**PRACTICAL**

**OBJECT:**

Sliding pages with ViewPager, working with Toolbar

**THEORY:**

A RecyclerView is used to display lists. A toolbar allows actions list on the top of the application.

**CODE:**

**Create activity layout**

Add the ViewPager to your main xml layout. There are no dependencies for this component.

Listing 1: activity\_screen\_slide.xml

*<?****xml version="1.0" encoding="utf-8"****?>*

<**LinearLayout**

**xmlns:android="http://schemas.android.com/apk/res/android"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent"** >

<**android.support.v4.view.ViewPager**

**android:id="@+id/pager"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent"** />

</**LinearLayout**>

**Create row layout**

Each page in our ViewPager contains two TextView object. The first one just shows page number, while the second one shows movie title on the page. Create a new layout resource file with the following contents.

Listing 2: fragment\_screen\_slide\_page.xml

*<?****xml version="1.0" encoding="utf-8"****?>*

<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"**

**android:id="@+id/content"**

**android:orientation="vertical"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent"** >

<**TextView style="?android:textAppearanceMedium"**

**android:padding="12dp"**

**android:id="@+id/tvPageNumber"**

**android:lineSpacingMultiplier="1.2"**

**android:layout\_width="match\_parent"**

**android:layout\_height="wrap\_content"**

**android:text="viewpager demo"** />

<**TextView style="?android:textAppearanceMedium"**

**android:padding="16dp"**

**android:id="@+id/tvMovieName"**

**android:lineSpacingMultiplier="1.2"**

**android:layout\_width="match\_parent"**

**android:layout\_height="wrap\_content"**

**android:text="viewpager demo"** />

</**LinearLayout**>

**Create the adapter**

The ViewPager needs an adapter to generate the page contents dynamically. Create a new java class named MyPagerAdapter with the following contents.

Listing 3: MyPagerAdapter class

**import** android.os.Bundle;

**import** android.support.v4.app.Fragment;

**import** android.support.v4.app.FragmentManager;

**import** android.support.v4.app.FragmentStatePagerAdapter;

*//A simple pager adapter that represents different ScreenSlidePageFragment objects*

**public class** MyPagerAdapter **extends** FragmentStatePagerAdapter {

**private** String[] **pageItems**;

**public** MyPagerAdapter(FragmentManager fm, String[] pageItems) {

**super**(fm);

**this**.**pageItems** = pageItems;

}

@Override

**public** Fragment getItem(**int** position) {

ScreenSlidePageFragment screenSlidePageFragment = **new** ScreenSlidePageFragment();

Bundle args = **new** Bundle();

args.putInt(**"position"**, position);

args.putString(**"movie"**, **pageItems**[position]);

screenSlidePageFragment.setArguments(args);

**return** screenSlidePageFragment;

}

@Override

**public int** getCount() {

**return pageItems**.**length**;

}

}

MyPagerAdapter class extends FragmentStatePagerAdapter. In the constructor, a FragmentManager with the context of the calling activity is provided. The second argument is pageItems string array which is the data to be displayed on each page on sliding. We could also make this class the inner class of our ScreenSlidePagerActivity. In that case, we do not need this second argument of pageItems, which we could access directly from its container class ScreenSlidePagerActivity.

The class overrides which allows to generate the page from ScreenSlidePageFragment class, passing it the arguments of page number and movie title to be shown on generated page. The last method getCount simply returns the total number of pages available to swipe.

Listing 4: ScreenSlidePageFragment class

**import** android.content.Context;

**import** android.content.Intent;

**import** android.net.Uri;

**import** android.os.Bundle;

**import** android.support.v4.app.Fragment;

**import** android.view.LayoutInflater;

**import** android.view.View;

**import** android.view.ViewGroup;

**import** android.widget.TextView;

**public class** ScreenSlidePageFragment **extends** Fragment {

@Override

**public** View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

ViewGroup rootView = (ViewGroup) inflater.inflate(

R.layout.***fragment\_screen\_slide\_page***, container, **false**);

TextView tvPage = (TextView)rootView.findViewById(R.id.***tvPageNumber***);

TextView tvMovie = (TextView)rootView.findViewById(R.id.***tvMovieName***);

**int** pageNum = getArguments().getInt(**"position"**);

String movie = getArguments().getString(**"movie"**);

tvPage.setText( **"Page: "**+Integer.*toString*(pageNum));

tvMovie.setText( **"Movie Name: "**+ movie);

**return** rootView;

}

}

The ScreenSlidePageFragment class contains only one method that initializes the page along with the values of the two TextView objects and returns the page.

Listing 5: ScreenSlidePagerActivity class

**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.FragmentStatePagerAdapter;

**import** android.support.v4.view.PagerAdapter;

**import** android.support.v4.view.ViewPager;

**import** android.support.v7.app.AppCompatActivity;

**public class** ScreenSlidePagerActivity **extends** AppCompatActivity{

**private static** String[] *movies* = {**"Matrix"**, **"Rush Hour"**, **"Twin Dragons"**};

*// The pager widget, which handles animation and allows swiping horizontally to access previous and next pages.*

**private** ViewPager **mPager**;

*// The pager adapter, which provides the pages to the view pager widget.*

**private** PagerAdapter **mPagerAdapter**;

@Override

**protected void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.***activity\_screen\_slide***);

*// Instantiate a ViewPager and a PagerAdapter.*

**mPager** = (ViewPager) findViewById(R.id.***pager***);

**mPagerAdapter** = **new** MyPagerAdapter(getSupportFragmentManager(), *movies*);

**mPager**.setAdapter(**mPagerAdapter**);

}

@Override

**public void** onBackPressed() {

**if** (**mPager**.getCurrentItem() == 0) {

*// If the user is currently looking at the first step, allow the system to handle the*

*// Back button. This calls finish() on this activity and pops the back stack.*

**super**.onBackPressed();

} **else** {

*// Otherwise, select the previous step.*

**mPager**.setCurrentItem(**mPager**.getCurrentItem() - 1);

}

}

}

Listing 5 defines ScreenSlidePagerActivity activity class. In onCreate method, we initialize the ViewPager and PagerAdapter objects to initialize the page sliding functionality and dynamic page generation. The onBackPressed method is handles the physical back button press event.

**Adding Toolbar for AppCompatActivity**

Update the above application to add a toolbar at the top of the screen. The first thing to do is to change the android:theme property in the application manifest file as:

**android:theme="@style/Theme.AppCompat.Light.NoActionBar"**>

If you do not, the application will crash on startup.

Listing 6: activity\_screen\_slide updated with Toolbar

<**LinearLayout**

**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"**

>

<**android.support.v7.widget.Toolbar**

**android:id="@+id/my\_toolbar"**

**android:layout\_width="match\_parent"**

**android:layout\_height="?attr/actionBarSize"**

**android:background="?attr/colorPrimary"**

**android:elevation="4dp"**

**android:theme="@style/ThemeOverlay.AppCompat.ActionBar"**

**app:popupTheme="@style/ThemeOverlay.AppCompat.Light"**/>

<**android.support.v4.view.ViewPager**

**android:id="@+id/pager"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent"** />

</**LinearLayout**>

Next we add the toolbar in activity\_screen\_slide.xml as shown in Listing 6.

Now add a menu directory in the res folder and add a main\_menu.xml file(res/menu/main\_menu.xml) with following contents.

Listing 7: main\_menu resource

<**menu xmlns:android="http://schemas.android.com/apk/res/android"**

**xmlns:app="http://schemas.android.com/apk/res-auto"**>

<**item**

**android:id="@+id/action\_favorite"**

**android:orderInCategory="300"**

**android:title="User"**

**android:icon="@drawable/baseline\_grade\_black\_18dp"**

**app:showAsAction="ifRoom"**></**item**>

<**item android:id="@+id/action\_settings"**

**android:title="@string/action\_settings"**

**app:showAsAction="never"**/>

<**item android:id="@+id/actionTheme"**

**android:title="@string/selectTheme"**

/>

</**menu**>

Also note that we are using an icon here, which you can download from the link given at the end of the lab. Otherwise you can omit this attribute.

Now back to main activity, initialize the Toolbar along with click event.

Listing 8: ScreenSlidePagerActivity with Toolbar added

**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.FragmentStatePagerAdapter;

**import** android.support.v4.view.PagerAdapter;

**import** android.support.v4.view.ViewPager;

**import** android.support.v7.app.AppCompatActivity;

**import** android.support.v7.widget.Toolbar;

**import** android.view.Menu;

**import** android.view.MenuItem;

**import** android.widget.Toast;

**public class** ScreenSlidePagerActivity **extends** AppCompatActivity{

*//FragmentActivity {*

**private static** String[] *movies* = {**"Matrix"**, **"Rush Hour"**, **"Twin Dragons"**};

*// The pager widget, which handles animation and allows swiping horizontally to access previous*

*// and next pages.*

**private** ViewPager **mPager**;

*// The pager adapter, which provides the pages to the view pager widget.*

**private** PagerAdapter **mPagerAdapter**;

**private** Toolbar **mTopToolbar**;

@Override

**protected void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.***activity\_screen\_slide***);

*// Instantiate a ViewPager and a PagerAdapter.*

**mPager** = (ViewPager) findViewById(R.id.***pager***);

**mPagerAdapter** = **new** MyPagerAdapter(getSupportFragmentManager(), *movies*);

**mPager**.setAdapter(**mPagerAdapter**);

*// Instanciate and set toolbar*

**mTopToolbar** = (Toolbar) findViewById(R.id.***my\_toolbar***);

setSupportActionBar(**mTopToolbar**);

}

@Override

**public void** onBackPressed() {

**if** (**mPager**.getCurrentItem() == 0) {

*// If the user is currently at the first step, allow the system to handle the*

*// Back button. This calls finish() on this activity and pops the back stack.*

**super**.onBackPressed();

} **else** {

*// Otherwise, select the previous step.*

**mPager**.setCurrentItem(**mPager**.getCurrentItem() - 1);

}

}

@Override

**public boolean** onCreateOptionsMenu(Menu menu) {

*// Inflate the menu; this adds items to the action bar if it is present.*

getMenuInflater().inflate(R.menu.***menu\_main***, menu);

**return true**;

}

@Override

**public boolean** onOptionsItemSelected(MenuItem item) {

*// Handle action bar item clicks here.*

**int** id = item.getItemId();

**if** (id == R.id.***action\_favorite***) {

Toast.*makeText*(**this**, **"Action clicked"**, Toast.***LENGTH\_LONG***).show();

**return true**;

}**else if** (id == R.id.***action\_settings***) {

Toast.*makeText*(**this**, **"Settings clicked"**, Toast.***LENGTH\_LONG***).show();

**return true**;

} **else if**(id==R.id.***actionTheme***){

Toast.*makeText*(**this**, **"Theme clicked"**, Toast.***LENGTH\_LONG***).show();

**return true**;

}

**return super**.onOptionsItemSelected(item);

}

}

In onCreateOptionsMenu, the menu is inflated dynamically. In the method onOptionsItemSelected, we show a Toast message on menu item click .

Useful resources:

Icons

<https://material.io/tools/icons/?icon=format_list_bulleted&style=baseline>

**ACTIVITIES**

**Activity 1**

Update the above application to add a movie class containing movie title, genre, director and description. Show all of these items on the pages for each movie.

**REVIEW QUESTIONS**

1. What is a ViewPager?
2. Why do we need an adapter?
3. Which methods should be override in inherited FragmentStatePagerAdapter?