菜单

本文内容

关键类

- `Menu`
- `MenuItem`
- `ContextMenu`
- `ActionMode`

另请参阅

- 添加应用栏
- 菜单资源
- 从此告别菜单按钮

菜单是许多应用类型中常见的用户界面组件。要提供熟悉而一致的用户体验，您应使用 `Menu` API 呈现 Activity 中的用户操作和其他选项。

从 Android 3.0（API 级别 11）开始，采用 Android 技术的设备不必再提供一个专用“*菜单*”按钮。随着这种改变，Android 应用需摆脱对包含 6 个项目的传统菜单面板的依赖，取而代之的是要提供一个应用栏来呈现常见的用户操作。

尽管某些菜单项的设计和用户体验已发生改变，但定义一系列操作和选项所使用的语义仍是以 `Menu` API 为基础。本指南将介绍所有 Android 版本系统中三种基本菜单或操作呈现效果的创建方法：

选项菜单和应用栏

> 选项菜单是某个 Activity 的主菜单项， 供您放置对应用产生全局影响的操作，如“搜索”、“撰写电子邮件”和“设置”。
>
> 请参阅创建选项菜单部分。

上下文菜单和上下文操作模式

> 上下文菜单是用户长按某一元素时出现的浮动菜单。 它提供的操作将影响所选内容或上下文框架。
>
> 上下文操作模式在屏幕顶部栏显示影响所选内容的操作项目，并允许用户选择多项。
>
> 请参阅创建上下文菜单部分。

弹出菜单

弹出菜单将以垂直列表形式显示一系列项目，这些项目将锚定到调用该菜单的视图中。 它特别适用于提供与特定内容相关的大量操作，或者为命令的另一部分提供选项。 弹出菜单中的操作**不会**直接影响对应的内容，而上下文操作则会影响。 相反，弹出菜单适用于与您 Activity 中的内容区域相关的扩展操作。

请参阅创建弹出菜单部分。

使用 XML 定义菜单

对于所有菜单类型，Android 提供了标准的 XML 格式来定义菜单项。您应在 XML 菜单资源中定义菜单及其所有项，而不是在 Activity 的代码中构建菜单。定义后，您可以在 Activity 或片段中扩充菜单资源（将其作为 `Menu` 对象加载）。

使用菜单资源是一种很好的做法，原因如下：

- 更易于使用 XML 可视化菜单结构
- 将菜单内容与应用的行为代码分离
- 允许您利用应用资源框架，为不同的平台版本、屏幕尺寸和其他配置创建备用菜单配置

要定义菜单，请在项目 `res/menu/` 目录内创建一个 XML 文件，并使用以下元素构建菜单：

`<menu>`

定义 `Menu`，即菜单项的容器。`<menu>` 元素必须是该文件的根节点，并且能够包含一个或多个 `<item>` 和 `<group>` 元素。

`<item>`

创建 `MenuItem`，此元素表示菜单中的一项，可能包含嵌套的 `<menu>` 元素，以便创建子菜单。

`<group>`

`<item>` 元素的不可见容器（可选）。它支持您对菜单项进行分类，使其共享活动状态和可见性等属性。 如需了解详细信息，请参阅创建菜单组部分。

以下是名为 `game_menu.xml` 的菜单示例：

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game"
          android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```

`<item>` 元素支持多个属性，您可使用这些属性定义项目的外观和行为。上述菜单中的项目包括以下属性：

`android:id`

项目特有的资源 ID，让应用能够在用户选择项目时识别该项目。

`android:icon`

引用一个要用作项目图标的可绘制对象。

`android:title`

引用一个要用作项目标题的字符串。

`android:showAsAction`

指定此项应作为操作项目显示在应用栏中的时间和方式。

这些是您应使用的最重要属性，但还有许多其他属性。有关所有受支持属性的信息，请参阅菜单资源文档。

您可以通过以 `<item>` 子元素的形式添加 `<menu>` 元素，向任何菜单（子菜单除外）中的某个菜单项添加子菜单。当应用具有大量可按主题进行组织的功能时，类似于 PC 应用程序菜单栏中的菜单项（“文件”、“编辑”、“视图”等），子菜单非常有用。 例如：

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/file"
          android:title="@string/file" >
        <!-- "file" submenu -->
        <menu>
            <item android:id="@+id/create_new"
                  android:title="@string/create_new" />
            <item android:id="@+id/open"
                  android:title="@string/open" />
        </menu>
    </item>
</menu>
```

要在 Activity 中使用菜单，您需要使用 `MenuInflater.inflate()` 扩充菜单资源（将 XML 资源转换为可编程对象）。在下文中，您将了解如何扩充每种类型的菜单。

创建选项菜单

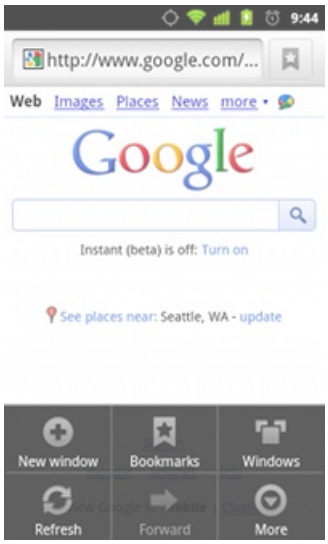

图 1. Android 2.3 系统上浏览器中的选项菜单。

在选项菜单中，您应当包括与当前 Activity 上下文相关的操作和其他选项，如“搜索”、“撰写电子邮件”和“设置”。

选项菜单中的项目在屏幕上的显示位置取决于您开发的应用所适用的 Android 版本：

- 如果您开发的应用适用于 **Android 2.3.x（API 级别 10）或更低版本**，则当用户按*“菜单”*按钮时，选项菜单的内容会出现在屏幕底部，如图 1 所示。打开时，第一个可见部分是图标菜单，其中包含多达 6 个菜单项。 如果菜单包括 6 个以上项目，则 Android 会将第六项和其余项目放入溢出菜单。用户可以通过选择*“更多”*打开该菜单。
- 如果您开发的应用适用于 **Android 3.0（API 级别 11）及更高版本**，则选项菜单中的项目将出现在应用栏中。 默认情况下，系统会将所有项目均放入操作溢出菜单中。用户可以使用应用栏右侧的操作溢出菜单图标（或者，通过按设备*“菜单”*按钮（如有））显示操作溢出菜单。要支持快速访问重要操作，您可以将 `android:showAsAction="ifRoom"` 添加到对应的 `<item>` 元素，从而将几个项目提升到应用栏中（请参阅图 2）。

  如需了解有关操作项目和其他应用栏行为的详细信息，请参阅添加应用栏培训课程。

图 2. Honeycomb Gallery 应用中的应用栏，其中显示了导航标签和相机操作项目（以及操作溢出菜单按钮）。

您可以通过 `Activity` 子类或 `Fragment` 子类为选项菜单声明项目。如果您的 Activity 和片段均为选项菜单声明项目，则这些项目将合并到 UI 中。 系统将首先显示 Activity 的项目，随后按每个片段添加到 Activity 中的顺序显示各片段的项目。 如有必要，您可以使用 `android:orderInCategory` 属性，对需要移动的每个 `<item>` 中的菜单项重新排序。

要为 Activity 指定选项菜单，请重写 `onCreateOptionsMenu()`（片段会提供自己的 `onCreateOptionsMenu()` 回调）。通过此方法，您可以将菜单资源（使用 XML 定义）扩充到回调中提供的 `Menu` 中。 例如：

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

此外，您还可以使用 `add()` 添加菜单项，并使用 `findItem()` 检索项目，以便使用 `MenuItem` API 修改其属性。

如果您开发的应用适用于 Android 2.3.x 及更低版本，则当用户首次打开选项菜单时，系统会调用 `onCreateOptionsMenu()` 来创建该菜单。如果您开发的应用适用于 Android 3.0 及更高版本，则系统将在启动 Activity 时调用 `onCreateOptionsMenu()`，以便向应用栏显示项目。

处理点击事件

用户从选项菜单中选择项目（包括应用栏中的操作项目）时，系统将调用 Activity 的 `onOptionsItemSelected()` 方法。 此方法将传递所选的 `MenuItem`。您可以通过调用 `getItemId()` 方法来识别项目，该方法将返回菜单项的唯一 ID（由菜单资源中的 `android:id` 属性定义，或通过提供给 `add()` 方法的整型数定义）。 您可以将此 ID 与已知的菜单项匹配，以执行适当的操作。例如：

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    switch (item.getItemId()) {
        case R.id.new_game:
            newGame();
            return true;
        case R.id.help:
            showHelp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

成功处理菜单项后，系统将返回 `true`。如果未处理菜单项，则应调用 `onOptionsItemSelected()` 的超类实现（默认实现将返回 false）。

如果 Activity 包括片段，则系统将依次为 Activity 和每个片段（按照每个片段的添加顺序）调用 `onOptionsItemSelected()`，直到有一个返回结果为 `true` 或所有片段均调用完毕为止。

> **提示：**Android 3.0 新增了一项功能，支持您在 XML 中使用 `android:onClick` 属性为菜单项定义点击行为。该属性的值必须是 Activity 使用菜单定义的方法的名称。 该方法必须是公用的，且接受单个 `MenuItem` 参数；当系统调用此方法时，它会传递所选的菜单项。如需了解详细信息和示例，请参阅菜单资源文档。

> **提示：**如果应用包含多个 Activity，且其中某些 Activity 提供相同的选项菜单，则可考虑创建一个仅实现 `onCreateOptionsMenu()` 和

> `onOptionsItemSelected()` 方法的 Activity。然后，为每个应共享相同选项菜单的 Activity 扩展此类。 通过这种方式，您可以管理一个用于处理菜单操作的代码集，且每个子级类均会继承菜单行为。若要将菜单项添加到一个子级 Activity，请重写该 Activity 中的 `onCreateOptionsMenu()`。 调用 `super.onCreateOptionsMenu(menu)`，以便创建原始菜单项，然后使用 `menu.add()` 添加新菜单项。此外，您还可以替代各个菜单项的超类行为。

在运行时更改菜单项

系统调用 `onCreateOptionsMenu()` 后，将保留您填充的 `Menu` 实例。除非菜单由于某些原因而失效，否则不会再次调用 `onCreateOptionsMenu()`。但是，您仅应使用 `onCreateOptionsMenu()` 来创建初始菜单状态，而不应使用它在 Activity 生命周期中执行任何更改。

如需根据在 Activity 生命周期中发生的事件修改选项菜单，则可通过 `onPrepareOptionsMenu()` 方法执行此操作。此方法向您传递 `Menu` 对象（因为该对象目前存在），以便您能够对其进行修改，如添加、移除或禁用项目。（此外，片段还提供 `onPrepareOptionsMenu()` 回调。）

在 Android 2.3.x 及更低版本中，每当用户打开选项菜单时（按“*菜单*”按钮），系统均会调用 `onPrepareOptionsMenu()`。

在 Android 3.0 及更高版本中，当菜单项显示在应用栏中时，选项菜单被视为始终处于打开状态。 发生事件时，如果您要执行菜单更新，则必须调用 `invalidateOptionsMenu()` 来请求系统调用 `onPrepareOptionsMenu()`。

> **注：**切勿根据目前处于焦点的 `View` 更改选项菜单中的项目。 处于触摸模式（用户未使用轨迹球或方向键）时，视图无法形成焦点，因此切勿根据焦点修改选项菜单中的项目。 若要为 `View` 提供上下文相关的菜单项，请使用上下文菜单。

创建上下文菜单

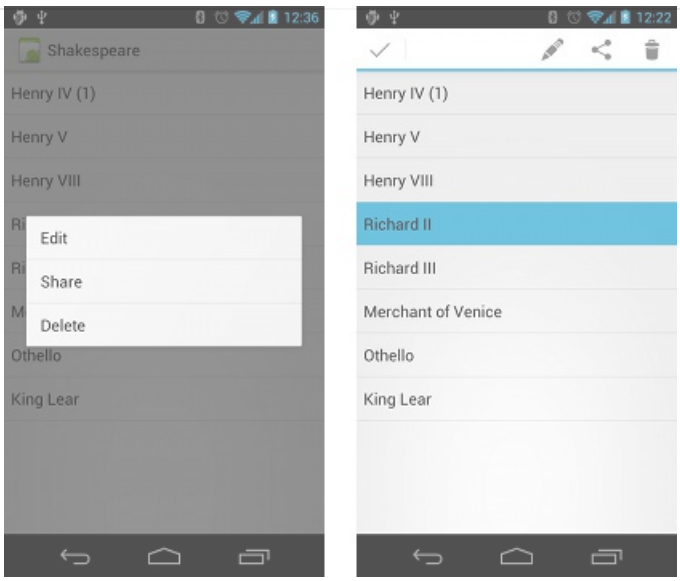

图 3. 浮动上下文菜单（左）和上下文操作栏（右）的屏幕截图。

上下文菜单提供了许多操作，这些操作影响 UI 中的特定项目或上下文框架。您可以为任何视图提供上下文菜单，但这些菜单通常用于 `ListView`、`GridView` 或用户可直接操作每个项目的其他视图集合中的项目。

提供上下文操作的方法有两种：

- 使用浮动上下文菜单。用户长按（按住）一个声明支持上下文菜单的视图时，菜单显示为菜单项的浮动列表（类似于对话框）。 用户一次可对一个项目执行上下文操作。
- 使用上下文操作模式。此模式是 `ActionMode` 的系统实现，它将在屏幕顶部显示上下文操作栏，其中包括影响所选项的操作项目。当此模式处于活动状态时，用户可以同时对多项执行操作（如果应用允许）。

> **注：**上下文操作模式可用于 Android 3.0（API 级别 11）及更高版本，是显示上下文操作（如果可用）的首选方法。如果应用支持低于 3.0 版本的系统，则应在这些设备上回退到浮动上下文菜单。

创建浮动上下文菜单

要提供浮动上下文菜单，请执行以下操作：

1. 通过调用 `registerForContextMenu()`，注册应与上下文菜单关联的 `View` 并将其传递给 `View`。

   如果 Activity 使用 `ListView` 或 `GridView` 且您希望每个项目均提供相同的上下文菜单，请通过将 `ListView` 或 `GridView` 传递给 `registerForContextMenu()`，为上下文菜单注册所有项目。

2. 在 `Activity` 或 `Fragment` 中实现 `onCreateContextMenu()` 方法。

   当注册后的视图收到长按事件时，系统将调用您的 `onCreateContextMenu()` 方法。在此方法中，您通常可通过扩充菜单资源来定义菜单项。例如：

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

   `MenuInflater` 允许您通过菜单资源扩充上下文菜单。回调方法参数包括用户所选的 `View`，以及一个提供有关所选项的附加信息的 `ContextMenu.ContextMenuInfo` 对象。如果 Activity 有多个视图，每个视图均提供不同的上下文菜单，则可使用这些参数确定要扩充的上下文菜单。

3. 实现 `onContextItemSelected()`。

   用户选择菜单项时，系统将调用此方法，以便您能够执行适当的操作。 例如：

   ```
   @Override
   public boolean onContextItemSelected(MenuItem item) {
       AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.getMenuInfo();
       switch (item.getItemId()) {
           case R.id.edit:
               editNote(info.id);
               return true;
           case R.id.delete:
               deleteNote(info.id);
               return true;
           default:
               return super.onContextItemSelected(item);
       }
   }
   ```

   `getItemId()` 方法将查询所选菜单项的 ID，您应使用 `android:id` 属性将此 ID 分配给 XML 中的每个菜单项，如使用 XML 定义菜单部分所示。

   成功处理菜单项后，系统将返回 `true`。如果未处理菜单项，则应将菜单项传递给超类实现。 如果 Activity 包括片段，则 Activity 将先收到此回调。 通过在未处理的情况下调用超类，系统会将事件逐一传递给每个片段中相应的回调方法（按照每个片段的添加顺序），直到返回 `true` 或 `false` 为止。（`Activity` 和 `android.app.Fragment` 的默认实现返回 `false`，因此您始终应在未处理的情况下调用超类。）

使用上下文操作模式

上下文操作模式是 `ActionMode` 的一种系统实现，它将用户交互的重点转到执行上下文操作上。用户通过选择项目启用此模式时，屏幕顶部将出现一个“上下文操作栏”，显示用户可对当前所选项执行的操作。 启用此模式后，用户可以选择多个项目（若您允许）、取消选择项目以及继续在 Activity 内导航（在您允许的最大范围内）。 当用户取消选择所有项目、按“返回”按钮或选择操作栏左侧的“*完成*”操作时，该操作模式将会停用，且上下文操作栏将会消失。

> **注：**上下文操作栏不一定与应用栏相关联。 尽管表面上看来上下文操作栏取代了应用栏的位置，但事实上二者独立运行。

对于提供上下文操作的视图，当出现以下两个事件（或之一）时，您通常应调用上下文操作模式：

- 用户长按视图。
- 用户选中复选框或视图内的类似 UI 组件。

应用如何调用上下文操作模式以及如何定义每个操作的行为，具体取决于您的设计。 设计基本上分为两种：

- 针对单个任意视图的上下文操作。
- 针对 `ListView` 或 `GridView` 中项目组的批处理上下文操作（允许用户选择多个项目并针对所有项目执行操作）。

下文介绍每种场景所需的设置。

为单个视图启用上下文操作模式

如果希望仅当用户选择特定视图时才调用上下文操作模式，则应：

1. 实现 `ActionMode.Callback` 接口。在其回调方法中，您既可以为上下文操作栏指定操作，又可以响应操作项目的点击事件，还可以处理操作模式的其他生命周期事件。
2. 当需要显示操作栏时（例如，用户长按视图），请调用 `startActionMode()`。

例如：

1. 实现 `ActionMode.Callback` 接口：

```
private ActionMode.Callback mActionModeCallback = new ActionMode.Callback() {

    // Called when the action mode is created; startActionMode() was called
    @Override
    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        // Inflate a menu resource providing context menu items
        MenuInflater inflater = mode.getMenuInflater();
        inflater.inflate(R.menu.context_menu, menu);
        return true;
    }

    // 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
    }

    // Called when the user selects a contextual menu item
    @Override
    public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_share:
                shareCurrentItem();
                mode.finish(); // Action picked, so close the CAB
                return true;
            default:
                return false;
        }
    }

    // Called when the user exits the action mode
    @Override
    public void onDestroyActionMode(ActionMode mode) {
        mActionMode = null;
    }
};
```

请注意，这些事件回调与选项菜单的回调几乎完全相同，只是其中每个回调还会传递与事件相关联的 `ActionMode` 对象。您可以使用 `ActionMode` API 对 CAB 进行各种更改，例如：使用 `setTitle()` 和 `setSubtitle()`（这对指示要选择多少个项目非常有用）修改标题和副标题。

另请注意，操作模式被销毁时，上述示例会将 `mActionMode` 变量设置为 null。 在下一步中，您将了解如何初始化该变量，以及保存 Activity 或片段中的成员变量有何作用。

2. 调用 `startActionMode()` 以便适时启用上下文操作模式，例如：响应对 `View` 的长按操作：

```
someView.setOnLongClickListener(new View.OnLongClickListener() {
    // Called when the user long-clicks on someView
    public boolean onLongClick(View view) {
        if (mActionMode != null) {
            return false;
        }

        // Start the CAB using the ActionMode.Callback defined above
        mActionMode = getActivity().startActionMode(mActionModeCallback);
        view.setSelected(true);
        return true;
    }
});
```

当您调用 `startActionMode()` 时，系统将返回已创建的 `ActionMode`。通过将其保存在成员变量中，您可以更改上下文操作栏来响应其他事件。 在上述示例中， `ActionMode` 用于在启动操作模式之前检查成员是否为空，以确保当 `ActionMode` 实例已激活时不再重建该实例。

在 ListView 或 GridView 中启用批处理上下文操作

如果您在 `ListView` 或 `GridView` 中有一组项目（或 `AbsListView` 的其他扩展），且需要允许用户执行批处理操作，则应：

- 实现 `AbsListView.MultiChoiceModeListener` 接口，并使用 `setMultiChoiceModeListener()` 为视图组设置该接口。在侦听器的回调方法中，您既可以为上下文操作栏指定操作，也可以响应操作项目的点击事件，还可以处理从 `ActionMode.Callback` 接口继承的其他回调。
- 使用 `CHOICE_MODE_MULTIPLE_MODAL` 参数调用 `setChoiceMode()`。

例如：

```
ListView listView = getListView();
listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE_MODAL);
listView.setMultiChoiceModeListener(new MultiChoiceModeListener() {

    @Override
    public void onItemCheckedStateChanged(ActionMode mode, int position,
                                          long id, boolean checked) {
        // Here you can do something when items are selected/de-selected,
        // such as update the title in the CAB
    }

    @Override
    public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
        // Respond to clicks on the actions in the CAB
        switch (item.getItemId()) {
            case R.id.menu_delete:
                deleteSelectedItems();
                mode.finish(); // Action picked, so close the CAB
                return true;
            default:
                return false;
        }
    }

    @Override
    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        // Inflate the menu for the CAB
        MenuInflater inflater = mode.getMenuInflater();
        inflater.inflate(R.menu.context, menu);
        return true;
    }

    @Override
    public void onDestroyActionMode(ActionMode mode) {
        // Here you can make any necessary updates to the activity when
        // the CAB is removed. By default, selected items are deselected/unchecked.
    }

    @Override
    public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
        // Here you can perform updates to the CAB due to
        // an invalidate() request
        return false;
    }
});
```

就这么简单。现在，当用户通过长按选择项目时，系统即会调用 `onCreateActionMode()` 方法，并显示包含指定操作的上下文操作栏。当上下文操作栏可见时，用户可以选择其他项目。

在某些情况下，如果上下文操作提供常用的操作项目，则您可能需要添加一个复选框或类似的 UI 元素来支持用户选择项目，这是因为他们可能没有发现长按行为。用户选中该复选框时，您可以通过使用 `setItemChecked()` 将相应的列表项设置为选中状态，以此调用上下文操作模式。

创建弹出菜单

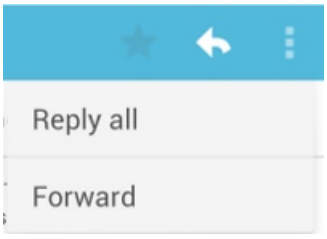

图 4. Gmail 应用中的弹出菜单，锚定到右上角的溢出按钮。

`PopupMenu` 是锚定到 `View` 的模态菜单。如果空间足够，它将显示在定位视图下方，否则显示在其上方。它适用于：

- 为与特定内容确切相关的操作提供溢出样式菜单（例如，Gmail 的电子邮件标头，如图 4 所示）。

> **注：**这与上下文菜单不同，后者通常用于影响所选内容的操作。 对于影响所选内容的操作，请使用上下文操作模式或浮动上下文菜单。

- 提供命令语句的另一部分（例如，标记为“添加”且使用不同的“添加”选项生成弹出菜单的按钮）。
- 提供类似于 `Spinner` 且不保留永久选择的下拉菜单。

> **注：**`PopupMenu` 在 API 级别 11 及更高版本中可用。

如果使用 XML 定义菜单，则显示弹出菜单的方法如下：

1. 实例化 `PopupMenu` 及其构造函数，该函数将提取当前应用的 `Context` 以及菜单应锚定到的 `View`。
2. 使用 `MenuInflater` 将菜单资源扩充到 `PopupMenu.getMenu()` 返回的 `Menu` 对象中。
3. 调用 `PopupMenu.show()`。

例如，以下是一个使用 `android:onClick` 属性显示弹出菜单的按钮：

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_overflow_holo_dark"
    android:contentDescription="@string/descr_overflow_button"
    android:onClick="showPopup" />
```

稍后，Activity 可按照如下方式显示弹出菜单：

```
public void showPopup(View v) {
    PopupMenu popup = new PopupMenu(this, v);
    MenuInflater inflater = popup.getMenuInflater();
    inflater.inflate(R.menu.actions, popup.getMenu());
    popup.show();
}
```

在 API 级别 14 及更高版本中，您可以将两行合并在一起，使用 `PopupMenu.inflate()` 扩充菜单。

当用户选择项目或触摸菜单以外的区域时，系统即会清除此菜单。 您可使用 `PopupMenu.OnDismissListener` 侦听清除事件。

处理点击事件

要在用户选择菜单项时执行操作，您必须实现 `PopupMenu.OnMenuItemClickListener` 接口，并通过调用 `setOnMenuItemclickListener()` 将其注册到 `PopupMenu`。 用户选择项目时，系统会在接口中调用 `onMenuItemClick()` 回调。

例如：

```java
public void showMenu(View v) {
    PopupMenu popup = new PopupMenu(this, v);

    // This activity implements OnMenuItemClickListener
    popup.setOnMenuItemClickListener(this);
    popup.inflate(R.menu.actions);
    popup.show();
}

@Override
public boolean onMenuItemClick(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.archive:
            archive(item);
            return true;
        case R.id.delete:
            delete(item);
            return true;
        default:
            return false;
    }
}
```

创建菜单组

菜单组是指一系列具有某些共同特征的菜单项。通过菜单组，您可以：

- 使用 `setGroupVisible()` 显示或隐藏所有项目
- 使用 `setGroupEnabled()` 启用或禁用所有项目
- 使用 `setGroupCheckable()` 指定所有项目是否可选中

通过将 `<item>` 元素嵌套在菜单资源中的 `<group>` 元素内，或者通过使用 `add()` 方法指定组 ID，您可以创建组。

以下是包含组的菜单资源示例：

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_save"
          android:icon="@drawable/menu_save"
          android:title="@string/menu_save" />
    <!-- menu group -->
    <group android:id="@+id/group_delete">
        <item android:id="@+id/menu_archive"
              android:title="@string/menu_archive" />
        <item android:id="@+id/menu_delete"
              android:title="@string/menu_delete" />
    </group>
</menu>
```

组中的项目出现在与第一项相同的级别，即：菜单中的所有三项均为同级。 但是，您可以通过引用组 ID 并使用上面列出的方法，修改组中两项的特征。此外，系统也绝不会分离已分组的项目。 例如，如果为每个项目声明 `android:showAsAction="ifRoom"`，则它们会同时显示在操作栏或操作溢出菜单中。

使用可选中的菜单项

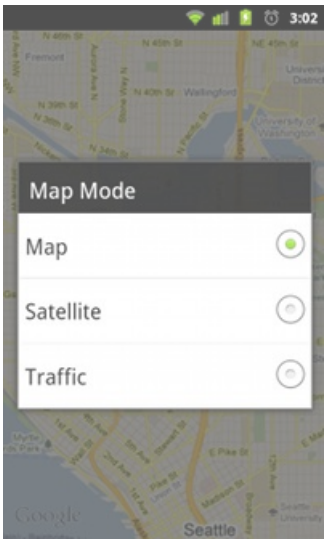

图 5. 含可选中项目的子菜单的屏幕截图。

作为启用/禁用选项的接口，菜单非常实用，既可针对独立选项使用复选框，也可针对互斥选项组使用单选按钮。 图 5 显示了一个子菜单，其中的项目可使用单选按钮选中。

> **注：**“图标菜单”（在选项菜单中）的菜单项无法显示复选框或单选按钮。 如果您选择使“图标菜单”中的项目可选中，则必须在选中状态每次发生变化时交换图标和/或文本，手动指出该状态。

您可以使用 `<item>` 元素中的 `android:checkable` 属性为各个菜单项定义可选中的行为，或者使用 `<group>` 元素中的 `android:checkableBehavior` 属性为整个组定义可选中的行为。例如，此菜单组中的所有项目均可使用单选按钮选中：

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item android:id="@+id/red"
              android:title="@string/red" />
        <item android:id="@+id/blue"
              android:title="@string/blue" />
    </group>
</menu>
```

`android:checkableBehavior` 属性接受以下任一选项：

`single`

组中只有一个项目可以选中（单选按钮）

`all`

所有项目均可选中（复选框）

`none`

所有项目均无法选中

您可以使用 `<item>` 元素中的 `android:checked` 属性将默认的选中状态应用于项目，并可使用 `setChecked()` 方法在代码中更改此默认状态。

选择可选中项目后，系统将调用所选项目的相应回调方法（例如，`onOptionsItemSelected()`）。 此时，您必须设置复选框的状态，因为复选框或单选按钮不会自动更改其状态。 您可以使用 `isChecked()` 查询项目的当前状态（正如用户选择该项目之前一样），然后使用 `setChecked()` 设置选中状态。例如：

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.vibrate:
        case R.id.dont_vibrate:
            if (item.isChecked()) item.setChecked(false);
            else item.setChecked(true);
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

如果未通过这种方式设置选中状态，则项目的可见状态（复选框或单选按钮）不会因为用户选择它而发生变化。 如果已设置该状态，则 Activity 会保留项目的选中状态。这样一来，当用户稍后打开菜单时，您设置的选中状态将会可见。

> **注：**可选中菜单项的使用往往因会话而异，且在应用销毁后不予保存。 如果您想为用户保存某些应用设置，则应使用共享首选项存储数据。

添加基于 Intent 的菜单项

有时，您希望菜单项通过使用 `Intent` 启动 Activity（无论该 Activity 是位于您的应用还是其他应用中）。如果您知道自己要使用的 Intent，且具有启动 Intent 的特定菜单项，则可在相应的 on-item-selected 回调方法（例如，`onOptionsItemSelected()` 回调）期间使用 `startActivity()` 执行 Intent。

但是，如果不确定用户的设备是否包含可处理 Intent 的应用，则添加调用 Intent 的菜单项可能会导致该菜单项无法正常工作，这是因为 Intent 可能无法解析为 Activity。 为了解决这一问题，当 Android 在设备上找到可处理 Intent 的 Activity 时，则允许您向菜单动态添加菜单项。

要根据接受 Intent 的可用 Activity 添加菜单项，请执行以下操作：

1. 使用类别 `CATEGORY_ALTERNATIVE` 和/或 `CATEGORY_SELECTED_ALTERNATIVE` 以及任何其他要求定义 Intent。
2. 调用 `Menu.addIntentOptions()`。Android 随后即会搜索能够执行 Intent 的所有应用，并将其添加到菜单中。

如果未安装可处理 Intent 的应用，则不会添加任何菜单项。

> **注：** `CATEGORY_SELECTED_ALTERNATIVE` 用于处理屏幕上当前所选的元素。因此，只有在 `onCreateContextMenu()` 中创建菜单时，才能使用它。

例如：

```
@Override
public boolean onCreateOptionsMenu(Menu menu){
    super.onCreateOptionsMenu(menu);

    // Create an Intent that describes the requirements to fulfill, to be included
    // in our menu. The offering app must include a category value of Intent.CATEGORY_ALTERNATIVE.
    Intent intent = new Intent(null, dataUri);
    intent.addCategory(Intent.CATEGORY_ALTERNATIVE);

    // Search and populate the menu with acceptable offering applications.
    menu.addIntentOptions(
         R.id.intent_group,  // Menu group to which new items will be added
         0,      // Unique item ID (none)
         0,      // Order for the items (none)
         this.getComponentName(),   // The current activity name
         null,   // Specific items to place first (none)
         intent, // Intent created above that describes our requirements
         0,      // Additional flags to control items (none)
         null);  // Array of MenuItems that correlate to specific items (none)

    return true;
}
```

如果发现 Activity 提供的 Intent 过滤器与定义的 Intent 匹配，则会添加菜单项，并使用 Intent 过滤器 `android:label` 中的值作为菜单项标

题，使用应用图标作为菜单项图标。`addIntentOptions()` 方法将返回已添加的菜单项数量。

> **注：**调用 `addIntentOptions()` 方法时，它将使用第一个参数中指定的菜单组替代所有菜单项。

允许将 Activity 添加到其他菜单中

此外，您还可以为其他应用提供您的 Activity 服务，以便您的应用能够包含在其他应用的菜单中（与上述角色相反）。

要包含在其他应用菜单中，您需要按常规方式定义 Intent 过滤器，但请确保为 Intent 过滤器类别添加 `CATEGORY_ALTERNATIVE` 和/或 `CATEGORY_SELECTED_ALTERNATIVE` 值。例如：

```
<intent-filter label="@string/resize_image">
    ...
    <category android:name="android.intent.category.ALTERNATIVE" />
    <category android:name="android.intent.category.SELECTED_ALTERNATIVE" />
    ...
</intent-filter>
```

请仔细阅读 Intent 和 Intent 过滤器文档中更多有关编写 Intent 过滤器的内容。

有关使用此方法的应用示例，请参阅记事本示例代码。