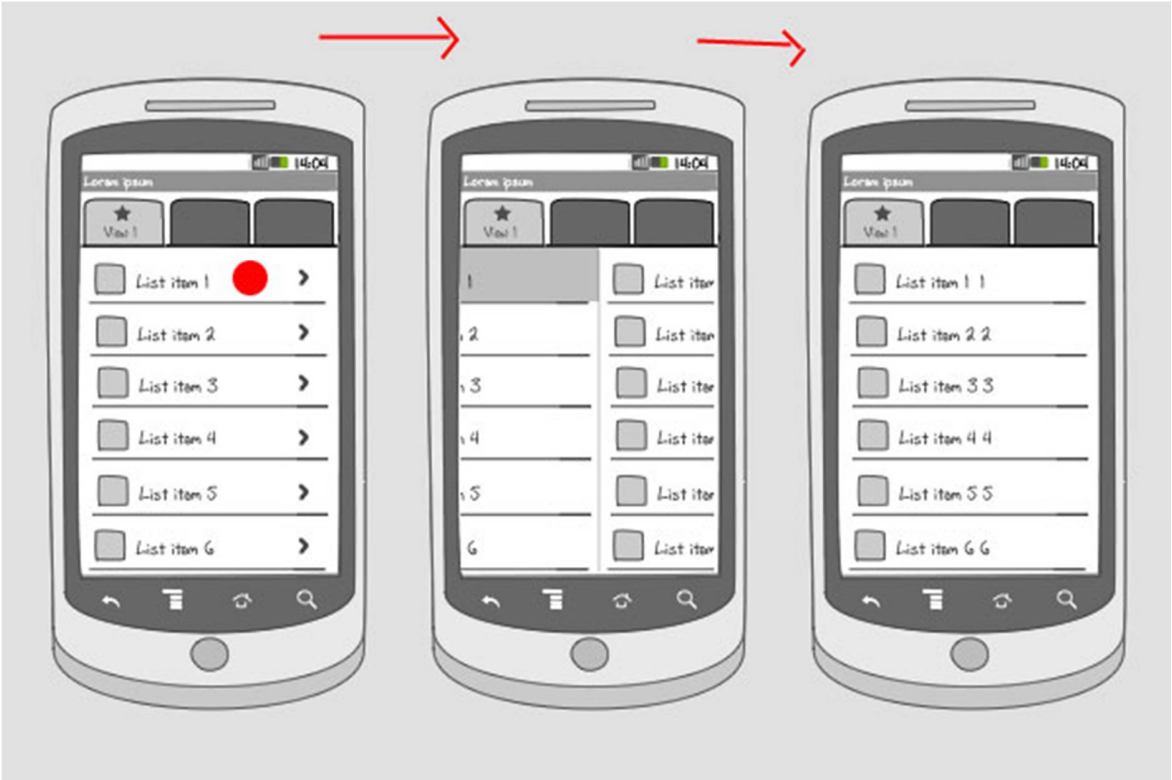
- View 를 좌/우로 스크롤 할 때 사용할 수 있는 클래스
- "android-support-v4.jar" 라이브러리에서 제공
- PagerAdapter 를 사용하여 View 를 관리
- ViewPager 는 주로 Fragment 와 결합되어 사용된다.
- ViewPager 는 ViewGroup 의 일종으로 AdapterView 와 유사한 형식으로 동작합니다.

ViewPager 는 터치 슬라이드 이벤트에 바로 이전이나 다음 페이지가 함께 보여야 하기 때문에 현재 페이지의 이전 페이지와 다음 페이지를 미리 생성하는 특성을 가지고 있다.

setOffscreenPageLimit(int) 메서드를 이용하면 양쪽에 유지되는 페이지수를 설정 할 수가 있다. 예를 들어 총 5 개의 페이지가 있는 경우 4 로 설정 하면 5 개의 페이지를 초반에 미리 로딩한다. (default 는 1)



ViewPager


Loren ipsum
Vie1
List item 1
List item 2
List item 3
List item 4
List item 5
List item 6
Loren ipsum
Vie1
Loren ipsum
Vie1
List item 1 1
List item 2 2
List item 3 3
List item 4 4
List item 5 5
List item 6 6

2. ViewPager 사용방법

```
// layout xml
<android.support.v4.view.ViewPager></android.support.v4.view.ViewPager>

// java
import android.support.v4.view.ViewPager;
import android.support.v4.view.PagerAdapter;
```

3. PagerAdapter

ViewPagers 에 표시되는 View 는 PagerAdapter 를 통해 공급 받는다. PagerAdapter 를 통해 화면에 표시 될 View 를 관리 할 수 있습니다. 즉, PagerAdapter 는 ViewPager 에 표시되는 View 를 관리한다.

setAdapter 를 통해 ViewPager 와 PagerAdapter 를 연결 합니다.

```
mPager = (ViewPager)findViewById(R.id.pager);
mPager.setAdapter(new PagerAdapterClass(getApplicationContext()));
```

PagerAdapter 종류

- PagerAdapter
- FragmentPagerAdapter
- FragmentStatePagerAdapter

각각의 PagerAdapter 는 조금 다른 특징을 가지고 있습니다.

FragmentPagerAdapter 는 ViewPager 에서 Fragment 를 활용할 수 있습니다. FragmentPagerAdapter 는 화면을 슬라이딩으로 전환할 때 한 번 생성된, 화면에 보인 Fragment 를 계속 메모리상 가지고 있습니다. 이전 Fragment 로 슬라이딩을 해서 돌아간다고 하면 이전에 생성된 Fragment 로 돌아가는 겁니다.

FragmentStatePagerAdapter 는 화면이 전환될 때 이전(화면에서 보이지 않는) Fragment 는 메모리상 제거(destroy)가 됩니다. Adapter 의 Fragment 가 많거나 개수를 알 수 없을 때 메모리관련 이슈를 위해 사용하는 것이 좋다고 하네요

FragmentPagetAdapter 나 FragmentStatePagerAdapter 생성자의 파라미터로 FragmentManager 를 넘겨주어야 한다.

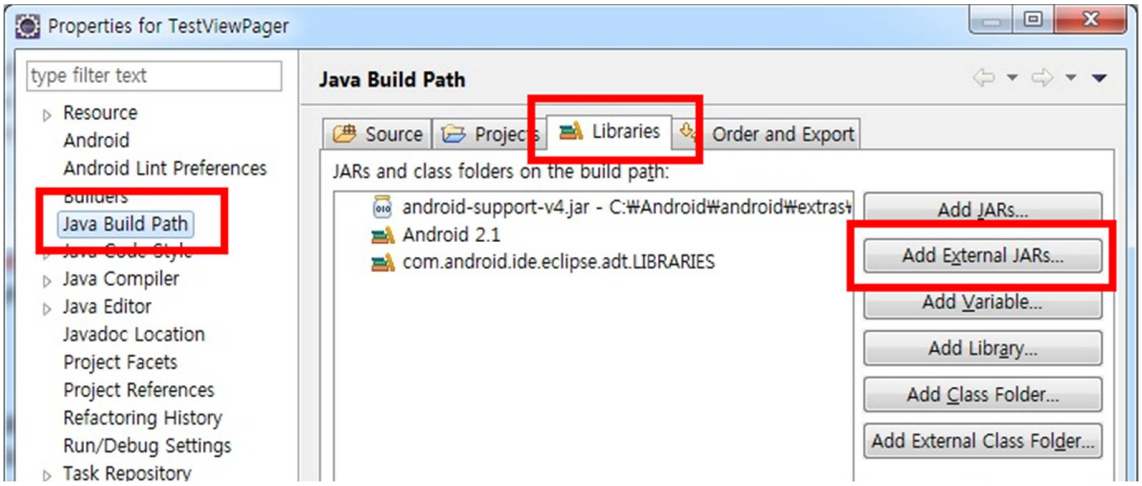
PagerAdapter 메서드

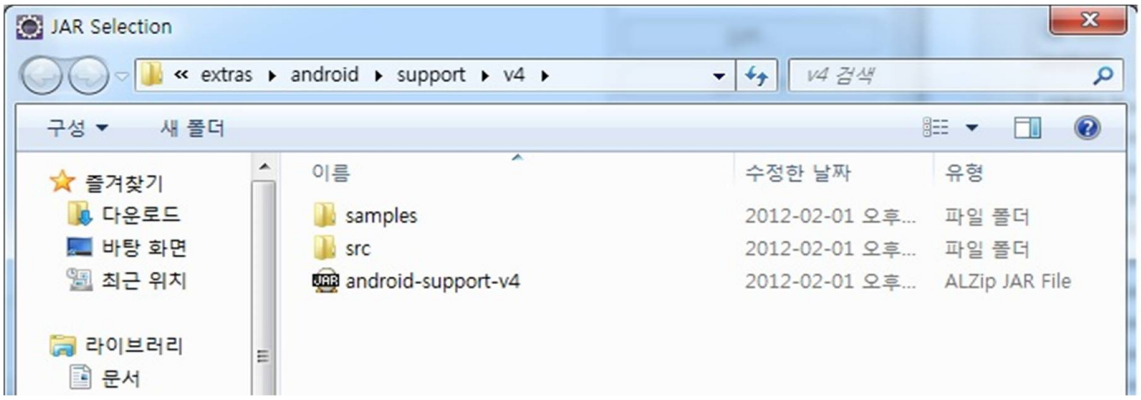
getCount()	현재 PagerAdapter 에서 관리할 갯수를 반환 한다.
instantiateItem()	ViewPager 에서 사용할 뷰객체 생성 및 등록 한다.
destroyItem()	View 객체를 삭제 한다.
isViewFromObject()	instantiateItem 메소드에서 생성한 객체를 이용할 것인지 여부를 반환 한다.
restoreState()	saveState() 상태에서 저장했던 Adapter 와 page 를 복구 한다.
saveState()	현재 UI 상태를 저장하기 위해 Adapter 와 Page 관련 인스턴스 상태를 저장 합니다.
startUpdate()	페이지 변경이 시작될때 호출 됩니다.
finishUpdate()	페이지 변경이 완료되었을때 호출 됩니다.

setCurrentInflateItem()	강제로 페이지 이동할 때

4. android-support-v4 라이브러리 추가

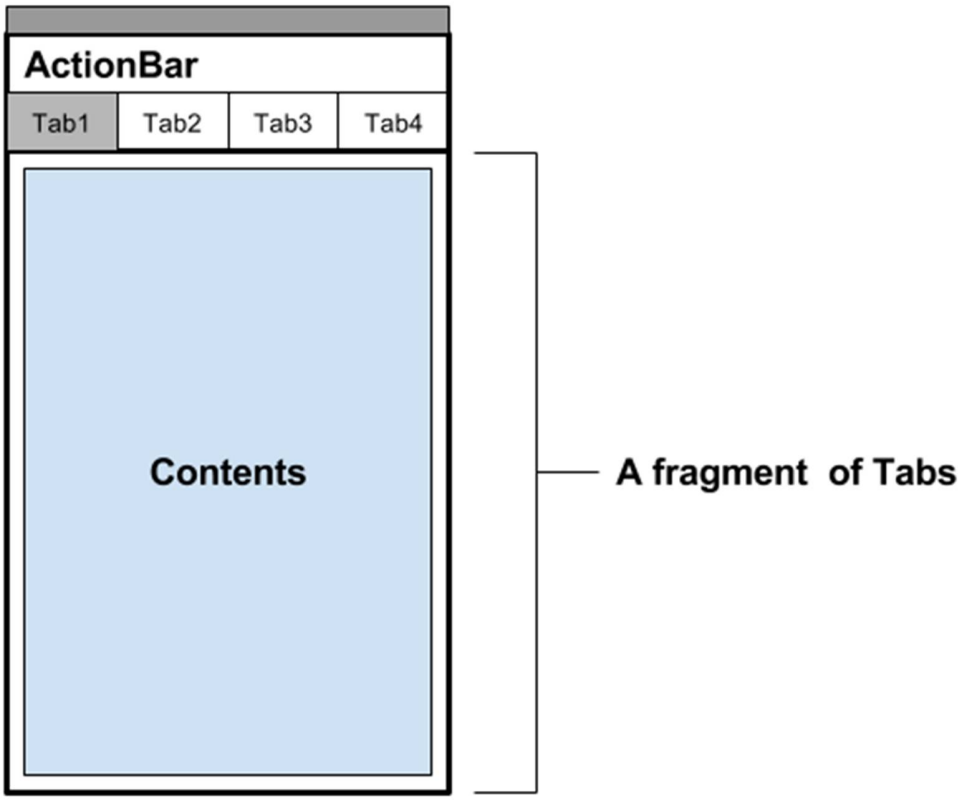
Viewpager 는 안드로이드 SDK 에서 기본으로 제공해주는 클래스가 아니기 때문에 "android-support-v4" Libary 를 추가 해서 사용 해야 합니다.

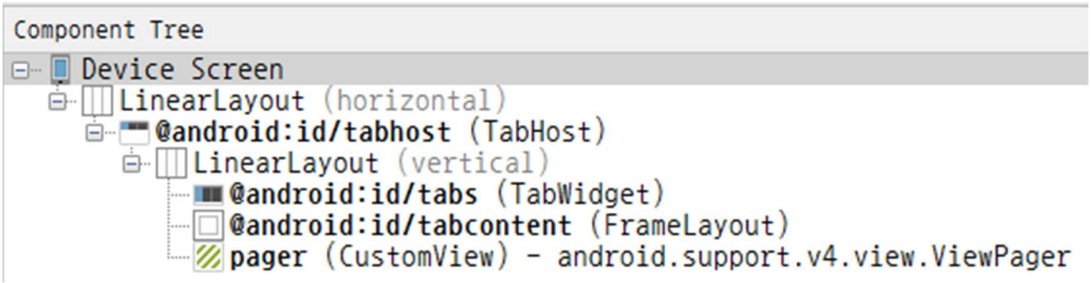




5. 예제 1

슬라이드 화면 구성





ViewPager XML 적용

라이브러리를 재대로 추가 한 후에 XML 에 ViewPager 를 사용한 코드 입니다. Viewpager 를 사용하기 위해서는<android.support.v4.view.ViewPager> 클래스를 풀네임을 모두 적어 주어야 하는데요. 그 이유는 아까 말씀 드렸듯이 안드로이드가 기본적으로 지원하는 클래스가 아니기 때문 입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="50dip"
    android:orientation="horizontal">

    <TabHost xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@android:id/tabhost"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <TabWidget
                android:id="@android:id/tabs"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_weight="0"
                android:orientation="horizontal" />

            <FrameLayout
                android:id="@android:id/tabcontent"
                android:layout_width="0dp"
                android:layout_height="0dp"
                android:layout_weight="0" />

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

        </LinearLayout>
    </TabHost>

</LinearLayout>
```

TabHost 추가

PagerAdapter 클래스 생성

```
/**
 * PagerAdapter
 */
public class PagerAdapterClass extends PagerAdapter
        implements TabHost.OnTabChangeListener, ViewPager.OnPageChangeListener{

    private LayoutInflater mInflater;

    public PagerAdapterClass(Context c){
        super();
        mInflater = LayoutInflater.from(c);
    }

    @Override
    public int getCount() {
        return 3;
    }

    @Override
    public Object instantiateItem(View pager, int position) {
        View v = null;
        if(position==0){
            v = mInflater.inflate(R.layout.inflate_one, null);
            v.findViewById(R.id.iv_one);
            v.findViewById(R.id.btn_click).setOnClickListener(mPagerListener);
        }
        else if(position==1){
            v = mInflater.inflate(R.layout.inflate_two, null);
        }else{
            v = mInflater.inflate(R.layout.inflate_three, null);
        }

        ((ViewPager)pager).addView(v, 0);

        return v;
    }

    @Override
    public void destroyItem(View pager, int position, Object view) {
        ((ViewPager)pager).removeView((View)view);
    }

    @Override
    public boolean isViewFromObject(View pager, Object obj) {
        return pager == obj;
    }

    @Override public void restoreState(Parcelable arg0, ClassLoader arg1) {}
    @Override public Parcelable saveState() { return null; }
    @Override public void startUpdate(View arg0) {}
    @Override public void finishUpdate(View arg0) {}



    @Override
    public void onPageScrolled(int position, float positionOffset
            , int positionOffsetPixels) {

    }
```

```
    @Override
    public void onPageSelected(int position) {

    }

    @Override
    public void onPageScrollStateChanged(int state) {

    }

    @Override
    public void onTabChanged(String tabId) {

    }

}
```

Adapter 를 ViewPager 에 연결

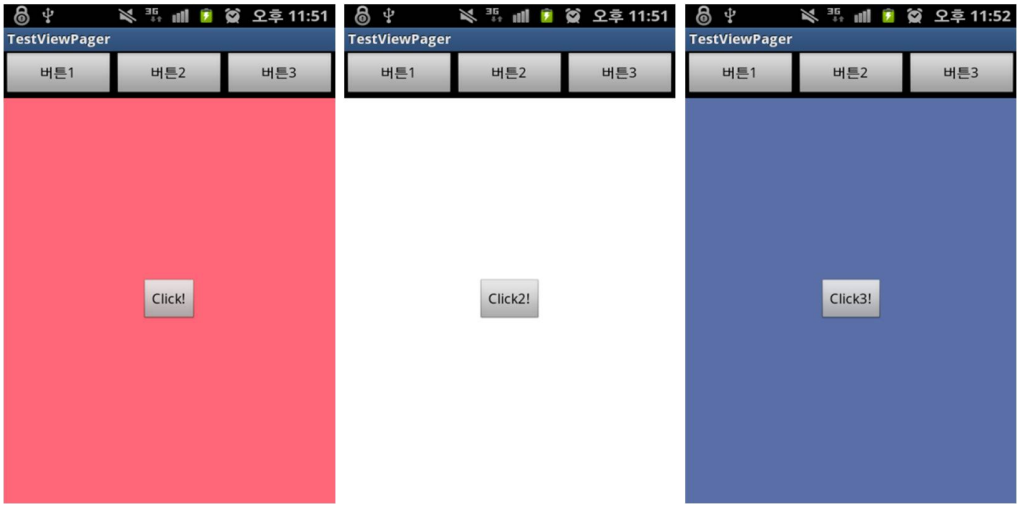
```
public class TestViewPagerActivity extends Activity implements OnClickListener{
    private ViewPager mPager;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mPager = (ViewPager)findViewById(R.id.pager);
        mPager.setAdapter(new PagerAdapterClass(getApplicationContext()));
    }

}
```

코드 설명

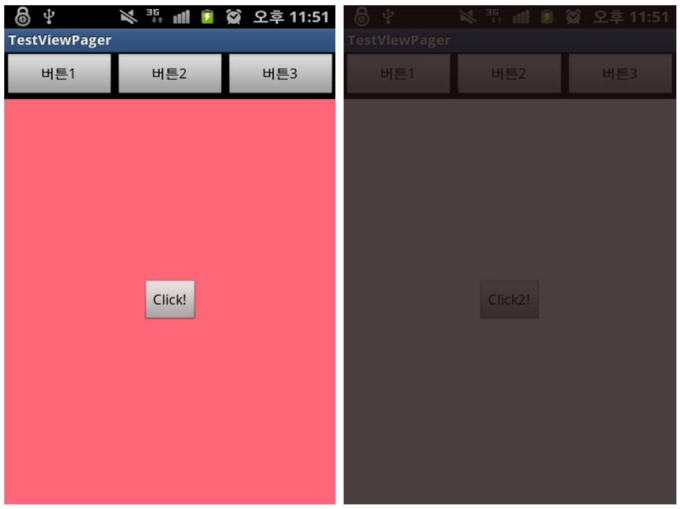
PagerAdapter 상속 구현 클래스 부분 입니다. PagerAdapter 에서 instantiateItem() 라는 오버라이드 메서드가 있는데 ViewPager 의 getCount()에서 얻어온 Count 의 Position 별로 Pager 에 등록할 item 을 처리할 수 있는 메서드 입니다. 즉, VieaPager 에서 사용할 뷰 객체를 생성하는 작업 입니다.

단 주의 해야 할 점이 있는데, 처음 실행시 PagerAdapter 는 현재의 position 의 Item 의 양쪽 옆의 View 들을 모두 instantiateItem() 해준다는 점인데요. 처음 시작시에는 position 션이 첫번째인 item 을 보여주기 때문에 아래와 같은 형태로 View 를 그려 줍니다.

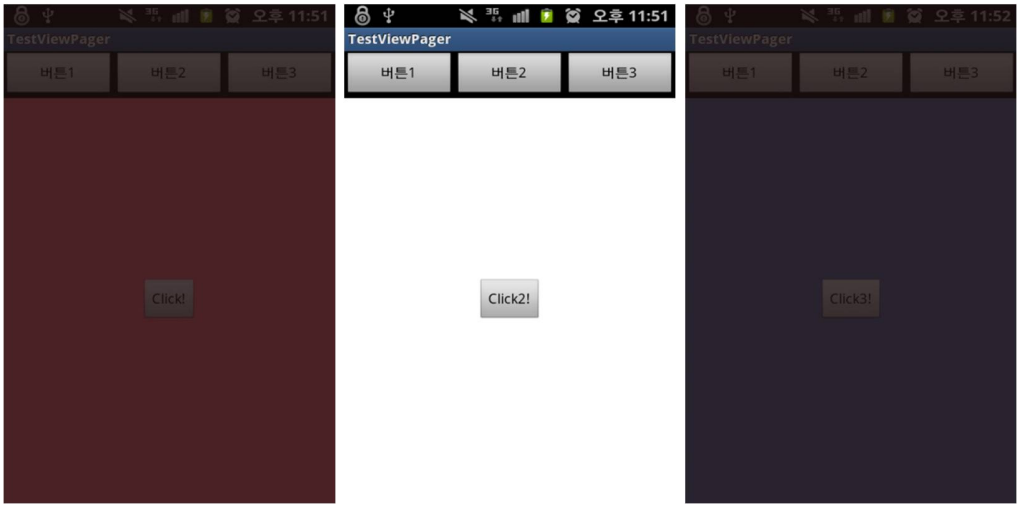
처음 저희가 사용할 View 는 총 3 개의 View 입니다.(Position 첫번째, 두번째 , 세번째)



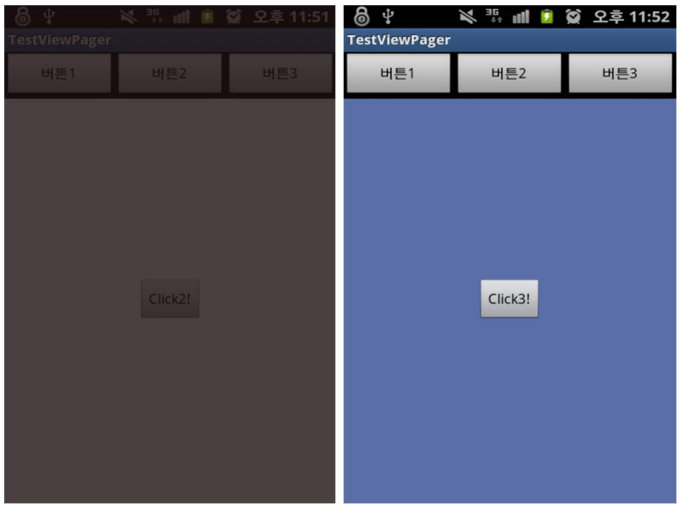
1. 첫번째 포지션으로 시작했을 경우 입니다. (첫번째 두번째 View 가 모두 생성 됩니다. 이말은 즉 instantiateItem() 을 두번 탓다는 얘기 입니다.)



2. 두번째 포지션으로 이동했을 경우 입니다. (세번째 포지션이 생성 됩니다. 그리고 첫번째 포지션도 유지하고 있습니다.) 포지션이동은 좌/우 스크롤을 이용하거나 setCurrentInflateItem(int position) 메서드로 원하는 포지션으로 이동할 수 있습니다.



3. 세번째 포지션으로 이동했을 경우 입니다. (첫번째 포지션을 삭제 합니다.)



이러한 형태로 PagerAdapter 의 View 관리가 이루어 집니다. 항상 양쪽의 View 를 생성하거나 유지 시키고 나머지 포지션은 삭제 하는 형태 입니다.

http://blog.daum.net/mailss/16
http://blog.naver.com/PostView.nhn?blogId=huewu&logNo=110116958816
http://arabiannight.tistory.com/entry/안드로이드 Andorid-Viewpager-사용-하기

6. 무한 스크롤 ViewPager 예제

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v4.view.ViewPager.OnPageChangeListener;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;

//support package 에 있는 FragmentActivity 를 상속 받는다.
public class FragmentPagerAdapterSampeActivity extends FragmentActivity {

    private final static int COUNT=4; // 표시할 페이지 수
    private ViewPager mPager; //뷰 페이저

    private static OnClickListener mButtonClick = new OnClickListener() {  //클릭 이벤트 객체
        public void onClick(View v) {
            String text = ((Button)v).getText().toString();  //버튼에만 이벤트를 등록. 버튼 글자 가져옴.
            Toast.makeText(v.getContext(), text, Toast.LENGTH_SHORT).show(); //토스트로 출력
        }
    };

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.viewpager);
        mPager = (ViewPager)findViewById(R.id.pager);
        mPager.setAdapter(new BkFragmentAdapter(getSupportFragmentManager()));
```

```
    //무한 스크롤 구현 부분
    mPager.setOnPageChangeListener(new OnPageChangeListener() {

        public void onPageSelected(int position) {
            // TODO Auto-generated method stub
            if(position < COUNT) {
                mPager.setCurrentItem(position + COUNT, false);  // 두번째 false 인자값은
                //위치이동 에니메이션을 꺼준다
            } else if (position >= COUNT*2) {      // true 로 바꿔서 실행해보면 어떤얘긴지
                //단박에 알수 있다.
                mPager.setCurrentItem(position - COUNT, false);
            }
        }

        public void onPageScrolled(int position, float positionOffest,
        int positionOffestPixel) { }

        public void onPageScrollStateChanged(int state) { }
    });
}

//FragmentPager 구현
private class BkFragmentAdapter extends FragmentPagerAdapter{
    //생성자
    public BkFragmentAdapter(FragmentManager fm) {super(fm);}

    /**
    * 실제 뷰페이저에서 보여질 fragment 를 반환.
    * 일반 아답터(갤러리, 리스트뷰 등)의 getView 와 같은 역할
    * @param position - 뷰페이저에서 보여저야할 페이지 값( 0 부터 )
    * @return 보여질 fragment
    */
    @Override public Fragment getItem(int position) {
        return ArrayFragment.newInstance(position%COUNT);
        // 3 배로 리턴된 페이지를 보여준다. [1 2 3 4] [1 2 3 4] [1 2 3 4] 이렇게 보여지는 원리..
    }
```

```
    //뷰페이저에서 보여질 총 페이지 수
    @Override public int getCount() { return COUNT*3; }
    // 보여줄 페이지의 갯수를 3 배로 리턴한다.
}

//뷰 페이저의 페이지에 맞는 fragment 를 생성하는 객체
private static class ArrayFragment extends Fragment {
    int mPosition; //뷰 페이저의 페이지 값

    //fragment 생성하는 static 메소드 뷰페이저의 position 을 값을 받는다.
    static ArrayFragment newInstance(int position) {
        ArrayFragment f = new ArrayFragment(); //객체 생성
        Bundle args = new Bundle();     //해당 fragment 에서 사용될 정보 담을 번들 객체
        args.putInt("position", position);    //포지션 값을 저장
        f.setArguments(args);        //fragment 에 정보 전달.
        return f;            //fragment 반환
    }

    //fragment 가 만들어질 때
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mPosition = getArguments() != null ? getArguments().getInt("position") : 0;
        // 뷰페이저의 position 값을  넘겨 받음
    }

    //fragment 의 UI 생성
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.layout1+mPosition, container, false);
        //미리 알고 있는 레이아웃을 inflate 한다.

        if(mPosition==0){
            v.findViewById(R.id.btn1).setOnClickListener(mButtonClick);
            v.findViewById(R.id.btn2).setOnClickListener(mButtonClick);
        } else if(mPosition==1) {
```

```
                v.findViewById(R.id.btn3).setOnClickListener(mButtonClick);
                v.findViewById(R.id.btn4).setOnClickListener(mButtonClick);
            } else{
            }
            return v;
        }
    }
}
```

7. ViewPager.OnPageChangeListener 이벤트 전달 순서

ViewPager 의 OnPageChangeListener 를 통해서 페이지가 변경 되었을 때 onPageScrollStateChanged 으로 변경된 페이지 Position 을 CallBack 받을 수 있다.

ViewPager.OnPageChangeListner 에는

- onPageScrollStateChanged(int state)
- onPageScrolled(int position, float positionOffset, int positionOffsetPixels)
- onPageSelected(int position)

3 개의 method 를 가진다.

```
setOnPageChangeListener(new OnPageChangeListener() {

    @Override
    public void onPageSelected(int state) {

    }

    @Override
    public void onPageScrolled(int position, float positionOffset,
            int positionOffsetPixels) {

    }

    @Override
    public void onPageScrollStateChanged(int position) {

    }
});
```

이번에 작업하는 UI 는 ViewPager 의 0 번 탭에서 1 번 탭으로 이동할 때 특정한 애니메이션을 실행한다. 일반적인 탭이라면 별 복잡한 처리가 필요하지 않겠지만, ViewPager 의 특성 상 0 번 탭에서 1 번 탭 이동이 swipe 일 수도 있고, 탭을 꼭 집어 이동하는 케이스도 있다.

실제 케이스는 더 복잡하지만 일단 서두는 이 정도로 해 두고, ViewPager 의 상태가 어떻게 바뀌어가는지 알아야 하기에 ViewPager.OnPageChangeListner 를 이용했다.

1. Swipe (A->B)

onScrollStateChanged: dragging
onPageScrolled: 드래그 하는 동안 계속 호출
어느 정도 드래그 되었다면
onScrollStateChanged: settling
onPageSelected: B
onPageScrolled: 제대로 안착할 때 까지 계속 호출
onScrollStateChanged: idle

2. Swipe 중도 취소 (A->B 로 가려다->A)

onScrollStateChanged: dragging
onPageScrolled: 드래그 하는 동안 계속 호출
중간에 취소되었다면 (손을 놓아버리는)
onScrollStateChanged: settling
onPageScrolled: 제대로 안착할 때 까지 계속 호출
onScrollStateChanged: idle
1 번 경우와 거의 비슷하지만 onPageSelected 가 settling 다음에 호출되지 않은 부분이 다르다.

3. 탭 선택 (A->B)

onScrollStateChanged: settling
onPageSelected: B
onPageScrolled: 제대로 안착할 때 까지 계속 호출
onScrollStateChanged: idle
onPageSelected 가 먼저 호출될 줄 알았는데 onScrollStateChanged 가 먼저 호출되었다.

내 문제의 요구사항은 다음과 같다.

- 0 번 탭에서 다른 탭을 선택할 경우 애니메이션 실행
- 0 번 탭에서 1 번 탭으로 스와이프 할 경우 애니메이션 실행
- 0 번 탭에서 1 번 탭으로 다가가 중도에 취소할 경우 반대 애니메이션 실행
- 그 외의 경우엔 애니메이션 실행 안함

두번째 경우의 취소만 없었어도 복잡도가 훨씬 줄어들었을 것 같은데, 이것까지 고려하려다 보니 꽤나 복잡해졌다.

내 문제를 해결하기 위해선 각 경우를 판단할 수 있어야 해서 대략 다음과 같이 처리했다.

onPageScrollStateChanged : state 를 별도의 변수에 저장해 둔다. SCROLL_STATE_SETTLING 의 경우 이 다음에 바로 onPageSelected 가 호출되기 때문에 원래 어느 탭에서 시작된 애니메이션인지 식별하기가 어렵다. 따라서 이동을 시작하는 위치와 끝의 탭 위치를 알기 쉽게 하려고 prevPosition 과 targetPosition 두개의 변수를 두었고, prevPosition 은 SCROLL_STATE_SETTLING 상태일 때 설정했다.
onPageSelected : settle 대상 페이지가 지정된 경우 호출된다. 따라서 targetPosition 변수에 넘어온 값을 저장한다.
onPageScrolled : 여기 넘어온 position 값 자체는 활용하기가 쉽지 않다. settle 상태에선 대상 페이지가 넘어오고, settle 하기 전인 dragging 상태에선 출발 페이지가 넘어온다. 그래서 인자로 넘어온 페이지값은 그냥 무시한다.
대충 코드는 다음과 같다.

```
int scrollState=0;
int targetPage= 0;
int prevPage= 0;

public void onPageScrollStateChanged(int state) {
  if (pagerScrollState == ViewPager.SCROLL_STATE_SETTLING) {
    prevPage = targetPage;
  }
  scrollState = state;
}

public void onPageSelected(int position) {
  targetPage = position;
}
```

진짜배기는 여기부터인데, 0 번에서 1 번으로 한창 스와이프 중인지, 0 번에서 1 번으로 스와이프 하다가 취소해서 되돌아오는 중인지, 0 번에서 1 번으로 충분히 스와이프를 했거나 0 번에서 1 번으로 탭을 직접 눌러서 이동하는 중인지를 onPageScrolled 에서 판단을 해서 애니메이션 처리를 해야 했다.

알고 있는 정보로 케이스를 조합해보자면 다음과 같다.

dragging 상태인데 target 이 0 이면 0 번 탭에서 한창 손으로 끌고 있는 중이므로 해당.
settling 상태인데 target, prev 모두 0 이면 드래깅하다 관둔 경우이므로 해당.
settling 상태인데 target=1, prev=0 이면 드래깅 충분히 해서 안착하는 경우이므로 해당.

```
boolean needAnimation() {
  switch(scrollState) {
    case SCROLL_STATE_DRAGGING:
      if (targetPage == 0) {
        return true;
      }
      break;
    case SCROLL_STATE_SETTLING:
      if (targetPage == 1 && prevPage == 0 ) {
        return true;
```

```
        } else if( targetPage ==0 && prevPage ==0) {
          return true;
        }
       break;
   }
   return false;
}
```

내 문제에선 이것 말고도 몇가지 자질구래한 경우가 더 있어서 조건문이 몇개 더 붙긴 했지만 대략 이 정도 내용을 염두에 둔다면 pager 의 tab 이동과 관련한 상태 검색은 충분히 구현할 수 있을 것이다.

8. PageAdapter 내에서 View 생성시 문제점

PageAdapter 에서 Fragment 를 새로 생성하여 View 를 만든 후 notifyDataSetChanged 를 하게 되더라도 Fragment 내부의 View 들이 새로고침이 되지 않는 문제점이 발생합니다.

이를 해결하기 위해서는 instantiateItem 에서 View 들에 대한 정보를 저장해놓은 다음 데이터 업데이트시 View 정보를 불러와 refresh 하도록 구현하길 권합니다.

```
SparseArray< View > views = new SparseArray< View >();

@Override
public Object instantiateItem(View container, int position) {
    View root = //refresh 할 뷰
    ((ViewPager) container).addView(root);
    views.put(position, root);
    return root;
}

@Override
public void destroyItem(View collection, int position, Object o) {
    View view = (View)o;
    ((ViewPager) collection).removeView(view);
    views.remove(position);
    view = null;
}

@Override
public void notifyDataSetChanged() {
    int key = 0;
    for(int i = 0; i < views.size(); i++) {
       key = views.keyAt(i);
       View view = views.get(key);
       //refresh 할 작업들
    }
    super.notifyDataSetChanged();
}
```

이 처럼 작업하여 자원의 낭비를 막을 수 있습니다.

9. Reference

http://arabiannight.tistory.com/53

http://blog.daum.net/mailss/16

http://blog.naver.com/PostView.nhn?blogId=huewu&logNo=110116958816

http://emstudio.kr/2014/01/12/94/

http://kingorihouse.tumblr.com/post/87079690019/android-viewpager-onpagechangelistener

http://www.kmshack.kr/android-viewpager-%EC%84%B1%EB%8A%A5%ED%96%A5%EC%83%81-%EB%B0%A9%EB%B2%95

http://androphil.tistory.com/6

http://androidtrainningcenter.blogspot.kr/2012/12/swipe-screens-in-android-viewpager.html
http://blog.daum.net/mailss/19
http://kingorihouse.tumblr.com/post/87079690019