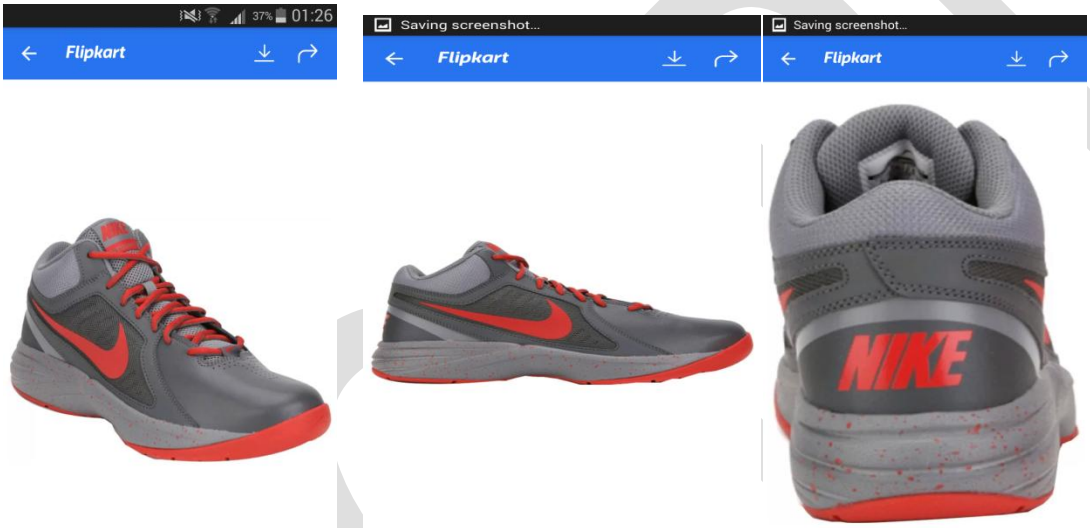
Image Slider

An image slider widget is one of the modern widget that we come across in most of the leading apps. Basically, an image slider helps the user to view a series of pictures simply by using the swiping action of the finger.

The best example of Image slider is **Flipkart** app where we see the pictures of the products we wish to purchase.



In the above examples we can swap between the images.

Here is the explanation of how to code an image slider in our android project.

Step1: Firstly, we'll need to have a ViewPager element in our layout file.

Android provides us with the ViewPager class that allows the user to flip left and right through pages of data. To use it, you'll have to put the <ViewPager> element (from the support library) inside your main XML layout file that'll contain multiple child views (which will be the different pages).

```
<android.support.v4.view.ViewPager
    android:id="@+id/view_pager"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    >
    </android.support.v4.view.ViewPager>
```

Step 2: In our onCreate() Activity method, we'll instantiate our custom PagerAdapter implementation and bind it to our ViewPager object.

Once this code is placed in say activity_main.xml, we'll have to hook the layout to a PagerAdapter which will populate our ViewPager with pages. Basically a PagerAdapter is responsible for creating the content for each page.

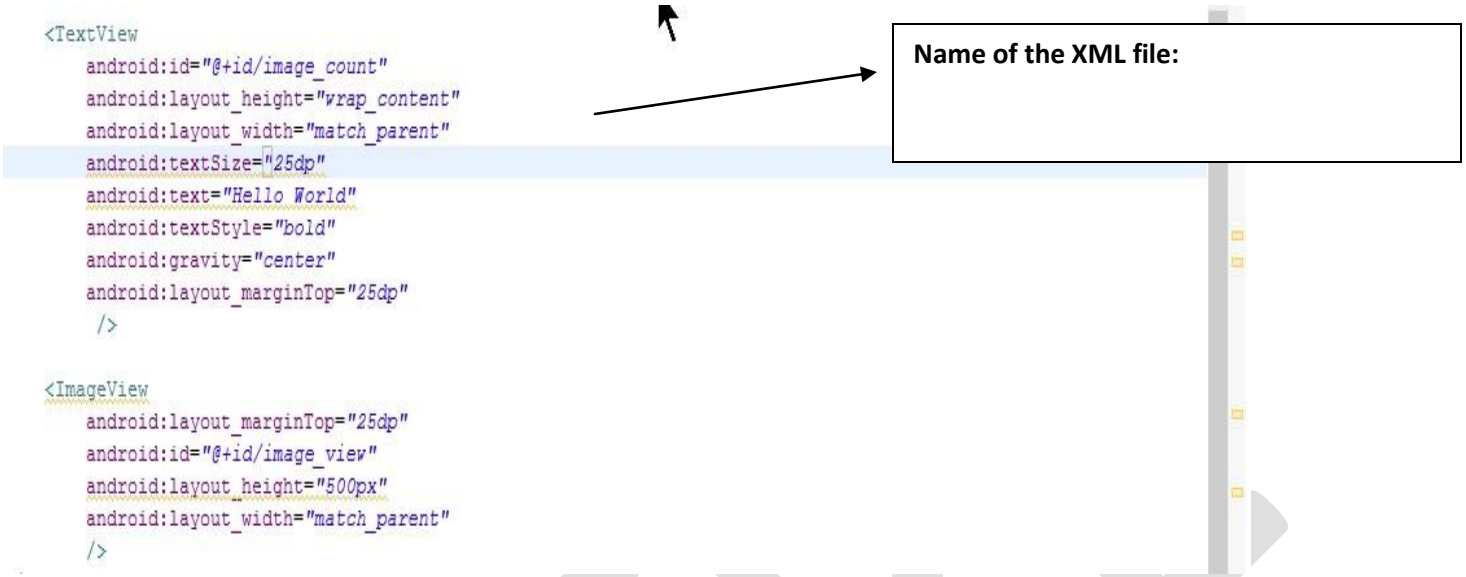
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

  ViewPager  viewpager = (ViewPager) findViewById(R.id.view_pager);
   CustomSwipeAdapter adapter = new CustomSwipeAdapter(this);
    viewpager.setAdapter(adapter);
}
```

CustomSwipeAdapter **extends PageAdapter class?**

Step 3: Create a new Xml layout defining ImageView on which the images will be shown one at a time.

- Right-click on the layout folder and create a new xml layout file.
- Code a TextView and Imageview in the xml file.
- Give both of them appropriate ids.



Step 4 : Copy or move all the images you want to include in the image slider into the res/drawable folder & give suitable names to all images.

Step 5 : Create a java class file which will extend the PagerAdapter class.

- Right-click on src/package and select new java class.
- The class should extend the PagerAdapter class.
- All the abstract methods of the PagerAdapter class should be implemented & rest should be overridden.

Following are the things that should be coded in the new java class file.

a. Create a one dimensional array that'll contain drawable resource IDs. (IDs should correspond to the names of images in drawable folder.

```
private int[] image_resources = {R.drawable.image1,R.drawable.image2,R.drawable.image3,
        R.drawable.image4,R.drawable.image5};
```

b. Finally it's time to write our custom pager adapter implementation that'll do the real job of populating content pages within our ViewPager.

```
public class CustomSwipeAdapter extends PagerAdapter{

    private int[] image_resources = {R.drawable.image1,R.drawable.image2,R.drawable.image3,
            R.drawable.image4,R.drawable.image5};
    private LayoutInflater layout_inflater;
    private Context ctx;

    public CustomSwipeAdapter(Context ctx)
    {
        this.ctx=ctx;
    }

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

    @Override
    public boolean isViewFromObject(View view, Object o) {
        // TODO Auto-generated method stub
        return (view==(LinearLayout)o);
    }

    @Override
    public Object instantiateItem(ViewGroup container, int position) {
        layout_inflater = (LayoutInflater) ctx.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View item_view = layout_inflater.inflate(R.layout.swipe_layout,container,false);
        ImageView image_view = (ImageView) item_view.findViewById(R.id.image_view);
        TextView text_view = (TextView) item_view.findViewById(R.id.image_count);

        image_view.setImageResource(image_resources[position]);
        text_view.setText("image : "+position);
        container.addView(item_view);
        return item_view;
    }

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



}
```

The above overridden methods are explained as follows:

i. **getCount()** – This method should return the number of views available, i.e., number of pages to be displayed/created in the ViewPager.

ii. **instantiateItem()** – This method should create the page for the given position passed to it as an argument. In our case, we inflate() our layout resource to create the hierarchy of view objects and then set resource for the ImageView in it. Finally, the inflated view is added to the container (which should be the ViewPager) and return it as well.

iii. **destroyItem()** – Removes the page from the container for the given position. We simply removed object using removeView() but could've also used removeViewAt() by passing it the position.

iv. **isViewFromObject()** – The object returned by instantiateItem() is a key/identifier. This method checks whether the View passed to it (representing the page) is associated with that key or not. It is required by a PagerAdapter to function properly. For our example, the implementation of this method is really simple, we just compare the two instances and return the evaluated boolean.

Run the app and experience the effect of image slider.

Note:

It is important to understand that as the user navigates to a particular page, the one next to it is generated by calling ***instantiateItem()*** while the one before the previous one gets destroyed by calling ***destroyItem()***.

This caching limit (destruction and rebuilding limit) can be specified by the ***setOffscreenPageLimit()*** method on the ViewPager object which is set to 1 by default. Increasing this value to a higher number leads to a smoother navigation as far as the animations and interactions are concerned as everything is retained in memory but then it can also cause a memory overhead affecting the app's performance.

The complete project and .apk file is available along with the documentation online on http://android.suven.net