Setting Up the App Bar

This lesson teaches you to

- Add a Toolbar to an Activity
- Use App Bar Utility Methods

You should also read

- Setting Up the Support Library

In its most basic form, the action bar displays the title for the activity on one side and an *overflow menu* on the other. Even in this simple form, the app bar provides useful information to the users, and helps to give Android apps a consistent look and feel.

Figure 1. An app bar with the app title and overflow menu.

Beginning with Android 3.0 (API level 11), all activities that use the default theme have an `ActionBar` (`https://developer.android.com/reference/android/app/ActionBar.html`) as an app bar. However, app bar features have gradually been added to the native `ActionBar` (`https://developer.android.com/reference/android/app/ActionBar.html`) over various Android releases. As a result, the native `ActionBar` (`https://developer.android.com/reference/android/app/ActionBar.html`) behaves differently depending on what version of the Android system a device may be using. By contrast, the most recent features are added to the support library's version of `Toolbar` (`https://developer.android.com/reference/android/support/v7/widget/Toolbar.html`), and they are available on any device that can use the support library.

For this reason, you should use the support library's `Toolbar` (`https://developer.android.com/reference/android/support/v7/widget/Toolbar.html`) class to implement your activities' app bars. Using the support library's toolbar helps ensure that your app will have consistent behavior across the widest range of devices. For example, the `Toolbar` (`https://developer.android.com/reference/android/support/v7/widget/Toolbar.html`) widget provides a material design

(https://developer.android.com/design/material/index.html) experience on devices running Android 2.1 (API level 7) or later, but the native action bar doesn't support material design unless the device is running Android 5.0 (API level 21) or later.

Add a Toolbar to an Activity

These steps describe how to set up a `Toolbar` (https://developer.android.com/reference/android/support/v7/widget/Toolbar.html) as your activity's app bar:

1. Add the v7 appcompat (https://developer.android.com/tools/support-library/features.html#v7-appcompat) support library to your project, as described in Support Library Setup (https://developer.android.com/tools/support-library/setup.html).

2. Make sure the activity extends `AppCompatActivity` (https://developer.android.com/reference/android/support/v7/app/AppCompatActivity.html):

   ```
   public class MyActivity extends AppCompatActivity {
     // ...
   }
   ```

   > **Note:** Make this change for every activity in your app that uses a `Toolbar` (https://developer.android.com/reference/android/support/v7/widget/Toolbar.html) as an app bar.

3. In the app manifest, set the <application> (https://developer.android.com/guide/topics/manifest/application-element.html) element to use one of appcompat's `NoActionBar` (https://developer.android.com/reference/android/support/v7/appcompat/R.style.html#Theme_AppCompat_NoActionBar) themes. Using one of these themes prevents the app from using the native `ActionBar` (https://developer.android.com/reference/android/app/ActionBar.html) class to provide the app bar. For example:

   ```
   <application
       android:theme="@style/Theme.AppCompat.Light.NoActionBar"
       />
   ```

4. Add a `Toolbar` (https://developer.android.com/reference/android/support/v7/widget/Toolbar.html) to the activity's layout. For example, the following layout code adds a `Toolbar` (https://developer.android.com/reference/android/support/v7/widget/Toolbar.html) and gives it the appearance of floating above the activity:

   ```
   <android.support.v7.widget.Toolbar
      android:id="@+id/my_toolbar"
      android:layout_width="match_parent"
      android:layout_height="?attr/actionBarSize"
      android:background="?attr/colorPrimary"
      android:elevation="4dp"
      android:theme="@style/ThemeOverlay.AppCompat.ActionBar"
      app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>
   ```

   The Material Design specification (https://www.google.com/design/spec/what-is-material/elevation-shadows.html#elevation-shadows-shadows) recommends that app bars have an elevation of 4 dp.

Position the toolbar at the top of the activity's layout (https://developer.android.com/guide/topics/ui/declaring-layout.html), since you are using it as an app bar.

5. In the activity's `onCreate()` (https://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) method, call the activity's `setSupportActionBar()` (https://developer.android.com/reference/android/support/v7/app/AppCompatActivity.html#setSupportActionBar(android.support.v7.widget.Toolbar)) method, and pass the activity's toolbar. This method sets the toolbar as the app bar for the activity. For example:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my);
    Toolbar myToolbar = (Toolbar) findViewById(R.id.my_toolbar);
    setSupportActionBar(myToolbar);
    }
```

Your app now has a basic action bar. By default, the action bar contains just the name of the app and an overflow menu. The options menu initially contains just the **Settings** item. You can add more actions to the action bar and the overflow menu, as described in Adding and Handling Actions (https://developer.android.com/training/appbar/actions.html).

Use App Bar Utility Methods

Once you set the toolbar as an activity's app bar, you have access to the various utility methods provided by the v7 appcompat (https://developer.android.com/tools/support-library/features.html#v7-appcompat) support library's `ActionBar` (https://developer.android.com/reference/android/support/v7/app/ActionBar.html) class. This approach lets you do a number of useful things, like hide and show the app bar.

To use the `ActionBar` (https://developer.android.com/reference/android/support/v7/app/ActionBar.html) utility methods, call the activity's `getSupportActionBar()` (https://developer.android.com/reference/android/support/v7/app/AppCompatActivity.html#getSupportActionBar()) method. This method returns a reference to an appcompat `ActionBar` (https://developer.android.com/reference/android/support/v7/app/ActionBar.html) object. Once you have that reference, you can call any of the `ActionBar` (https://developer.android.com/reference/android/support/v7/app/ActionBar.html) methods to adjust the app bar. For example, to hide the app bar, call `ActionBar.hide()` (https://developer.android.com/reference/android/support/v7/app/ActionBar.html#hide()).