

# Android

## Explorando a Action Bar e o Uso de Menus



Softblue  
cursos online

---

---

---

---


---

---

---

---

Tópicos Abordados



- Action Bar
  - A classe *ActionBar*
  - Action buttons
    - Detectando cliques
    - Definidos em fragments
    - Visibilidade
    - Action overflow
  - Up navigation
    - Via configuração
    - Via programação
  - Mais sobre a action bar
  - Action bar em versões antigas do Android

---

---

---

---


---

---

---

---

Tópicos Abordados



- Menus
  - Options menu / Action bar
    - Alteração do menu
    - Submenus
  - Context Menus
    - Floating context menu
    - Contextual action mode
      - Action mode com uma view
      - Action mode com um grupo de views
  - Popup menus
  - Navigation Drawer

---

---

---

---

---

---

---

---

Action Bar


- A action bar é a barra superior da aplicação
- Surgiu no Android 3.0 (API Level 11)
- Benefícios
  - Identifica a aplicação
  - Permite a execução de ações
  - Torna a navegação mais intuitiva



---

---

---


---

---

---

---

---

A Classe *ActionBar*


- O objeto da classe **ActionBar** é o ponto de partida para gerenciar a action bar via programação
- A instância pode ser obtida através de uma activity

```
ActionBar actionBar = getSupportActionBar();
```

---

---

---


---

---


---

---

---

Action Buttons


- A action bar possui suporte aos chamados *action buttons*



- As ações são normalmente criadas através de um resource do tipo **menu**
  - Também é possível criar via programação

---

---

---

---

---

---

---

---

## Definindo os Action Buttons



```
/res/menu/activity_menu.xml
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item
    android:id="@+id/action_call"
    android:icon="@drawable/device_access_call"
    android:showAsAction="always"
    android:title="@string/call"/>
  <item
    android:id="@+id/action_camera"
    android:icon="@drawable/device_access_camera"
    android:showAsAction="ifRoom"
    android:title="@string/camera"/>
  <item
    android:id="@+id/action_network"
    android:icon="@drawable/device_access_network_wifi"
    android:showAsAction="ifRoom|withText"
    android:title="@string/network"/>
</menu>
```

---

---

---

---

---

---

---

---

## Definindo os Action Buttons



- O Android chama **onCreateOptionsMenu()** quando é iniciada, para exibir as ações

```
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_activity, menu);
    return super.onCreateOptionsMenu(menu);
}
```

O método **inflate()** extrai os itens definidos no XML

Deve ser **true**, mas é uma boa prática chamar o método da superclasse

---

---

---

---

---

---

---

---

## Detectando o Action Button Clicado



- Quando um action button é clicado, o método **onOptionsItemSelected()** é chamado na activity

```
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.action_call) {
        //...
        return true;
    } else if (id == R.id.action_camera) {
        //...
        return true;
    } else if (id == R.id.action_network) {
        //...
        return true;
    } else {
        return super.onOptionsItemSelected(item);
    }
}
```

**item.getItemId()** retorna o ID do resource clicado

Retorna **true** se o evento for tratado, ou chama o método da superclasse

---

---

---

---

---

---

---

---

## Detectando o Action Button Clicado



- É possível também definir o atributo **onClick** no XML, que referencia o método que será chamado quando o action button for clicado

```
<menu>
  <item>
    ...
    android:onClick="add"
    ... />
</menu>
```

```
public void add(MenuItem item) {
  //...
}
```

Recebe como parâmetro  
o item clicado

## Action Buttons e Fragments



- Fragments também podem adicionar action buttons na action bar

```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setHasOptionsMenu(true);
}

public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
  MenuInflater inflater = getActivity().getMenuInflater();
  inflater.inflate(R.menu.menu_activity, menu);
  super.onCreateOptionsMenu(menu);
}
```

Habilita a inserção  
de action buttons

O método **onOptionsItemSelected()**  
funciona da mesma forma que na activity

O atributo **orderInCategory** no XML pode  
ser utilizado na definição da ordem

## Visibilidade dos Action Buttons



- O Android determina como os botões serão dispostos na action bar
  - De acordo com o tamanho da tela
  - De acordo com parâmetros de configuração
- A configuração pode ser feita na definição do item, no XML
  - Atributo **showAsAction**

## Visibilidade dos Action Buttons

- Valores do atributo *showAsAction*

| Valor           | O action button...          |
|-----------------|-----------------------------|
| <b>ifRoom</b>   | Aparece se houver espaço    |
| <b>always</b>   | Sempre aparece              |
| <b>never</b>    | Nunca aparece               |
| <b>withText</b> | Aparece junto com seu texto |

---

---

---

---

---

---

---

---

## Action Overflow

- Local onde os action buttons ficam
  - Se marcados como *showAsAction = never*
  - Se não há espaço na action bar
- A forma de acessar o action overflow varia de acordo com o dispositivo

---

---

---

---

---

---

---

---

## Up Navigation

- A action bar permite o que chamamos de *up navigation*
  - Retornar para a tela anterior na hierarquia
- Conceitualmente, *back navigation* e *up navigation* são diferentes
- É preciso habilitar na activity
 

```
getActionBar().setDisplayHomeAsUpEnabled(true);
```

---

---

---

---

---

---

---

---

## Fazendo a Up Navigation Funcionar



- Apenas habilitar a up navigation na activity não é o suficiente
  - É preciso determinar para onde ir quando o botão up é pressionado
- Existem duas formas de fazer isso
  - Via configuração
  - Via programação

---

---

---

---

---

---

---

## Up Navigation via Configuração



- A “activity-pai” na hierarquia é definida no *AndroidManifest.xml*
- Utilizada quando a hierarquia é fixa



---

---

---

---

---

---

---

## Up Navigation via Programação



- O método **onOptionsItemSelected()** implementa o comportamento
- Utilizada quando a hierarquia pode variar

```
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == android.R.id.home) {
        Intent i = new Intent(getApplicationContext(), ActivityA.class);
        i.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        startActivity(i);
        return true;
    } else {
        return super.onOptionsItemSelected(item);
    }
}
```

ID associado ao botão Up

---

---

---

---

---

---

---

## Mais Sobre a Action Bar



- O objeto *ActionBar* tem mais alguns métodos interessantes

| Método                                    | Descrição                               |
|-------------------------------------------|-----------------------------------------|
| <code>show()</code>                       | Exibe a action bar                      |
| <code>hide()</code>                       | Esconde a action bar                    |
| <code>setDisplayShowTitleEnabled()</code> | Mostra ou esconde o título da activity  |
| <code>setDisplayShowHomeEnabled()</code>  | Mostra ou esconde o ícone da action bar |

- A action bar também é totalmente customizável em sua parte gráfica
  - Consulte a documentação do Android para mais informações

---

---

---

---

---

---

---

## Action Bar em Versões Anteriores



- A action bar surgiu no Android 3.0 (API Level 11)
- Para ser usada no Android 2.1 (API Level 7) ou acima, é preciso usar a API de compatibilidade
- Não existe suporte à action bar para versões anteriores à 2.1
  - A não ser com o uso de APIs de terceiros

---

---

---

---

---

---

---

## Referenciando a Action Bar



- A activity deve herdar de **ActionBarActivity** e chamar o método **getSupportActionBar()**

```
public class MainActivity extends ActionBarActivity {  
    public void onCreate(Bundle savedInstanceState) {  
        ...  
        ActionBar actionBar = getSupportActionBar();  
        ...  
    }  
}
```

Classe do pacote  
`android.support.v7.app`

---

---


---

---

---

---

---

Criando os Action Buttons


- O atributo **showAsAction** não existe nas versões anteriores do Android
- A referência deve ser feita usando outro namespace

```

<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:myapp="http://schemas.android.com/apk/res-auto">

  <item
    ...
    myapp:showAsAction="ifRoom"
    ... />

```

O namespace **android** não pode ser utilizado

---

---

---


---

---

---

---

---

Menus no Android


- Menus são elementos gráficos que permitem que os usuários executem ações através de cliques
- O Android suporta diversos tipos de menus
  - Options menu / Action bar
  - Context menu
  - Popup menus

---

---

---


---

---


---

---


---

Options Menu / Action bar


- A action bar pode ter um menu de opções



- Usar a action bar é a forma recomendada
- Em versões anteriores ao Android 3.0, é possível usar o options menu
  - Acessível através do botão “Menu” do dispositivo



Em termos de programação, ambas as técnicas são iguais

---

---

---

---

---

---

---

---



## Alteração do Menu



- O método `onCreateOptionsMenu()` é chamado apenas uma vez
  - Isto faz com que o menu seja estático
- Para atualizar o menu de forma dinâmica, de acordo com outros parâmetros, é preciso implementar **`onPrepareOptionsMenu()`**

```
public boolean onPrepareOptionsMenu(Menu menu) {  
    //....  
    return super.onPrepareOptionsMenu(menu);  
}
```

---

---

---

---

---

---

---

## Alteração do Menu



- Se a versão do Android for anterior à 3.0 (API Level 11), `onPrepareOptionsMenu()` é chamado toda vez que o options menu é aberto
- Se a versão do Android for 3.0 ou superior, é preciso chamar **`invalidateOptionsMenu()`** para que `onPrepareOptionsMenu()` seja invocado

---

---

---

---

---

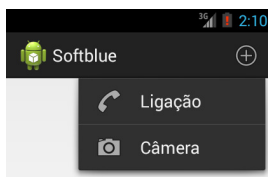
---

---

## Submenus



- O options menu e action bar podem contar com submenus



---

---

---

---

---

---

---

## Definindo Submenus



- A definição da hierarquia dos menus e submenus é feita no arquivo de layout

```
<menu>
  <item
    android:id="@+id/action_more"
    android:icon="@android:drawable/ic_menu_add"
    android:showAsAction="ifRoom"
    android:title="Ações">
    <menu>
      <item
        android:id="@+id/action_call"
        android:icon="@android:drawable/ic_menu_call"
        android:title="Ligação" />
      <item
        android:id="@+id/action_camera"
        android:icon="@android:drawable/ic_menu_camera"
        android:title="Câmera" />
    </menu>
  </item>
</menu>
```

---

---

---

---

---

---

---

## Context Menus



- São menus de contexto
  - Associados a uma ou mais views da tela
- São normalmente usados em listas de elementos
- Podem ser exibidos de duas formas
  - Floating context menu
  - Contextual action mode

---

---

---

---

---

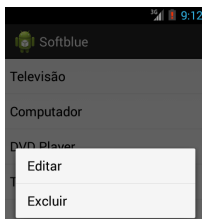
---

---

## Floating Context Menu



- O menu aparece como uma caixa de diálogo
- Ativado com um clique longo na view



---

---


---

---

---

---

---

Criando Floating Context Menus


- Registrar as views que acionarão o menu

```

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ...
    setListAdapter(adapter);
    registerForContextMenu(getListView());
}

```

- Implementar **onCreateContextMenu()**,  
invocado sempre que o context menu é aberto

```

public void onCreateContextMenu(ContextMenu menu,
    View v, ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    //...
}

```

---

---

---


---

---

---

---

---

Criando Floating Context Menus


- Toda vez que uma opção de um context menu é escolhida, o Android chama o método **onContextItemSelected()**

```

public boolean onContextItemSelected(MenuItem item) {
    //...
    return super.onOptionsItemSelected(item);
}

```

---

---

---


---

---

---

---

---

Contextual Action Mode


- Floating context menus eram usados até o android 2.3.3 (API Level 10)
- Do Android 3.0 em diante, é recomendado usar o contextual action mode
- Continua sendo um menu de contexto associado a uma ou mais views
- O menu aparece no topo da tela, no local onde fica a action bar
- Pode ser de dois tipos
  - Associado a uma view qualquer
  - Associado a um grupo de views (Ex: *ListView*)

---

---

---

---

---

---

---

---

## Action Mode em uma View



- Primeiramente é preciso implementar a interface **ActionMode.Callback**

```
public class MainActivity extends Activity implements
    ActionMode.Callback {

    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
    }

    public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
    }

    public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
    }

    public void onDestroyActionMode(ActionMode mode) {
    }
}
```

---

---

---

---

---

---

---

## Action Mode em uma View



- Chamar o método **startActionMode()** quando for o momento de exibir a barra de menu de contexto

```
public void onCreate(Bundle savedInstanceState) {
    View view = findViewById(R.id.view);
    view.setOnLongClickListener(new View.OnLongClickListener() {
        public boolean onLongClick(View v) {
            startActionMode(MainActivity.this);
            return true;
        }
    });
}
```



Context menu

Teste de menu

---

---

---

---

---

---

---

## Action Mode em um Grupo de Views



- É preciso implementar a interface **AbsListView.MultiChoiceModeListener**

```
public class MainActivity extends Activity implements
    AbsListView.MultiChoiceModeListener {

    public boolean onCreateActionMode(ActionMode mode, Menu menu) { }

    public boolean onPrepareActionMode(ActionMode mode, Menu menu) { }

    public boolean onActionItemClicked(ActionMode mode, MenuItem item) { }

    public void onItemCheckedStateChanged(ActionMode mode, int position,
        long id, boolean checked) { }

    public void onDestroyActionMode(ActionMode mode) { }
}
```

```
listView.setChoiceMode(AbsListView.CHOICE_MODE_MULTIPLE_MODAL)
```

---

---

---

---

---

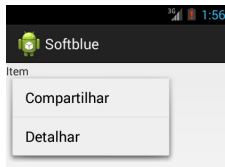
---

---

## Popup Menu



- Também é um menu associado a uma view
- O popup menu deve ser usado para oferecer opções que não interferem na view
  - Context menus podem interferir na view



---

---

---

---

---

---

---

## Criando Popup Menus



- Criar uma instância de **PopupMenu**

```
PopupMenu popup = new PopupMenu(this, view);  
MenuInflater inflater = popup.getMenuInflater();  
inflater.inflate(R.menu.menu_activity, popup.getMenu());  
popup.show();
```

```
PopupMenu popup = new PopupMenu(this, view);  
popup.inflate(R.menu.menu_activity);  
popup.show();
```

---

---

---

---

---

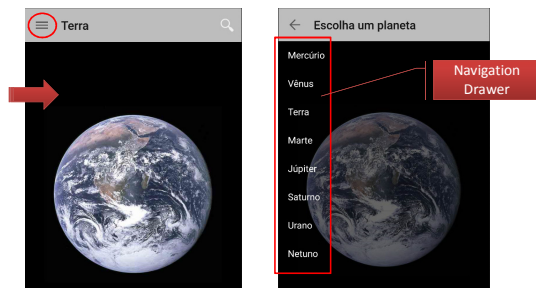
---

---

## O padrão Navigation Drawer



- O Navigation Drawer é um menu que aparece e desaparece a partir da lateral



---

---

---

---

---

---

---

## O padrão Navigation Drawer



- O Navigation Drawer precisa da API de compatibilidade para funcionar
  - *support-v4*
    - Layout `android.support.v4.widget.DrawerLayout`
  - *appcompat-v7*
    - Classe `android.support.v7.app.ActionBarDrawerToggle`

---

---

---

---

---

---

---



---

---

---

---

---

---

---