Android

Ben Gavan

July 16, 2019

Contents

1 Dialogs

1.1 Setting a Target Fragment

When displaying a dialog view from a fragment, we need to create a relationship between them se we can send data back from the dialog to the fragment.
We need to pass a reference to the dialog of the fragment, as well as a request code to identify the payload when it is sent back/ so the fragment can 'listen' out for it.
We do this by setting the target fragment on the dialog object:

```
1 dialog.setTargetFragment(FragmentClass.this, REQUEST_CODE);
```

1.2 Sending data back to the Target Fragment from the Dialog

We should also check that the target fragment has been set before we do anything
First, we need to get a reference to the target fragment (set by the fragment requesting the display of the dialog via using setTargetFragment on the dialog). We then call 'onActivityResult' on the target fragment.
So if we want to do something in the fragment i.e. get the data back, we have to override this method in the target fragment.
The data we pass back from the dialog is contained within an intent by putExtra.

```
1  private void sendResult(int resultCode, Date date) {
2    if (getTargetFragment() == null) {
3      return;
4    }
5
6    Intent intent = new Intent();
7    intent.putExtra(EXTRA_DATE, date);
8
9    this.getTargetFragment().onActivityResult(this.
         getTargetRequestCode(), resultCode, intent);
10 }
```

1.2.1 Receiving the Data from the intent

```
1  @Override
2  public void onActivityResult(int requestCode, int resultCode,
       Intent data) {
3    if (resultCode != Activity.RESULT_OK) {
4      return;
5    }
6
7    if (requestCode == REQUEST_DATE) {
8      Date date = (Date) data.getSerializableExtra(DatePickerFragment
           .EXTRA_DATE);
9      mCrime.setDate(date);
10     mDateButton.setText(mCrime.getDate().toString());
11   }
12 }
```

We override the 'onActivityResult' within the target fragment we are sending data back to.
First we check what the result code is (what button the user pressed on the dialog)

```
1  if (resultCode != Activity.RESULT_OK) {
2    return;
3  }
```

We then check what the request code is (which was set by the fragment creating the dialog) so we know that we are responding to the correct result (A fragment can display and react to multiple dialogs).

After this, we get the data sent back in the form of an extra from the dialog inside an intent by 'getSerializableExtra(...)'.
In this case, we cast this data back to a date so it can be used.

```
1 if (requestCode == REQUEST_DATE) {
2   Date date = (Date) data.getSerializableExtra(DatePickerFragment.
        EXTRA_DATE);
3   mCrime.setDate(date);
4   mDateButton.setText(mCrime.getDate().toString());
5 }
```

2 The Toolbar

The Toolbar provides additional mechanisms for navigation, nd also provides design consistency and branding.

2.0.1 History

The toolbar component was added to android 5.0 (Lollipop).
Prior to this, the action bar was the recommended component for navigation and actions within an app.
The toolbar and action bar are very similar.
The toolbar builds on top of the action bar .
It has a tweaked UI
It's more flexible in the ways you can use it.

2.0.2 Supported by

Since the toolbar has been added to the AppCompat library, it is available back to API 9 (Android 2.3)

2.1 Menus

The top-right portion of the toolbar is reserved for the toolbar's menu.
The menu consists of action items (sometimes referred to as menu items).
These can perform an action on the current screen or on the app as a whole.

2.1.1 Defining a menu in XML

Need to create an XML description of a menu, just like how you have to for layouts, with the resource file inside the res/menu directory.
To create a new menu resource file:

1. Right-click on the res directory
2. Select New → Android resource file
3. Change the Resource type to Menu
4. Name the resource (normally 'fragment_...' - the same naming convention as layout files)

5. Click OK

In this file, the XML should be:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
</menu>
```

2.1.2 Defining an item

```
<item
  android:id="@+id/new_crime"
  android:icon="@android:drawable/ic_menu_add"
  android:title="@string/new_crime"
  app:showAsAction="ifRoom|withText"/>
```

The line

```
app:showAsAction="ifRoom|withText"
```

makes the item be displayed inline/on the toolbar (where the menu icon should be) instead of having the item as a drop down item below the toolbar/menu button.
The showAsAction attribute refers to whether the item will appear in the toolbar itself or in the overflow menu.

In this case "ifRoom|withText" will make the items icon and text appear in the toolbar if there is room.
If there is room for the icon but not the text, then only the icon will be visible.
If there is no room for either, the item will be relegated to the overflow menu.

If there are items in the overflow menu, the three dots will appear and when these are pressed, the overflow menu will be shown below.

Multiple menu items can be displayed as Actions on the Toolbar.

Possible values for showAsAction

- always

  not recommended

  Better to use ifRoom and let the OS decide.

- ifRoom

  Only displays the item as an Action if there is room

- never

  never displayed as an action

will always appear in the overflow menu

so good for items that are not used very often - its good practice to avoid having to many items on the toolbar to help the screen keep decluttered

The AppCompat library defines its own custom showAsAction attribute and does not look for the native showAsAction attribute.

3 AppCompat Library

3.1 Requirements

The AppCompat requires that you:

- add the AppCompat dependency
- use one of the AppCompat themes
- ensure that all activities are a subclass of AppCompatActivity

4 Refactoring Techniques and Tools

There are many techniques and tools that can be used to make refactoring code easier.

4.1 Extracting a method with Android Studio

1. Highlight the code that you want to be extracted
2. Right-click and select Refactor → Extract → Method
3. Set the Visibility and method Name
4. Press Refractor or Preview to preview the changes
5. If there are multiple occurrences of the highlighted text being extracted, android studio will ask if you want to replace these as well

   You can either replace each occurrence one by one, or by choosing all.