Android Developer Fundamentals

User Interaction and Intuitive Navigation

Lesson 4

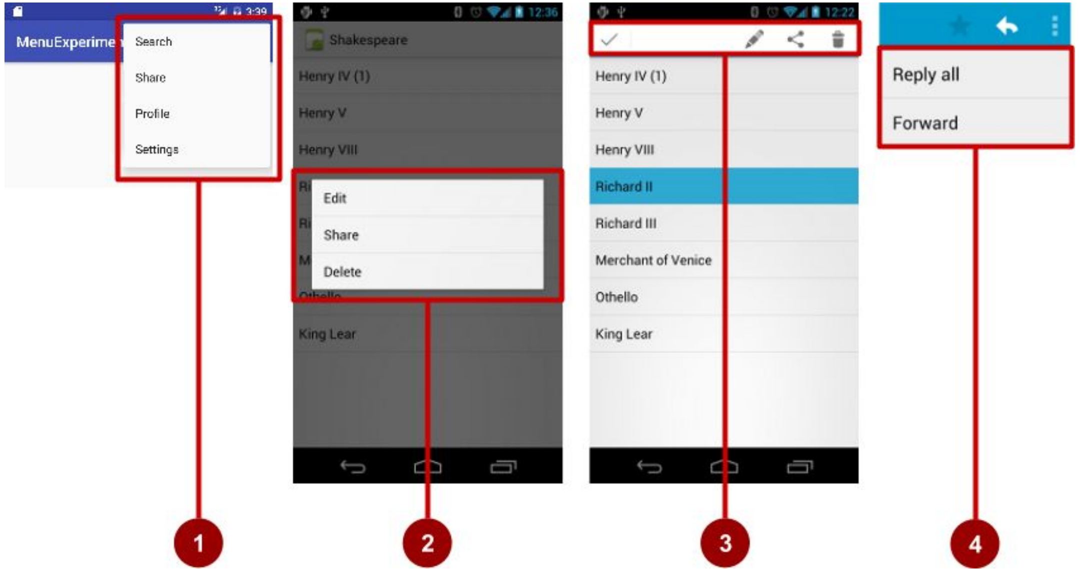
4.2 Menus

Contents

- App Bar with Options Menu
- Contextual menus
- Popup menus

Types of Menus

1. App bar with options menu
2. Contextual menu
3. Contextual action bar
4. Popup menu

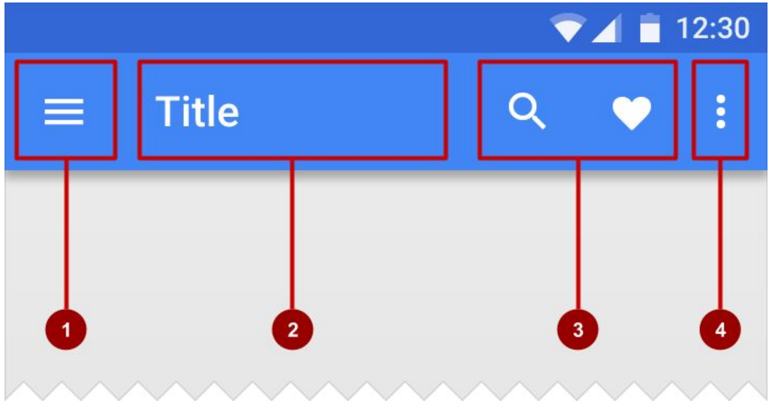


App Bar with Options Menu

What is the App Bar?

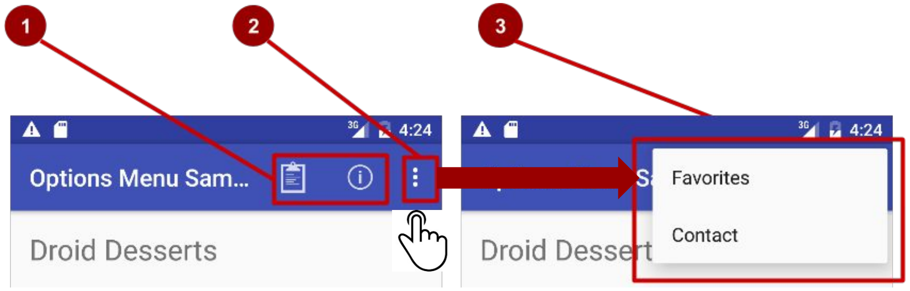
Bar at top of each screen—(usually) the same for all screens

1. Nav icon to open navigation drawer
2. Title of current activity
3. Icons for options menu items
4. Action overflow button for the rest of the options menu



What is the options menu?

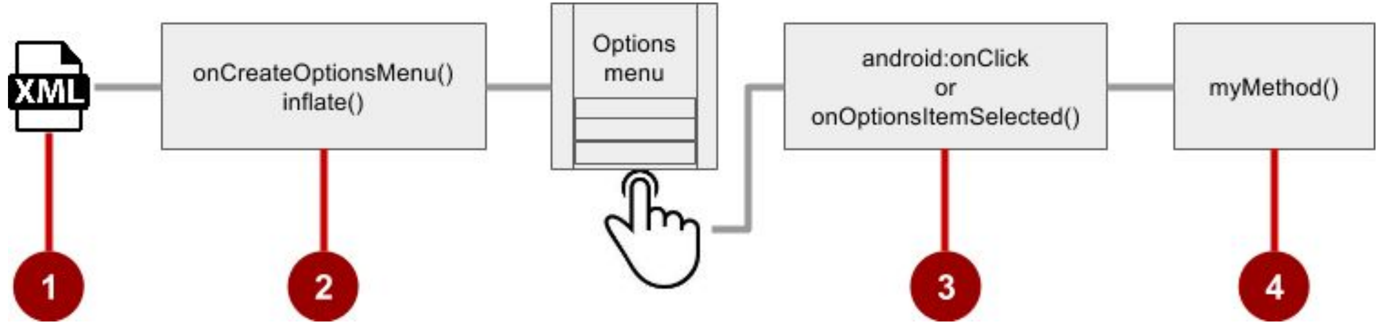
- Action icons in the app bar for important items (1)
- Tap the three dots, the "action overflow button" to see the options menu (2)
- Appears in the right corner of the app bar (3)
- For navigating to other activities and editing app settings



Adding Options Menu

Steps to implement options menu

1. XML menu resource (`menu_main.xml`)
2. `onCreateOptionsMenu()` to inflate the menu
3. `onClick` attribute or `onOptionsItemSelected()`
4. Method to handle item click



Create menu resource

1. Create menu resource directory
2. Create XML menu resource (`menu_main.xml`)
3. Add an entry for each menu item

```
<item  android:id="@+id/option_settings"
       android:title="@string/settings" />

<item  android:id="@+id/option_toast"
       android:title="@string/toast" />
```

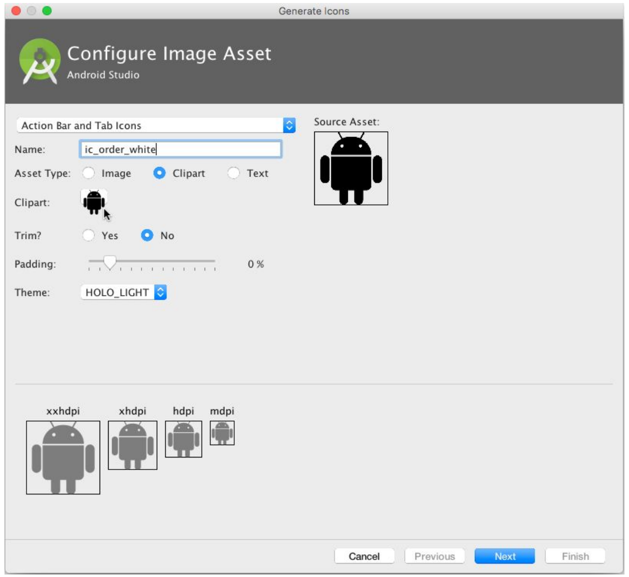
Inflate options menu

- Override onCreateOptionsMenu() in main activity

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   getMenuInflater().inflate(R.menu.menu_main, menu);
   return true;
}
```

Add icons for menu items

1. Right-click **drawable**
2. Choose **New > Image Asset**
3. Choose **Action Bar and Tab Items**
4. Edit the icon name
5. Click clipart image, and click icon
6. Click **Next**, then **Finish**



Action Bar and Tab Icons
Name: ic_order_white
Asset Type: Image [✓ Clipart] Text
Clipart:
Trim? Yes [✓ No]
Padding: 0 %
Theme: HOLO_LIGHT

Add menu item attributes

```
<item android:id="@+id/action_order"
      android:icon="@drawable/ic_toast_dark"
      android:title="@string/toast"
      android:titleCondensed="@string/toast_condensed"
      android:orderInCategory="1"
      app:showAsAction="ifRoom" />
```

Override onOptionsItemSelected()

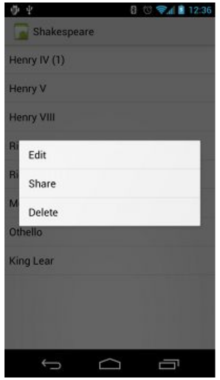
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId()) {
      case R.id.action_order:
         showOrder();
         return true;
      default:
         return super.onOptionsItemSelected(item);
    }
}
```

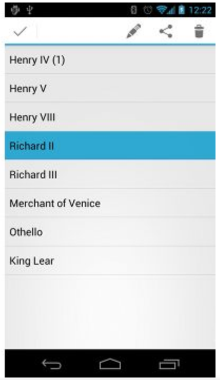
Contextual Menus

What are contextual menus?

- Allow users to perform an action on a selected view or content
- Can be deployed on any View object, but most often used for items in a RecyclerView, GridView, or other view collection

Types of contextual menus

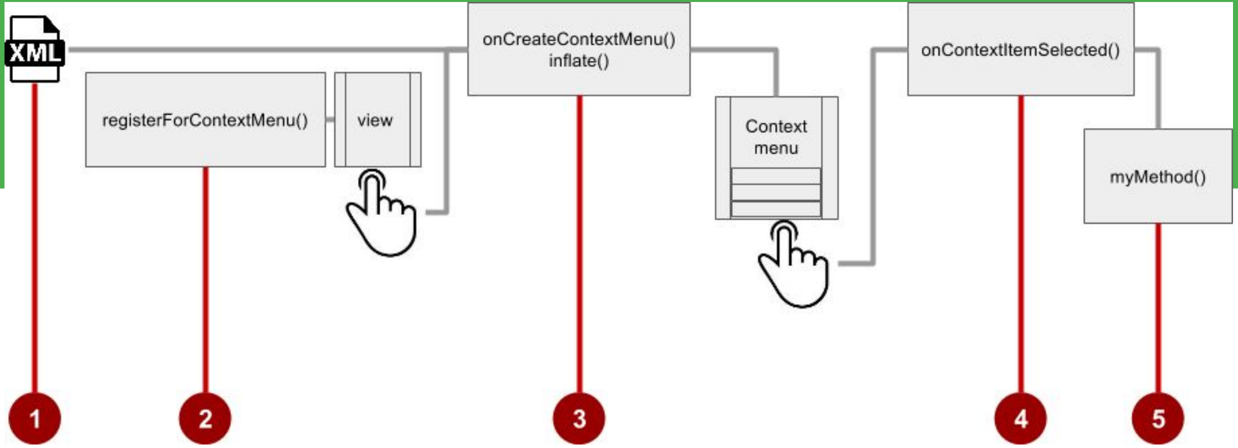




- Floating context menu—floating list of menu items when long-presses on a view element
  - User can modify the view element or use it in some fashion
  - Users perform a contextual action on one view element at a time
- Contextual action mode—temporary action bar in place of or underneath the app bar
  - Action items affect the selected view element(s)
  - Users can perform action on multiple view elements at once

Floating Context Menu

Steps



1. Create XML menu resource file and assign appearance and position attributes
2. Register view to use a context menu using `registerForContextMenu()`
3. Implement `onCreateContextMenu()` in the activity or fragment to inflate the menu
4. Implement `onContextItemSelected()` to handle menu item clicks
5. Create a method to perform an action for each context menu item

Create menu resource

- Create XML menu resource (`menu_context.xml`)

```
<item
   android:id="@+id/context_edit"
   android:title="@string/edit"
   android:orderInCategory="10"/>

<item
   android:id="@+id/context_share"
   android:title="@string/share"
   android:orderInCategory="20"/>
```

Register a view to a context menu

- in onCreate() of the activity
- registers View.OnCreateContextMenuListener
- Does not specify which context menu!

```
TextView article_text = (TextView) findViewById(R.id.article);
registerForContextMenu(article_text);
```

Implement onCreateContextMenu()

Specifies which context menu

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                         ContextMenu.ContextMenuInfo menuInfo) {
   super.onCreateContextMenu(menu, v, menuInfo);
   MenuInflater inflater = getMenuInflater();
   inflater.inflate(R.menu.menu_context, menu);
}
```

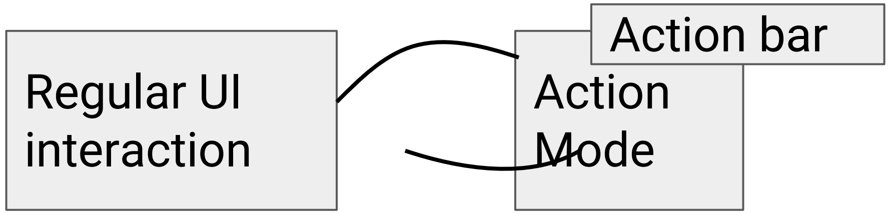
Implement onContextItemSelected()

```
@Override
public boolean onContextItemSelected(MenuItem item) {
   switch (item.getItemId()) {
      case R.id.context_edit:
         editNote();
         return true;
      default:
         return super.onContextItemSelected(item);
     }
}
```

Contextual Action Bar

What is Action Mode?

- ActionMode is a UI mode that lets you replace parts of the normal UI interactions temporarily
- For example, selecting a section of text or long-pressing an item could trigger action mode



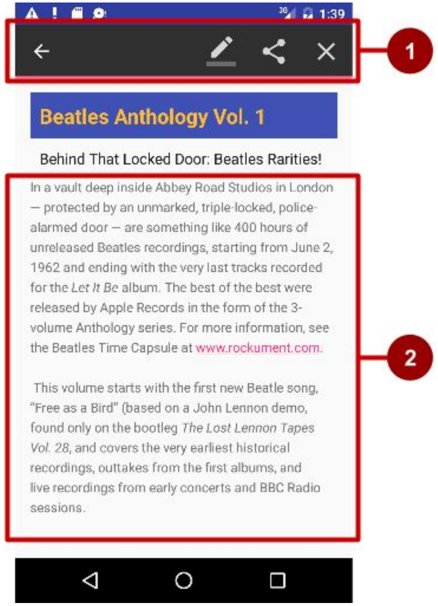
Action mode has a lifecycle

- Start it with startActionMode(), for example, in the listener
- ActionMode.Callback interface provides the lifecycle methods that you can override
  - `onCreateActionMode(ActionMode, Menu)` once on initial creation
  - `onPrepareActionMode(ActionMode, Menu)` after creation and any time ActionMode is invalidated
  - `onActionItemClicked(ActionMode, MenuItem)` any time a contextual action button is clicked
  - `onDestroyActionMode(ActionMode)` when the action mode is closed

What is a contextual action bar?

Long-tap on the view shows contextual action bar

1. Contextual action bar with actions
   - Edit, Share, and Delete
   - Done (left arrow icon) on the left side
2. View on which long press triggers the contextual action bar
   - Action bar is available until user taps Done



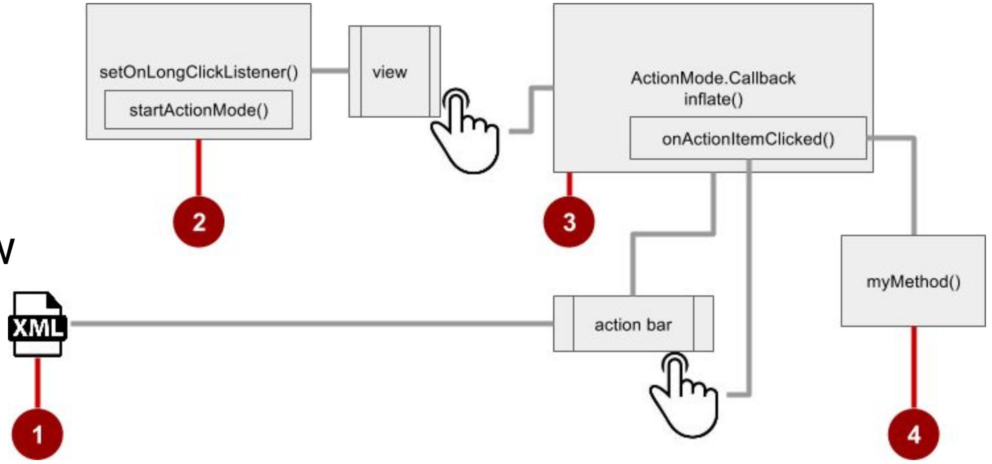
Beatles Anthology Vol. 1

Behind That Locked Door: Beatles Rarities!

In a vault deep inside Abbey Road Studios in London — protected by an unmarked, triple-locked, police-alarmed door — are something like 400 hours of unreleased Beatles recordings, starting from June 2, 1962 and ending with the very last tracks recorded for the *Let It Be* album. The best of the best were released by Apple Records in the form of the 3-volume Anthology series. For more information, see the Beatles Time Capsule at www.rockument.com.

This volume starts with the first new Beatle song, "Free as a Bird" (based on a John Lennon demo, found only on the bootleg *The Lost Lennon Tapes Vol. 28*, and covers the very earliest historical recordings, outtakes from the first albums, and live recordings from early concerts and BBC Radio sessions.

Steps for contextual action bar



1. Create XML menu resource file and assign icons for items
2. `setOnLongClickListener()` on view that triggers the contextual action bar and call `startActionMode()` to handle click
3. Implement `ActionMode.Callback` interface to handle ActionMode lifecycle; include action for a menu item click in `onActionItemClicked()` callback
4. Create a method to perform an action for each context menu item

Use setOnLongClickListener

```
private ActionMode mActionMode;
```

In onCreate

```
   View view = findViewById(article);
   view.setOnLongClickListener(new View.OnLongClickListener() {
      public boolean onLongClick(View view) {
         if (mActionMode != null) return false;
         mActionMode =
               MainActivity.this.startActionMode(mActionModeCallback);
         view.setSelected(true);
         return true;
      }
   });
```

Implement mActionModeCallback

```
public ActionMode.Callback mActionModeCallback =
   new ActionMode.Callback() {
    // Implement action mode callbacks here
};
```

Implement onCreateActionMode

```
@Override
public boolean onCreateActionMode(ActionMode mode, Menu menu) {
   MenuInflater inflater = mode.getMenuInflater();
   inflater.inflate(R.menu.menu_context, menu);
   return true;
}
```

Implement onPrepareActionMode

- Called each time the action mode is shown
- Always called after `onCreateActionMode`, but may be called multiple times if the mode is invalidated

```
@Override
public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
   return false; // Return false if nothing is done.
}
```

Implement onActionItemClicked

- Called when users selects an action
- Handle clicks in this method

```
@Override
public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
   switch (item.getItemId()) {
       case R.id.goodbyetextview:
         Toast.makeText(getApplicationContext(), "Menu Toast", Toast.LENGTH_SHORT).show();
         mode.finish(); // Action picked, so close the action bar
         return true;
       default:
         return false;
   }
}
```

Implement onDestroyActionMode

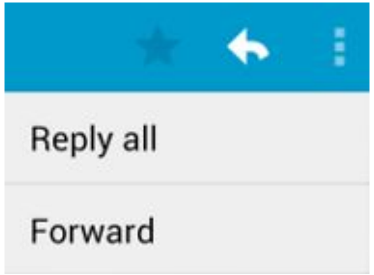
- Called when user exits the action mode

```
@Override
public void onDestroyActionMode(ActionMode mode) {
   mActionMode = null;
}
```

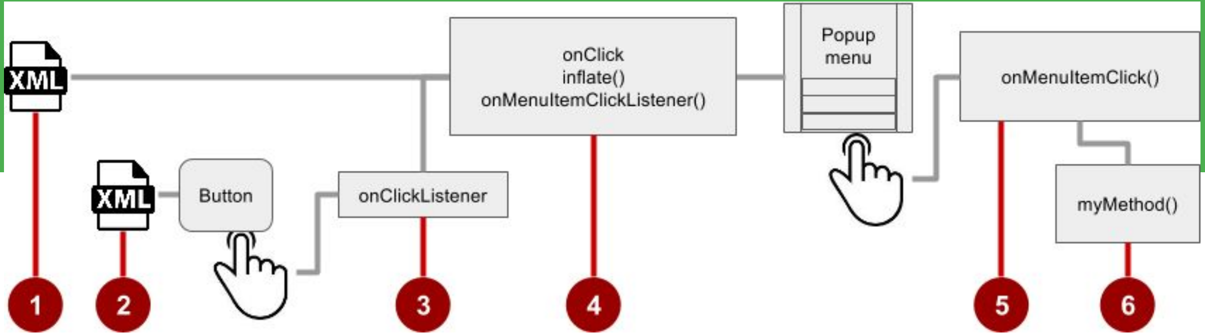
Popup Menu

What is a popup menu?

- Vertical list of items anchored to a view
- Typically anchored to a visible icon
- Actions should not directly affect view content
    - The options menu overflow that opens Settings
    - For example, in an email app, Reply All and Forward are related to the email message, but don't affect or act on the message



Steps



1. Create XML menu resource file and assign appearance and position attributes
2. Add an `ImageButton` for the popup menu icon in the XML activity layout file
3. Assign `onClickListener` to the button
4. Override `onClick()` to inflate the popup and register it with `onMenuItemClickListener()`
5. Implement `onMenuItemClick()`
6. Create a method to perform an action for each popup menu item

Add an ImageButton

```
<ImageButton
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/button_popup"
 android:src="@drawable/@drawable/ic_action_popup"/>
```

Assign onClickListener to button

```
private ImageButton mButton =
    (ImageButton) findViewById(R.id.button_popup);
```

In onCreate:

```
mButton.setOnClickListener(new View.OnClickListener() {
    // define onClick
});
```

Implement onClick

```
@Override
public void onClick(View v) {
    PopupMenu popup = new PopupMenu(MainActivity.this, mButton);
    popup.getMenuInflater().inflate(
        R.menu.menu_popup, popup.getMenu());
    popup.setOnMenuItemClickListener(
        new PopupMenu.OnMenuItemClickListener() {
            // implement click listener
        });
    popup.show();
}
```

Implement onMenuItemClick

```
public boolean onMenuItemClick(MenuItem item) {
   switch (item.getItemId()) {
       case R.id.option_toast:
           Toast.makeText(getApplicationContext(), "Popup Toast",
                   Toast.LENGTH_SHORT).show();
           return true;
       default:
           return false;
   }
}
```

Learn more

- Adding the App Bar
- Styles and Themes
- Menus
- Menu Resource

What's Next?

- Concept Chapter: 4.2 C Menus
- Practical: 4.2 P Using an Options Menu

END