Fragments

Summarized from
http://developer.android.com/training/basics/fragments/index.html

By Dr. Oren Mishali
Why Fragments?

• More flexibility in designing the UI
• Fragments are useful, e.g., in supporting a wide range of screens
• The same fragments may be reused to create different layout configurations
What Is a Fragment?

• A modular section of an activity, a “sub activity”
  • Has its own lifecycle
  • Receives its own input events
  • Can be added or removed while the activity is running
Required Setup

• We will extend the **Fragment** class using the Support Library

• Set up the project to use the **v4** library
  • Or the **v7 appcompat** library

This is the project created by Android Studio that by default includes both libraries.
How To Create a Fragment

• Extend the `Fragment` class
• Override key lifecycle methods to insert your app logic
• Define the layout within `onCreateView()`

```java
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class ArticleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.article_view, container, false);
    }
}
```

• like an activity, a fragment may implement other lifecycle callbacks
• E.g., when the activity's `onPause()` is called, any fragments in the activity also receive a call to `onPause()`
• More info about the fragment’s life cycle
Adding Fragments to an Activity Via XML

- Each instance of a Fragment class must be associated with a parent Activity

res/layout-large/news_articles.xml

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <Fragment android:name="com.example.android.fragments.HeadlinesFragment"
        android:id="@+id/headlines_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />

    <Fragment android:name="com.example.android.fragments.ArticleFragment"
        android:id="@+id/article_fragment"
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="match_parent" />

</LinearLayout>
```

```java
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class MainActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.news_articles);
    }
}
```

- If you're using the v7 appcompat library, extend `AppCompatActivity` instead
- If `min-sdk` is API 11 or higher, you can simply extend `Activity`
Adding a Fragment At Runtime

• The fragments are used differently in a small screen

• Here, the fragments are not declared in XML, but created and added at runtime

res/layout/news_articles.xml

```xml
```

The activity layout must include a container view

```java
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class MainActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.news_articles);

        // Check that the activity is using the layout version with
        // the fragment_container FrameLayout
        if (findViewById(R.id.fragment_container) != null) {
            // However, if we're being restored from a previous state,
            // then we don't need to do anything and should return or else
            // we could end up with overlapping fragments.
            if (savedInstanceState != null) {
                return;
            }

            // Create a new Fragment to be placed in the activity layout
            HeadlinesFragment firstFragment = new HeadlinesFragment();

            // In case this activity was started with special instructions from an
            // Intent, pass the Intent's extras to the fragment as arguments
            firstFragment.setArguments(getIntent().getExtras());

            // Add the fragment to the 'fragment_container' FrameLayout
            getFragmentManager().beginTransaction()
                .add(R.id.fragment_container, firstFragment).commit();
        }
    }
}
```
Replace One Fragment with Another

• To replace a fragment use `replace()` instead of `add()`
• To allow backward navigation, call `addToBackStack()` before commit

```java
// Create fragment and give it an argument specifying the article it should show
ArticleFragment newFragment = new ArticleFragment();
Bundle args = new Bundle();
args.putInt(ArticleFragment.ARG_POSITION, position);
newFragment.setArguments(args);

FragmentManager transaction = getSupportFragmentManager().beginTransaction();

// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack so the user can navigate back
transaction.replace(R.id.fragment_container, newFragment);
transaction.addBackStack(null);

// Commit the transaction
transaction.commit();
```

Note: the method `addToBackStack()` may be provided with a unique name for the transaction. The name is only needed to perform advanced operations, see here for more details.
Communication Between Fragments

• A Fragment may need to communicate with another Fragment
  • E.g., change the content based on a user event
• All Fragment-to-Fragment communication is done through the associated Activity
  • Two Fragments should never communicate directly
• See here an example for a possible communication pattern

The host activity can deliver messages to a fragment by
• capturing the Fragment instance with `findFragmentById()`,
  • and then call its public methods
Additional Reading

• Frgaments API Guide
  • Note the example at the end of the document