public static class ExampleFragment extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// Inflate the layout for this fragment

return inflater.inflate(R.layout.example\_fragment, container, false);

}

}

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

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

android:orientation="horizontal"

android:layout\_width="match\_parent"

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

<fragment android:name="com.example.news.ArticleListFragment"

android:id="@+id/list"

android:layout\_weight="1"

android:layout\_width="0dp"

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

<fragment android:name="com.example.news.ArticleReaderFragment"

android:id="@+id/viewer"

android:layout\_weight="2"

android:layout\_width="0dp"

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

</LinearLayout>

FragmentManager fragmentManager = getSupportFragmentManager();

FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

ExampleFragment fragment = new ExampleFragment();

fragmentTransaction.add(R.id.fragment\_container, fragment);

fragmentTransactio fragmentTransaction n.commit();

// Create new fragment and transaction

Fragment newFragment = new ExampleFragment();

FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();

// Replace whatever is in the fragment\_container view with this fragment,

// and add the transaction to the back stack

transaction.replace(R.id.fragment\_container, newFragment);

transaction.addToBackStack(null);

// Commit the transaction

transaction.commit();

public static class FragmentA extends ListFragment {

...

// Container Activity must implement this interface

public interface OnArticleSelectedListener {

public void onArticleSelected(Uri articleUri);

}

...

}

public static class FragmentA extends ListFragment {

OnArticleSelectedListener listener;

...

@Override

public void onAttach(Context context) {

super.onAttach(context);

try {

listener = (OnArticleSelectedListener) context;

} catch (ClassCastException e) {

throw new ClassCastException(context.toString() + " must implement OnArticleSelectedListener");

}

}

...

}

public static class FragmentA extends ListFragment {

OnArticleSelectedListener listener;

...

@Override

public void onListItemClick(ListView l, View v, int position, long id) {

// Append the clicked item's row ID with the content provider Uri

Uri noteUri = ContentUris.withAppendedId(ArticleColumns.CONTENT\_URI, id);

// Send the event and Uri to the host activity

listener.onArticleSelected(noteUri);

}

...

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.fragment\_layout);

}

The layout applied is fragment\_layout.xml:

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

android:orientation="horizontal"

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

<fragment class="com.example.android.apis.app.FragmentLayout$TitlesFragment"

android:id="@+id/titles" android:layout\_weight="1"

android:layout\_width="0px" android:layout\_height="match\_parent" />

<FrameLayout android:id="@+id/details" android:layout\_weight="1"

android:layout\_width="0px" android:layout\_height="match\_parent"

android:background="?android:attr/detailsElementBackground" />

</LinearLayout>

Using this layout, the system instantiates the TitlesFragment (which lists the play titles) as soon as the activity loads the layout, while the FrameLayout (where the fragment for showing the play summary appears) consumes space on the right side of the screen, but remains empty at first. As you'll see below, it's not until the user selects an item from the list that a fragment is placed into the FrameLayout.

However, not all screen configurations are wide enough to show both the list of plays and the summary, side by side. So, the layout above is used only for the landscape screen configuration, by saving it at res/layout-land/fragment\_layout.xml.

Thus, when the screen is in portrait orientation, the system applies the following layout, which is saved at res/layout/fragment\_layout.xml:

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

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

<fragment class="com.example.android.apis.app.FragmentLayout$TitlesFragment"

android:id="@+id/titles"

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

</FrameLayout>

This layout includes only TitlesFragment. This means that, when the device is in portrait orientation, only the list of play titles is visible. So, when the user clicks a list item in this configuration, the application starts a new activity to show the summary, instead of loading a second fragment.

Next, you can see how this is accomplished in the fragment classes. First is TitlesFragment, which shows the list of Shakespeare play titles. This fragment extends ListFragment and relies on it to handle most of the list view work.

As you inspect this code, notice that there are two possible behaviors when the user clicks a list item: depending on which of the two layouts is active, it can either create and display a new fragment to show the details in the same activity (adding the fragment to the FrameLayout), or start a new activity (where the fragment can be shown).

KOTLIN

JAVA

public static class TitlesFragment extends ListFragment {

boolean dualPane;

int curCheckPosition = 0;

@Override

public void onActivityCreated(Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

// Populate list with our static array of titles.

setListAdapter(new ArrayAdapter<String>(getActivity(),

android.R.layout.simple\_list\_item\_activated\_1, Shakespeare.TITLES));

// Check to see if we have a frame in which to embed the details

// fragment directly in the containing UI.

View detailsFrame = getActivity().findViewById(R.id.details);

dualPane = detailsFrame != null && detailsFrame.getVisibility() == View.VISIBLE;

if (savedInstanceState != null) {

// Restore last state for checked position.

curCheckPosition = savedInstanceState.getInt("curChoice", 0);

}

if (dualPane) {

// In dual-pane mode, the list view highlights the selected item.

getListView().setChoiceMode(ListView.CHOICE\_MODE\_SINGLE);

// Make sure our UI is in the correct state.

showDetails(curCheckPosition);

}

}

@Override

public void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

outState.putInt("curChoice", curCheckPosition);

}

@Override

public void onListItemClick(ListView l, View v, int position, long id) {

showDetails(position);

}

/\*\*

\* Helper function to show the details of a selected item, either by

\* displaying a fragment in-place in the current UI, or starting a

\* whole new activity in which it is displayed.

\*/

void showDetails(int index) {

curCheckPosition = index;

if (dualPane) {

// We can display everything in-place with fragments, so update

// the list to highlight the selected item and show the data.

getListView().setItemChecked(index, true);

// Check what fragment is currently shown, replace if needed.

DetailsFragment details = (DetailsFragment)

getSupportFragmentManager().findFragmentById(R.id.details);

if (details == null || details.getShownIndex() != index) {

// Make new fragment to show this selection.

details = DetailsFragment.newInstance(index);

// Execute a transaction, replacing any existing fragment

// with this one inside the frame.

FragmentTransaction ft = getSupportFragmentManager().beginTransaction();

if (index == 0) {

ft.replace(R.id.details, details);

} else {

ft.replace(R.id.a\_item, details);

}

ft.setTransition(FragmentTransaction.TRANSIT\_FRAGMENT\_FADE);

ft.commit();

}

} else {

// Otherwise we need to launch a new activity to display

// the dialog fragment with selected text.

Intent intent = new Intent();

intent.setClass(getActivity(), DetailsActivity.class);

intent.putExtra("index", index);

startActivity(intent);

}

}

}

The second fragment, DetailsFragment shows the play summary for the item selected from the list from TitlesFragment:

KOTLIN

JAVA

public static class DetailsFragment extends Fragment {

/\*\*

\* Create a new instance of DetailsFragment, initialized to

\* show the text at 'index'.

\*/

public static DetailsFragment newInstance(int index) {

DetailsFragment f = new DetailsFragment();

// Supply index input as an argument.

Bundle args = new Bundle();

args.putInt("index", index);

f.setArguments(args);

return f;

}

public int getShownIndex() {

return getArguments().getInt("index", 0);

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

if (container == null) {

// We have different layouts, and in one of them this

// fragment's containing frame doesn't exist. The fragment

// may still be created from its saved state, but there is

// no reason to try to create its view hierarchy because it

// isn't displayed. Note this isn't needed -- we could just

// run the code below, where we would create and return the

// view hierarchy; it would just never be used.

return null;

}

ScrollView scroller = new ScrollView(getActivity());

TextView text = new TextView(getActivity());

int padding = (int)TypedValue.applyDimension(TypedValue.COMPLEX\_UNIT\_DIP,

4, getActivity().getResources().getDisplayMetrics());

text.setPadding(padding, padding, padding, padding);

scroller.addView(text);

text.setText(Shakespeare.DIALOGUE[getShownIndex()]);

return scroller;

}

}

Recall from the TitlesFragment class, that, if the user clicks a list item and the current layout does not include the R.id.details view (which is where the DetailsFragment belongs), then the application starts the DetailsActivity activity to display the content of the item.

Here is the DetailsActivity, which simply embeds the DetailsFragment to display the selected play summary when the screen is in portrait orientation:

KOTLIN

JAVA

public static class DetailsActivity extends FragmentActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

if (getResources().getConfiguration().orientation

== Configuration.ORIENTATION\_LANDSCAPE) {

// If the screen is now in landscape mode, we can show the

// dialog in-line with the list so we don't need this activity.

finish();

return;

}

if (savedInstanceState == null) {

// During initial setup, plug in the details fragment.

DetailsFragment details = new DetailsFragment();

details.setArguments(getIntent().getExtras());

getSupportFragmentManager().beginTransaction().add(android.R.id.content, details).commit();

}

}

}