Tutorials \ *Android SDK* \ *Rating:* ★★★★

Android User Interface Design: Horizontal View Paging

Shane Conder & Lauren Darcey on Oct 18th 2011 with 37 Comments and 0 Reactions

Tutorial Details

Technology: Android SDK, Compatibility Package
Difficulty: Intermediate
Estimated Completion Time: 45-60 Minutes

This entry is part 19 of 21 in the series Android User Interface Design

- Android User Interface Design: Layout Basics
- Android User Interface Design: Basic Buttons
- Android User Interface Design: Linear Layouts
- Android UI Fundamentals Challenge: LinearLayout
- Android User Interface Design: Relative Layouts
- Android UI Fundamentals Challenge: RelativeLayout
- Android User Interface Design: Table Layouts
- Android User Interface Design: Frame Layouts
- Android User Interface Design: Building a ListView Application
- Android User Interface Design: Working With Fragments
- Android User Interface Design: Building Application Preference Screens
- Android User Interface Design: Basic Text Controls
- Android User Interface Design: Basic Image Controls
- Android User Interface Design: Working With Dialogs
- Android User Interface Design: Working With Date Picker Dialogs
- Android User Interface Design: Password Confirmation
- Android User Interface Design: The Basics of Control Focus Order
- Android User Interface Design: Radio Buttons
- Android User Interface Design: Horizontal View Paging
- Android User Interface Design: Icon Design
- Android User Interface Design: Creating a Numeric Keypad with GridLayout

Perhaps you've seen some of the new user interface features available as part of the Android compatibility package. One such feature, horizontal view paging, allows for easy left and right swipes to load different screens (pages), controlled by a single Activity. This feature has been showcased in several high profile applications like the Android Market application and the Google+ Android client.

There are a number of classes in the Android compatibility package that can be used to implement horizontal page swiping behavior in your Android packages. The ViewPager control (android.support.v4.view.ViewPager) provides the horizontal swiping behavior. It can be used within your layouts much like a Gallery or other adapter-populated user interface control would be. The PagerAdapter (android.support.v4.view.PagerAdapter) class is used to define the data displayed by the ViewPager control. Today we'll look at a simple example of how to use these classes to provide swiping behavior.

Step 0: Getting Started

We provide the full source code for the sample application discussed in this tutorial. You can download the sample source code we provide for review.

Step 1: Use the Compatibility Package

Horizontal view paging is based upon APIs only available with the Android Compatibility package v4, Revision 3; these APIs are not available in the standard Android SDK at this time. Therefore, you will need to add the Android compatibility package to your Android project to access the appropriate APIs.

To add the Android Compatibility package to your Eclipse Android project, right-click on the project in the Project Explorer. Choose Android Tools, Add Compatibility Library. You will now see the android-support-v4.jar file in your Referenced Libraries project folder. This means you have successfully added the package to your project and can now start using it.

Step 2: Define a ViewPager

Next, you'll need to define a ViewPager control in your layout resource file. In our simple example, we update the main.xml layout resource used by our Activity class, and define a ViewPager control within that layout. This control must be referenced by its fully-qualified name: android.support.v4.view.ViewPager.

For example, here's the updated main.xml layout resource with a ViewPager defined:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
<android.support.v4.view.ViewPager
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/myfivepanelpager"/>
</LinearLayout>
```

ViewPager controls often take up the entire screen, but this need not be the case. For this example, we will display five different layout “pages”, thus we call our ViewPager control by a unique identifier labeled myfivepanelpager.

mindfiresolutions
REACH YOUR TARGET AUDIENCE ACROSS MOBILE PLATFORMS
Titanium | PhoneGap | RhoMobile
CROSS-PLATFORM MOBILE DEVELOPMENT
Certified Specialists | 8+ Yrs App Dev Experience
FREE TRIAL

Step 3: Create Page Layout Resources

Next, you’ll want to create a set of resources that will make up the “pages” or “panes” for horizontal swiping. You can use the same layout resource file for each page and add different content, or you can load completely different layout resources for the individual pages. For this example, we created five separate layout resource files, called farleft.xml, left.xml, middle.xml, right.xml, and farright.xml. Each layout resource has different contents to display. The contents of each layout resource are up to you. You can use static or dynamic controls. To keep this example simple, we’ll stick with static controls like TextView and ImageView controls. For the far left and far right pages, we’ll include some Button controls.

This image shows the five different layout resource file results:

There is nothing special about the implementation of these layout files. Don’t forget to implement any Button onClick handlers in your Activity class. These layout resources will be loaded by the PagerAdapter at runtime for display on the screen. For implementation details, see the source code that accompanies this project.

Step 4: Implement a Custom PagerAdapter

Your ViewPager needs a data adapter to determine and load the appropriate content for each page the user swipes to. We have named our layout resource file “pages” in the order we want them to display, from far left to far right.

When you extend the PagerAdapter class, you’ll need to implement several key methods.

First, you’ll need to define the size of your paging range. In this case, we have a set of five pages to display. Therefore, you’ll want the getCount() method of the MyPagerAdapter class to return a page size of 5.

Next, you need to implement the instantiateItem() method to inflate the appropriate layout resource file, depending on the user’s swipe position. The farthest page to the left is in position 0, the next page to the right is position 1, and so on. The instantiateItem() method uses the LayoutInflater service to inflate the specific layout and add it to the collection used by the ViewPager.

This image shows the five different layout resource files and their “positions” in the terms of paging order:

The last important method you need to implement is the destroyItem() method, which removes the specific layout from the collection used by the ViewPager when it is no longer being displayed.

Here is a basic implementation for a five-page horizontal pager adapter, called MyPagerAdapter, which implements these core methods as well as a few others:

```
private class MyPagerAdapter extends PagerAdapter {
        public int getCount() {
            return 5;
        }
        public Object instantiateItem(View collection, int position) {
            LayoutInflater inflater = (LayoutInflater) collection.getContext()
                    .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            int resId = 0;
            switch (position) {
            case 0:
                resId = R.layout.farleft;
                break;
            case 1:
                resId = R.layout.left;
                break;
            case 2:
                resId = R.layout.middle;
                break;
            case 3:
                resId = R.layout.right;
                break;
            case 4:
                resId = R.layout.farright;
                break;
            }
            View view = inflater.inflate(resId, null);
            ((ViewPager) collection).addView(view, 0);
            return view;
        }
        @Override
        public void destroyItem(View arg0, int arg1, Object arg2) {
            ((ViewPager) arg0).removeView((View) arg2);
        }
        @Override
        public boolean isViewFromObject(View arg0, Object arg1) {
            return arg0 == ((View) arg1);
        }
        @Override
        public Parcelable saveState() {
            return null;
        }
}
```

Step 5: **Bind MyPagerAdapter**

Lastly, you need to update the onCreate() method of your Activity class to bind your MyPagerAdapter to the ViewPager control defined in your main.xml layout resource file.

You can also take this time to set the initial position of the pager. By default, it would start at position 0 (the far left layout with the simple Button control). However, we want to allow the user to swipe left and right so we set the initial position of the ViewPager to the middle layout (the monkey in the middle) using the setCurrentItem() method.

```java
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    MyPagerAdapter adapter = new MyPagerAdapter();
    ViewPager myPager = (ViewPager) findViewById(R.id.myfivepanelpager);
    myPager.setAdapter(adapter);
    myPager.setCurrentItem(2);
}
```

Now if you run your application, you'll begin with the monkey in the middle page, and be able to swipe two pages left or right, as shown here:

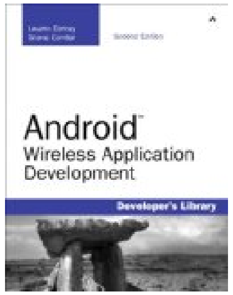
Conclusion

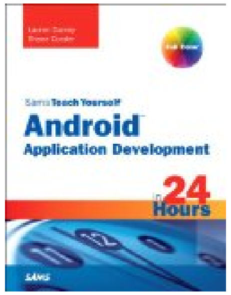
The horizontal view pager user interface control is a neat user interface control made available to Android developers through the Android compatibility package. Data for the individual "pages" is managed by a special data adapter called a PagerAdapter. There are also classes within the compatibility library for building fragment-compatible data adapters for driving ViewPager controls.

About the Authors

Mobile developers Lauren Darcey and Shane Conder have coauthored several books on Android development: an in-depth programming book entitled *Android Wireless Application Development, Second Edition* and *Sams Teach Yourself Android Application Development in 24 Hours, Second Edition*. When not writing, they spend their time developing mobile software at their company and providing consulting services. They can be reached at via email to androidwirelessdev+mt@gmail.com, via their blog at androidbook.blogspot.com, and on Twitter @androidwireless.

Need More Help Writing Android Apps? Check out our Latest Books and Resources!





Series Navigation

«Android User Interface Design: Radio Buttons Android User Interface Design: Icon Design»

By Shane Conder & Lauren Darcey

Rollover to read this author's bio or click through to see a full list of posts by this author.

Note: Want to add some source code? Type <pre><code> before it and </code></pre> after it. Find out more

37 comments

Leave a message...

Discussion Community Shar

Jade Byfield · 4 months ago

Excellent tutorial. I'm trying to modify your Adapter class so that it takes in an ArrayList of bitmaps to be displaye as imageViews. Here's my current implementation of instantiateItem(). Anyone know what I'm doing wrong?

```
public class ViewHolder {
ImageView photo;
}

@Override
public Object instantiateItem(View v, int position) {

ViewHolder holder = new ViewHolder();

LayoutInflater inflater = (LayoutInflater) v.getContext()
.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

View view = inflater.inflate(R.id.ivCurrentImage, null);
((ViewPager) v).addView(view, 0);


Bitmap b = photos.get(position);


holder.photo = (ImageView)view.findViewById(R.id.ivCurrentImage);
holder.photo.setImageBitmap(b);

return view;
}
```

1 ^ | v · Reply · Share ›

Jade Byfield ➜ Jade Byfield · 4 months ago

Nevermind guys, I got it to work. Logcat was complaining about a ResourceNotFoundException. Apparentl
I was referencing the id of my imageView rather than the resource id of the layout file

These lines

```
View view = inflater.inflate(R.id.ivCurrentImage, null);
((ViewPager) v).addView(view, 0);
```

Should be changed to

```
View view = inflater.inflate(R.layout.your_page_name, null);
((ViewPager) v).addView(view, 0);
```

Hope this helps someone else who had a similar issue!

0 ^ | v · Reply · Share ›

Sameer · 5 days ago

Your tutorial is good but i am disappointed that you did not respond to query of Reader generally. See the same example here http://androidtrainningcenter....

0 ^ | v · Reply · Share ›

Vidya · 24 days ago

Thanks a lot :-)

0 ^ | v · Reply · Share ›

Frank van der Horst · a month ago

First thank you for this great tutorial. I implemented this exampe and it works great! My only question is, "Can thi
example be supplemented with a actionbar with title???

0 ^ | v · Reply · Share ›

MiralDesai · 3 months ago

I'm having trouble with this. The swiping works fine but the app crashes when I try and click any of the onClicks I
have on other activitys other than my starting position. None of my TextView onClicks are recognised, when click
the text they do nothing. When clicking on imageviews they force close the app. The ImageViews work on my firs
the one you first see when the app launches and but my TextViews don't on this page and every other. Then the

Are there any solutions?

I followed the tutorial to the letter.

0 ^ | v · Reply · Share ›

voghdev · 4 months ago

very useful. Worked like a charm. Thanks!!!

0 ^ | v · Reply · Share ›

jayraj · 5 months ago

Thanks for this great tutorial...

0 ^ | v · Reply · Share ›

realtebo · 5 months ago

I copy/pasted your code and everyting is working, but ...

.. how can I use findViewById() from activity to (for example) change an image in one the three dinamically inflate

please, it's important, i'm blocked on this from 2 days !

0 ^ | v · Reply · Share ›

Janko · 5 months ago

Cool! Thanks, God bless U!

0 ^ | v · Reply · Share ›

lucano · 5 months ago

Excellent!!!!!!!!!!!! Drew your comment saved me long hours and work.

0 ^ | v · Reply · Share ›

Gary Blakely · 6 months ago

Instead of inflating and destroying, isn't there some way to inflate them all at the beginning then just let the user swipe between them. This seems viable when there are only a few pages and would take less resource.

0 ^ | v · Reply · Share ›

venkata reddy · 7 months ago

thank you for this example.

it there any possibility to do this example as circular swiping i.e circular viewpager...

0 ^ | v · Reply · Share ›

parag · 8 months ago

I have found this error in Icecream version

```
06-19 11:47:30.518: E/AndroidRuntime(1131): java.lang.IllegalStateException: Observer
android.support.v4.view.ViewPager$PagerObserver@41d68568 was not registered.
06-19 11:47:30.518: E/AndroidRuntime(1131): at android.database.Observable.unregisterObserver(Observable
06-19 11:47:30.518: E/AndroidRuntime(1131): at
android.support.v4.view.PagerAdapter.unregisterDataSetObserver(PagerAdapter.java:284)
06-19 11:47:30.518: E/AndroidRuntime(1131): at android.support.v4.view.ViewPager.setAdapter(ViewPager.java
```

0 · Reply · Share ›

Phil · 8 months ago

Thanks for this. I've spent a while going round in circles trying to figure this out and you've explained it perfectly.

I have question on populating the data on the 'page':

I've modified the MyPagerAdapter class so that I pass in a list of objects which contains the data to display.

Then, in the instantiateItem() method, I can get the object from the list based on the swipe position passed in. Fi
populate the view with the object data before it is returned, e.g:

```
object = _objects.get(position);

TextView descriptionView = (TextView)view.findViewById(R.id.description);
description.setText(object.GetDescription());
..
etc

return view
```

Is this the correct way to do it or should I perhaps farm this part out to another class?

Thanks

0 · Reply · Share ›

Jova · 9 months ago

I implemented an activity of my app following this tutorial and it works good, but there is a problem that sometime
appears, "java.lang.OutOfMemoryError: bitmap size exceeds VM budget". How can I remove it? I use only four im
the amount of memory used by them is 291,3kb... Why there is this error? o.o

Thanks

Jova

0 · Reply · Share ›

Naresh · 9 months ago

Hey Thanks for Tutorial...!!!

I have one problem.

In main.xml I have added three buttons and then added

But now when I run app these three buttons get duplicated that is on main screen it shows 6 buttons.

How can I resolve it...???

Thank U....

0 · Reply · Share ›

pete · 10 months ago

Nice tut. How would I add a text view?

0 · Reply · Share ›

Dilshan · 11 months ago

Nice Post

thanks a lot

0 · Reply · Share ›

Nanjou · 11 months ago

Hi, thanks for this one !

I would know if is it possible to include vertical swiping ..! it could be very interesting, if anyone had some way.. it really appreciated ! :)

0 · Reply · Share ›

Pawanv · 11 months ago

Hi Shane & Lauren, I used this tutorial and it worked fine but i also wanted to give next and previous Button to iterate the layout. And to give action on button I used code above given by Drew for button action and that is wor but how do I go onto next and previous screen because I cannot use setContentView() method in MyPagerAdapt because its not an activity . please give me some clue . thanks

0 · Reply · Share ›

Pawanv · 11 months ago

This article really very useful...and thanks a lot to give time for writing so helpful post..keep it up..really apreciate

0 · Reply · Share ›

Ruben · a year ago

how can you put a title bar in this example? I want to put a title bar like market for example..

0 · Reply · Share ›

DMVP · a year ago

Very Nice Article, I implemented it with tabs and it works great! Thanks so much for the contribution. Im learning so much from you guys...

0 · Reply · Share ›

Elisa · a year ago

Hi, this was a very useful tutorial. I was able to create viewpaging with webviews. Thanks!

0 · Reply · Share ›

Michael · a year ago

This is very clear, thanks! I do have a question about persisting data in, say for instance, an EditText view. I noticed that after shifting two pages over and back, the text I entered is gone. Do you have advice on how to har

0 · Reply · Share ›

Janis · a year ago

Hi,

Is it possible for the text to flow from webview to another?

(I'm creating a book reader and I chose this horizontal pager to behave as book pages. When I need to increase size, the text should flow to another page (another webview))

Thanks in advance,
Janis

0 · Reply · Share ›

Jose Granja · a year ago

Hi! The tutorial looks great!

But how about I initialize the ViewPager with a list os IDs (for example book IDS) and for each one of this IDs I ne to my webservice and pull out the info I need to display! How am I going to do this? What it try to say is that if it's to combine this with an AsyncTask? For me it's a key point! I don't want to have to preload all the info before goi view adapter! Does that make sense to you? Maybe showing in the page the loading spining dialog while the inf loaded!

Regards

0 · Reply · Share ›

Jacob · a year ago

THANK YOU!
I was searching such a solution for my problem for many hours today :) Helped me a lot!

Jacob

0 · Reply · Share ›

Drew · a year ago

After much frustration and pulling my hair out over this issue, I have solved it! At least for me. Assuming you use the tutsplus tutorial like I did, you have separate XML files for your screens. Now, I assume those layout XMLs co layouts within them (ie. LinearLayout, RelativeLayout, etc.). Now, those layouts contain your button and other wi What you have to do to be able to findViewById is, in the actual switch statement in the instatiateItem method, ini your button in this way:

```
public Object instantiateItem(View collection, int position) {

LayoutInflater inflater = (LayoutInflater) collection.getContext()
.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

int resId = 0;
switch (position) {
case 0:
resId = R.layout.lighttab;
View view = inflater.inflate(resId, null);
RelativeLayout layout=(RelativeLayout)view.findViewById(R.id.relLayout);
Button button=(Button)layout.findViewById(R.id.button);
((ViewPager) collection).addView(view, 0);
return view;
case 1:
resId = R.layout.securitytab;
...
...
return view;
}
}
```

So...first you have to inflate the view, then initialize the layout held within the view by casting the type of layout (RelativeLayout) and calling .findViewById(resource id). Then, you initialize the actual widget you're looking for b the same, but casting the widget type (Button) and calling .findViewById(resource id).

This worked for me, so I hope this saves you some trouble! Took forever for me to figure out.

0 ⌃ ⌄ · Reply · Share ›

Elisa ➜ Drew · a year ago

Thank you for this, I was having trouble referring to the buttons in my view in PagerAdapter.

0 ⌃ ⌄ · Reply · Share ›

Henrik Pedersen · a year ago

How would you go about adding a spinner control or dynamically setting a button handler inside this view?

I can see the XML is inflated dynamically and therefore you can't just use findViewById()...

Can someone help me out here, if i try to use findViewById, even on the newly inflated view it keeps returning a r
pointer!

0 ^ | ⌄ · Reply · Share ›

Kevin ➜ Henrik Pedersen · 5 months ago

I know this is a year too late, but for anyone else facing this problem you can use findView ById with inflatec
views.

With inflated views, using:
<pre name="code" class="java" myListView = (ListView)findViewById(R.id.account_list);
will return a null and most likely give you a null pointer down the line if you try to do anything with myListVie

The correct usage is:
<pre name="code" class="java" myListView = (ListView)myInflatedView.findViewById(R.id.account_list);
where myInflatedView is something like <pre name="code" class="java" View myInflatedView =
layoutinflater.inflate(R.layout.search_account, null);

This can be used with any type of widget from textviews to listviews. Hope that helps

0 ^ | ⌄ · Reply · Share ›

Nicolas · a year ago

Hi,
First, thank you for this tutorial,

In file main.xml, before android.support.v4.view.ViewPager, i added graphical component ImageView.
In file right.xml, i added button. In the class RightFragment, I defined function (setUpImageView (View v)) to be ca
time a click on this button is produced. I would like, from this method, change src parameter of component Image
main.xml.

Could you help me ?

0 ^ | ⌄ · Reply · Share ›

Stan · a year ago

Oh, I forgot to mention this. But some of the code blocks in this tutorial seem to have the HTML-entity rather thar
the actual lower/greater than signs (or sharp brackets, whatever you prefer).

Might look a bit confusing to the readers. :)

0 ^ | ⌄ · Reply · Share ›

Matt · a year ago

Thank you for this great tutorial !

And for going the extra mile by loading the XML of each page.

Matt

0 ^ | v · Reply · Share ›

Stan · a year ago

Thank you, Shane & Lauren, for this tutorial. I was already trying figuring this out myself via the Android Developers Blog, but I was having some difficulties.

Cheers.

0 ^ | v · Reply · Share ›

ALSO ON **MOBILETUTS+** What

Build a Multiplayer Minesweeper Game: Client-Side Creation
8 comments · 6 days ago
Vukašin Manojlović — Thanks! Can't wait for them!

iOS 6 and the Social Framework: Twitter Reques
2 comments · 5 days ago
Tommy Leirvik — Where did the appcelerator tuts go? greate with a alloy application.

Create a 3D Page Folding Animation: Polishing the Page Fold
4 comments · 12 days ago
Akiel — Thanks, Hamid.

Android SDK: Implementing Drag-and-Drop Functionality
1 comment · a month ago
Majo — works fine for me, but only if I add a "return tru DragEvent.ACTION_DRAG_STARTED.

Comment feed   Subscribe via email   DI