Developers

SEARCH

Search

ARCHIVE

► 2013 (3)
▼ 2012 (43)
  ► December (5)
  ► November (3)
  ► October (3)
  ► September (1)
  ► August (1)
  ► July (2)
  ▼ June (5)
    Android SDK Tools, Revision 20
    Introducing Android 4.1 (Jelly Bean) preview platf...

03 JUNE 2012

Using DialogFragments

[This post is by David Chandler, Android Developer Advocate — Tim Bray]

Honeycomb introduced Fragments to support reusing portions of UI and logic across multiple activities in an app. In parallel, the showDialog / dismissDialog methods in Activity are being deprecated in favor of DialogFragments.

In this post, I'll show how to use DialogFragments with the v4 support library (for backward compatibility on pre-Honeycomb devices) to show a simple edit dialog and return a result to the calling Activity using an interface. For design guidelines around Dialogs, see the Android Design site.

The Layout

Here's the layout for the dialog in a file named `fragment_edit_name.xml`.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/edit_name"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_gravity="center" android:orientation="vertical"  >

    <TextView
        android:id="@+id/lbl_your_name" android:text="Your name"
        android:layout_width="wrap_content" android:layout_height="wrap_content" />
    <EditText
        android:id="@+id/txt_your_name"
        android:layout_width="match_parent"  android:layout_height="wrap_content"
        android:inputType=”text”
        android:imeOptions="actionDone" />
</LinearLayout>
```

Note the use of two optional attributes. In conjunction with `android:inputType=”text”`, `android:imeOptions=”actionDone”` configures the soft keyboard to show a Done key in place of the Enter key.

The Dialog Code

The dialog extends DialogFragment, and since we want backward compatibility, we'll import it from the v4 support library. (To add the support library to an Eclipse project, right-click on the project and choose Android Tools | Add Support Library...).

```
import android.support.v4.app.DialogFragment;
// ...

public class EditNameDialog extends DialogFragment {

    private EditText mEditText;

    public EditNameDialog() {
        // Empty constructor required for DialogFragment
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_edit_name, container);
        mEditText = (EditText) view.findViewById(R.id.txt_your_name);
        getDialog().setTitle("Hello");

        return view;
    }
}
```

The dialog extends DialogFragment and includes the required empty constructor. Fragments implement the `onCreateView()` method to actually load the view using the provided LayoutInflater.

Showing the Dialog

Now we need some code in our Activity to show the dialog. Here is a simple example that immediately shows the EditNameDialog to enter the user's name. On completion, it shows a Toast with the entered text.

```
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
// ...

public class FragmentDialogDemo extends FragmentActivity implements EditNameDialogListe
ner {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        showEditDialog();
    }

    private void showEditDialog() {
        FragmentManager fm = getSupportFragmentManager();
        EditNameDialog editNameDialog = new EditNameDialog();
        editNameDialog.show(fm, "fragment_edit_name");
    }

    @Override
    public void onFinishEditDialog(String inputText) {
        Toast.makeText(this, "Hi, " + inputText, Toast.LENGTH_SHORT).show();
    }
}
```

There are a few things to notice here. First, because we're using the support library for backward compatibility with the Fragment API, our Activity extends FragmentActivity from the support library. Because we're using the support library, we call `getSupportFragmentManager()` instead of `getFragmentManager()`.

After loading the initial view, the activity immediately shows the EditNameDialog by calling its show() method. This allows the DialogFragment to ensure that what is happening with the Dialog and Fragment states remains consistent. By default, the back button will dismiss the dialog without any additional code.

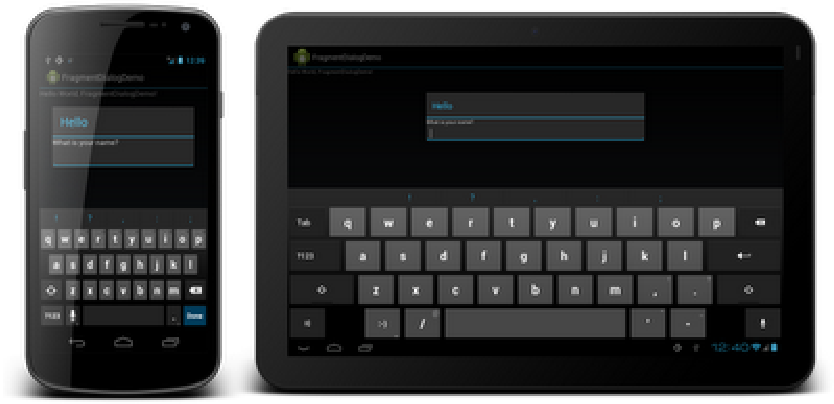
Using the Dialog

Next, let's enhance EditNameDialog so it can return a result string to the Activity.

```
import android.support.v4.app.DialogFragment;
// ...
public class EditNameDialog extends DialogFragment implements OnEditorActionListener {

    public interface EditNameDialogListener {
        void onFinishEditDialog(String inputText);
    }

    private EditText mEditText;

    public EditNameDialog() {
        // Empty constructor required for DialogFragment
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_edit_name, container);
        mEditText = (EditText) view.findViewById(R.id.txt_your_name);
        getDialog().setTitle("Hello");

        // Show soft keyboard automatically
        mEditText.requestFocus();
        getDialog().getWindow().setSoftInputMode(
                LayoutParams.SOFT_INPUT_STATE_VISIBLE);
        mEditText.setOnEditorActionListener(this);

        return view;
    }

    @Override
    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
        if (EditorInfo.IME_ACTION_DONE == actionId) {
            // Return input text to activity
            EditNameDialogListener activity = (EditNameDialogListener) getActivity();
            activity.onFinishEditDialog(mEditText.getText().toString());
            this.dismiss();
            return true;
        }
        return false;
    }
}
```

For user convenience, we programmatically focus on the EditText with `mEditText.requestFocus()`. Alternatively, we could have used the `<requestFocus/>` tag in the layout XML to do this; however, in some cases it's preferable to request focus programmatically. For example, an OnFocusChangeListener added in the Fragment's `onCreateView()` method won't get called if you request focus in the layout XML.

If the user focuses on an EditText, the soft keyboard will automatically appear. In order to force this to happen with our programmatic focus, we call `getDialog().getWindow().setSoftInputMode()`. Note that many Window operations you might have done previously in a Dialog can still be done in a DialogFragment, but you have to call `getDialog().getWindow()` instead of just `getWindow()`. The resulting dialog is shown on both a handset and tablet (not to scale):



The `onEditorAction()` method handles the callback when the user presses the Done key. It gets invoked because we've set an OnEditorActionListener on the EditText. It calls back to the Activity to send the entered text. To do this, EditNameDialog declares an interface EditNameDialogListener that is implemented by the Activity. This enables the dialog to be reused by many Activities. To invoke the callback method `onFinishEditDialog()`, it obtains a reference to the Activity which launched the dialog by calling getActivity(), which all Fragments provide, and then casts it to the interface type. In MVC architecture, this is a common pattern for allowing a view to communicate with a controller.

We can dismiss the dialog one of two ways. Here we are calling dismiss() within the Dialog class itself. It could also be called from the Activity like the show() method.

Hopefully this sheds some more light on Fragments as they relate to Dialogs. You can find the sample code in this blog post on Google Code.

References for learning more about Fragments:

- Fragments Dev Guide
- “Basic Training” on Fragments
- Updating Applications for On-screen Input Methods
- OnEditorActionListener
- EditorInfo and IME options

Posted by Tim Bray at 12:06 PM
Labels: App Components

Links to this post

Create a Link

Newer Post Home Older Post