Communicating with Other Fragments

In order to reuse the Fragment UI components, you should build each as a completely self-contained, modular component that defines its own layout and behavior. Once you have defined these reusable Fragments, you can associate them with an Activity and connect them with the application logic to realize the overall composite UI.

Often you will want one Fragment to communicate with another, for example to 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.

THIS LESSON TEACHES YOU TO

YOU SHOULD ALSO READ

- Fragments

TRY IT OUT

Download the sample

FragmentBasics.zip

Define an Interface

To allow a Fragment to communicate up to its Activity, you can define an interface in the Fragment class and implement it within the Activity. The Fragment captures the interface implementation during its onAttach() lifecycle method and can then call the Interface methods in order to communicate with the Activity.

Here is an example of Fragment to Activity communication:

```
public class HeadlinesFragment extends ListFragment {
    OnHeadlineSelectedListener mCallback;

    // Container Activity must implement this interface
    public interface OnHeadlineSelectedListener {
        public void onArticleSelected(int position);
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);

        // This makes sure that the container activity has implemented
        // the callback interface. If not, it throws an exception
        try {
            mCallback = (OnHeadlineSelectedListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnHeadlineSelectedListener");
        }
    }

    ...
}
```

Now the fragment can deliver messages to the activity by calling the `onArticleSelected()` method (or other methods in the interface) using the `mCallback` instance of the `OnHeadlineSelectedListener` interface.

For example, the following method in the fragment is called when the user clicks on a list item. The fragment uses the callback interface to deliver the event to the parent activity.

```
    @Override
    public void onListItemClick(ListView l, View v, int position, long id) {
        // Send the event to the host activity
        mCallback.onArticleSelected(position);
    }
```

Implement the Interface

In order to receive event callbacks from the fragment, the activity that hosts it must implement the interface defined in the fragment class.

For example, the following activity implements the interface from the above example.

```
public static class MainActivity extends Activity
        implements HeadlinesFragment.OnHeadlineSelectedListener{
    ...

    public void onArticleSelected(int position) {
        // The user selected the headline of an article from the HeadlinesFragment
        // Do something here to display that article
    }
}
```

Deliver a Message to a Fragment

The host activity can deliver messages to a fragment by capturing the `Fragment` (/reference/android/support/v4/app/Fragment.html) instance with `findFragmentById()` (/reference/android/support/v4/app/FragmentManager.html#findFragmentById(int)), then directly call the fragment's public methods.

For instance, imagine that the activity shown above may contain another fragment that's used to display the item specified by the data returned in the above callback method. In this case, the activity can pass the information received in the callback method to the other fragment that will display the item:

```
public static class MainActivity extends Activity
        implements HeadlinesFragment.OnHeadlineSelectedListener{
    ...

    public void onArticleSelected(int position) {
        // The user selected the headline of an article from the HeadlinesFragment
        // Do something here to display that article

        ArticleFragment articleFrag = (ArticleFragment)
                getSupportFragmentManager().findFragmentById(R.id.article_fragment);

        if (articleFrag != null) {
            // If article frag is available, we're in two-pane layout...

            // Call a method in the ArticleFragment to update its content
            articleFrag.updateArticleView(position);
        } else {
            // Otherwise, we're in the one-pane layout and must swap frags...

            // Create fragment and give it an argument for the selected article
            ArticleFragment newFragment = new ArticleFragment();
            Bundle args = new Bundle();
            args.putInt(ArticleFragment.ARG_POSITION, position);
            newFragment.setArguments(args);
            FragmentTransaction transaction = getSupportFragmentManager().beginTransac

            // 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.addToBackStack(null);

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