KAPITOLA 12

Widgety nabídek

V této kapitole:
- Přizpůsobení se okolnostem
- Seznamy nabídek
- Režimy výběru
- Rozevírací nabídky
- Mřížky
- O 35 % méně psaní na klávesnici při práci s textovými poli
- Galerie

V kapitole 11 jste se naučili, jak lze u textových polí používat omezení specifikující formát dat, která do nich lze vložit – například pouze čísla nebo pouze telefonní čísla. Tento druh omezení pomáhá uživateli vložit do pole data ve správném formátu, a to zejména v mobilních zařízeních s maličkou klávesnicí.

Výběr však lze samozřejmě omezit také na určitou množinu položek, například skupinu přepínačů. Klasické sady nástrojů na tvorbu uživatelského rozhraní za tímto účelem nabízejí seznamy, rozevírací seznamy, rozevírací nabídky a podobné ovládací prvky. Systém Android podporuje mnoho stejných typů widgetů a navíc ještě některé zvláštní typy, které se hodí zejména k použití v mobilním zařízení (například třídu `Gallery` k prohlížení uložených fotografií).

Kromě toho systém Android nabízí také flexibilní architekturu k určení položek, z nichž lze prostřednictvím těchto widgetů vybírat. Konkrétně se jedná o architekturu datových adaptérů, které zajišťují společné rozhraní pro seznamy možností a umí seznamům dodat data z nejrůznějších zdrojů, od statických polí až po databáze. Výběrová zobrazení – widgety pro prezentaci seznamů možností – tedy mají své adaptéry, které jim dodávají obsah.

Přizpůsobení se okolnostem

Z teoretického hlediska adaptéry zajišťují společné rozhraní pro více různých nesourodých rozhraní API. V případě systému Android se pak konkrétněji jedná o zajištění společného rozhraní pro datový model, který stojí za widgety nabídek, jako je například seznam. Tento způsob využití rozhraní jazyka Java je velmi častý (viz například adaptéry knihoven Swing jazyka Java pro třídu `JTable`) a Java není jediným prostředím, které tento druh abstrakce nabízí (architektura datových vazeb založená na formátu XML v jazyce Flex například pracuje s vloženým kódem XML nebo kódem XML staženým z Internetu).

Adaptéry systému Android však widgetům nedodávají pouze data, ale také konvertují jednotlivé položky na konkrétní zobrazení v nabídkách. Tato druhá funkce architektury adaptérů vám možná připadá poněkud zvláštní, ale ve skutečnosti se příliš neliší od způsobů, kterými nahrazují výchozí zobrazovací chování i jiné sady nástrojů na tvorbu uživatelského rozhraní. Pokud například chcete při práci s knihovnami Swing jazyka Java konvertovat seznam založený na třídě `JList` na seznam se zaškrtávacími políčky, skončíte nevyhnutelně u volání metody `setCellRenderer()`, abyste mohli dodat vaši vlastní instanci třídy `ListCellRenderer`, která pak konvertuje řetězce seznamu na složené widgety tvořené instancemi tříd `JCheckBox` a `JLabel`.

Nejjednodušší je práce s adaptérem `ArrayAdapter`. Jednoduše obalíte jeho instancí pole jazyka Java nebo instanci typu `java.util.List`, a máte plně funkční adaptér:

```
String[] items={"tohle", "je", "opravdu",
                "hodně", "uhozený", "seznam"};
new ArrayAdapter<String>(this,
  android.R.layout.simple_list_item_1, items);
```

Jeden z konstruktorů adaptéru `ArrayAdapter` přijímá tři parametry:

- kontext (instanci třídy `Context`), který se má použít (to bude obvykle instance vaší aktivity),
- ID prostředku zobrazení, který se má použít (například ID vestavěného systémového prostředku jako v předchozím příkladu),
- samotné pole nebo seznam položek, které se mají zobrazit.

Ve výchozím nastavení zavolá adaptér `ArrayAdapter` metodu `toString()` objektů v seznamu a obalí každý z řetězců zobrazením dodaným jako prostředek. Definice rozložení `android.R.layout.simple_list_item_1` pak jednoduše konvertuje tyto řetězce na objekty typu `TextView`. Tyto widgety typu `TextView` se následně zobrazí v seznamu, v rozevírací nabídce nebo v jakémkoliv jiném widgetu, který tento adaptér `ArrayAdapter` používá. Pokud si chcete definici rozložení `android.R.layout.simple_list_item_1` prohlédnout, vyhledejte ji ve své instalaci sady SDK.

V kapitole 13 se naučíte, jak vytvořit podtřídu třídy `Adapter` a přepsat operaci vytvoření řádku tak, abyste měli větší kontrolu nad jeho výslednou podobou.

Seznamy nabídek

Klasický widget seznamu nabídek je v systému Android třídy `ListView`. Vložte instanci této třídy do své definice rozložení, zavolejte metodu `setAdapter()`, abyste jí dodali data a zobrazení, která budou jejími potomky, přidělte jí prostřednictvím metody `setOnItemSelectedListener()` naslouchací objekt, abyste byli upozorněni na provedenou volbu, a máte plně funkční seznam nabídek.

Avšak tvoří-li vaši aktivitu jediný seznam nabídek, můžete také zvážit její vytvoření jako podtřídy třídy `ListActivity` místo podtřídy obvyklé základní třídy `Activity`. Pokud je vaše hlavní zobrazení pouhý seznam, nemusíte dokonce ani dodat definici jeho rozložení; třída `ListActivity` pro vás zkonstruuje seznam přes celou obrazovku. Chcete-li jeho podobu upravit, můžete to udělat, pokud svou instanci třídy `ListView` označíte identifikátorem `@android:id/list`, aby třída `ListActivity` věděla, který widget je hlavním seznamem aktivity.

Zde je jednoduchý příklad seznamu a jednoho popisku zobrazujícího aktuální volbu, který najdete ve složce vzorového projektu `Selection/List`:

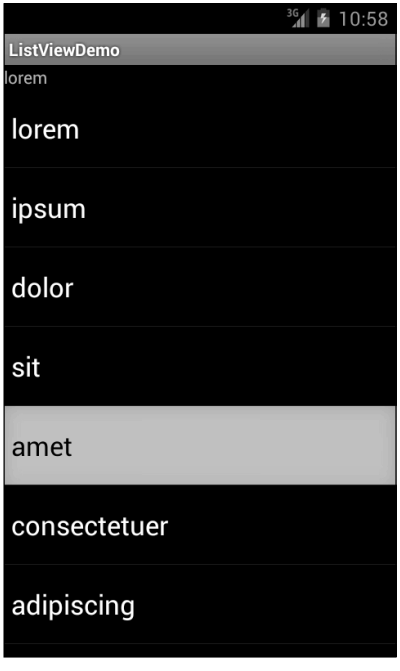
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent" >
  <TextView
    android:id="@+id/selection"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"/>
  <ListView
    android:id="@android:id/list"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:drawSelectorOnTop="false"
    />
</LinearLayout>
```

Kód jazyka Java, který slouží ke konfiguraci seznamu a jeho propojení s popiskem, vypadá takto:

```
public class ListViewDemo extends ListActivity {
  private TextView selection;
  private static final String[] items={"lorem", "ipsum", "dolor",
          "sit", "amet",
          "consectetuer", "adipiscing", "elit", "morbi", "vel",
          "ligula", "vitae", "arcu", "aliquet", "mollis",
          "etiam", "vel", "erat", "placerat", "ante",
          "porttitor", "sodales", "pellentesque", "augue", "purus"};

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    setListAdapter(new ArrayAdapter<String>(this,
                        android.R.layout.simple_list_item_1,
                        items));
```

```
    selection=(TextView)findViewById(R.id.selection);
  }

  public void onListItemClick(ListView parent, View v, int position,
                              long id) {
    selection.setText(items[position]);
  }
}
```

Odvodíte-li třídu své aktivity od třídy `ListActivity`, můžete nastavit adaptér seznamu pomocí metody `setListAdapter()`, které v tomto případě předáte instanci třídy `ArrayAdapter` obalující pole řetězců. Abyste měli možnost zjistit, zda se změnil výběr v seznamu, nahradíte metodu `onListItemClick()` a provedete příslušné kroky na základě dodaného podřazeného zobrazení a pozice v seznamu – v tomto případě se jedná o aktualizaci textu popisku textem této pozice. Výsledná aplikace je na obrázku 12.1.



Obrázek 12.1 Vzorová aplikace ListViewDemo

Druhý parametr adaptéru `ArrayAdapter` specifikuje vzhled řádků. Hodnota `android.R.layout.simple_list_item_1` tohoto parametru použitá v předchozím příkladu specifikuje standardní řádek seznamu nabídek systému Android: velký font, hodně výplně a bílý text.

Režimy výběru

Widget `ListView` je ve výchozím nastavení nastaven tak, aby jednoduše vracel klepnutí na položky seznamu. Pomocí widgetu `ListView` však lze také vytvořit seznam, který provedený výběr, nebo dokonce více výběrů za sebou uchovává, ale je třeba jej mírně upravit.

Zaprvé je třeba ve zdrojovém kódu jazyka Java zavolat metodu `setChoiceMode()` objektu `ListView` a definovat pomocí ní režim výběru – předáním parametru `CHOICE_MODE_SINGLE` nebo `CHOICE_MODE_MULTIPLE`. Widget `ListView` lze získat od objektu `ListActivity` pomocí metody `getListView()`. Režim výběru lze také deklarovat nastavením atributu `android:choiceMode` v definici rozložení XML na příslušnou hodnotu.

Poté je nutné použít místo definice rozložení řádků seznamu `android.R.layout.simple_list_item_1` definici `android.R.layout.simple_list_item_single_choice` nebo `android.R.layout.simple_list_item_multiple_choice`, která určuje seznam umožňující provedení jedné nebo více voleb (v uvedeném pořadí).

Zde je například rozložení aktivity ze vzorového projektu `Selection/Checklist`:

```
<?xml version="1.0" encoding="utf-8"?>
<ListView
xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@android:id/list"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:drawSelectorOnTop="false"
  android:choiceMode="multipleChoice"
/>
```

Jedná se o widget `ListView` přes celou obrazovku s atributem `android:choiceMode` nastaveným na hodnotu `multipleChoice`, který značí, že vyžadujeme podporu více voleb.

Aktivita používá pro seznam vzorových slov standardní adaptér `ArrayAdapter`, ale tentokrát používá definici rozložení řádku `android.R.layout.simple_list_item_multiple_choice`:

```
package com.commonsware.android.checklist;

import android.os.Bundle;
import android.app.ListActivity;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class ChecklistDemo extends ListActivity {
  private static final String[] items={"lorem", "ipsum", "dolor",
          "sit", "amet",
          "consectetuer", "adipiscing", "elit", "morbi", "vel",
          "ligula", "vitae", "arcu", "aliquet", "mollis",
```

```
          "etiam", "vel", "erat", "placerat", "ante",
          "porttitor", "sodales", "pellentesque", "augue", "purus"};

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    setListAdapter(new ArrayAdapter<String>(this,
                        android.R.layout.simple_list_item_multiple_choice,
                        items));
  }
}
```

Uživateli se zobrazí seznam slov se zaškrtávacími políčky u levého okraje, který si můžete prohlédnout na obrázku 12.2.

Obrázek 12.2. Režim výběru více položek

Kdybychom chtěli, mohli bychom také pomocí metody `getCheckedItemPositions()` widgetu `ListView` zjistit, které položky uživatel v seznamu zaškrtnul, nebo pomocí metody `setItemChecked()` zaškrtnout (nebo odškrtnout) některé položky sami.

Rozevírací nabídky

Widget `Spinner` je v systému Android ekvivalentem rozevírací nabídky, kterou najdete i v jiných sadách nástrojů na tvorbu uživatelského rozhraní (například třída `ComboBox` v knihovnách Swing jazyka Java). Stisknete-li prostřední tlačítko na směrovém ovladači telefonu, nabídka se rozbalí a uživatel z ní může vybrat nějakou položku. Rozevírací nabídka tak v zásadě nabízí možnost výběru z výčtu možností, aniž byste zabrali celou obrazovku instancí třídy `ListView`, avšak za cenu dalšího stisknutí tlačítka nebo klepnutí na obrazovku.

Stejně jako v případě widgetu `ListView` i widgetům `Spinner` dodáváte adaptér pro jejich data a podřazená zobrazení prostřednictvím metody `setAdapter` a prostřednictvím metody `setOnItemSelectedListener()` přiřazujete naslouchací objekt, který bude upozorněn na provedenou změnu.

Chcete-li si ušít definici rozložení používanou při zobrazování rozbalené nabídky na míru, musíte nastavit adaptér, a ne widget `Spinner`. K dodání ID prostředku zobrazení, který se má použít, použijte metodu `setDropDownViewResource()`.

Zde je jednoduchý příklad definice rozložení s widgetem spinner, který najdete ve složce vzorového projektu `Selection/Spinner`:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  >
  <TextView
    android:id="@+id/selection"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
  <Spinner android:id="@+id/spinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:drawSelectorOnTop="true"
  />
</LinearLayout>
```

Jedná se o stejné zobrazení, které jsme použili v předchozí podkapitole, ale tentokrát místo `widgetu ListView` obsahuje widget `Spinner`. Atribut widgetu `Spinner android:drawSelectorOnTop` určuje, zda rozbalovací tlačítko se šipkou překryje položku seznamu.

K naplnění nabídky položkami a k jejímu použití je třeba kód jazyka Java:

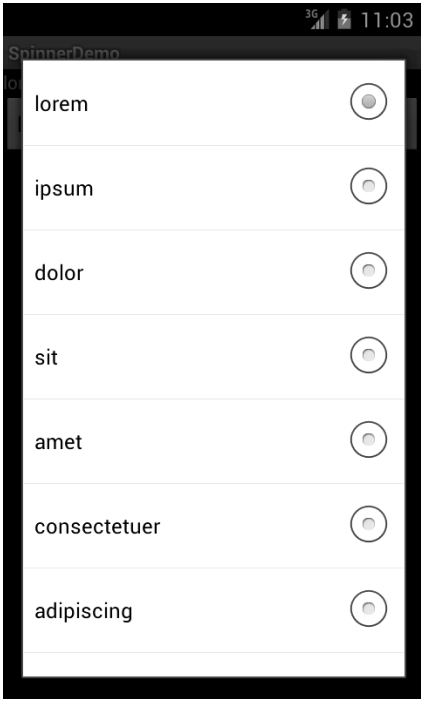
```
public class SpinnerDemo extends Activity
  implements AdapterView.OnItemSelectedListener {
  private TextView selection;
  private static final String[] items={"lorem", "ipsum", "dolor",
          "sit", "amet",
          "consectetuer", "adipiscing", "elit", "morbi", "vel",
          "ligula", "vitae", "arcu", "aliquet", "mollis",
          "etiam", "vel", "erat", "placerat", "ante",
          "porttitor", "sodales", "pellentesque", "augue", "purus"};

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    selection=(TextView)findViewById(R.id.selection);

    Spinner spin=(Spinner)findViewById(R.id.spinner);
    spin.setOnItemSelectedListener(this);

    ArrayAdapter<String> aa=new ArrayAdapter<String>(this,
                            android.R.layout.simple_spinner_item,
                            items);

    aa.setDropDownViewResource(
      android.R.layout.simple_spinner_dropdown_item);
    spin.setAdapter(aa);
  }

  public void onItemSelected(AdapterView<?> parent,
                             View v, int position, long id) {
    selection.setText(items[position]);
  }

  public void onNothingSelected(AdapterView<?> parent) {
    selection.setText("");
  }
}
```

Zde přiřazujeme jako naslouchací objekt výběru samotnou aktivitu (`spin.setOnItemSelectedListener(this)`). Tímto způsobem můžeme postupovat, protože aktivita implementuje rozhraní `OnItemSelectedListener`. Kromě seznamu vzorových slov pak definujeme také konkrétní prostředek, který se použije pro rozevírací nabídku (pomocí volání `aa.setDropDownViewResource()`). Všimněte si také použití definice rozložení `android.R.layout.`

`simple_spinner_item` jako vestavěného widgetu `View` pro zobrazování položek v samotném rozevíracím seznamu.

Nakonec implementujeme zpětně volané metody, které vyžaduje rozhraní `OnItemSelected-Listener` k úpravám popisku výběru na základě výběru provedeného uživatelem. Výsledek si můžete prohlédnout na obrázcích 12.3 a 12.4.

Obrázek 12.3 Vzorová aplikace SpinnerDemo po spuštění



Obrázek 12.4 Stejná aplikace se zobrazenou rozevírací nabídkou

Mřížky

Jak již napovídá jeho název, widget `GridView` poskytuje dvojdimenzionální mřížku položek, ze kterých si může uživatel vybrat. Zároveň máte možnost určité kontroly nad počtem a rozměry jejích sloupců, počet řádků se určuje dynamicky na základě počtu položek, které dodaný adaptér specifikuje jako zobrazitelné.

Počet sloupců a jejich rozměry specifikuje kombinace několika atributů:

- `android:numColumns`: Specifikuje počet sloupců. Atribut také můžete nastavit na hodnotu `auto_fit`, která značí, že se má počet sloupců vypočítat na základě volného místa na obrazovce a hodnot následujících atributů.
- `android:verticalSpacing` a `android:horizontalSpacing`: Tyto atributy specifikují mezery mezi jednotlivými položkami v mřížce.
- `android:columnWidth`: Specifikuje šířku sloupců v pixelech.
- `android:stretchMode`: Specifikuje pro mřížky s hodnotou atributu `android:numColumns` `auto_fit`, jak se má naložit s volným místem, které není zabrané sloupci nebo mezerami. Tento atribut může mít hodnotu `columnWidth`, která specifikuje, že mají

volné místo zabrat sloupce, nebo `spacingWidth`, která specifikuje, že mají volné místo zabrat mezery mezi sloupci.

Jinak funguje widget `GridView` v podstatě stejně jako jakýkoliv jiný widget nabídky: používá metodu `setAdapter()` k dodání dat a podřazených zobrazení, metodu `setOnItemSelectedListener()` k registraci naslouchacích objektů výběru atd.

Zde je jednoduchý příklad definice rozložení ve formátu XML, který najdete ve složce vzorového projektu `Selection/Grid`, demonstrující konfiguraci widgetu typu `GridView`:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  >
  <TextView
    android:id="@+id/selection"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
  <GridView
    android:id="@+id/grid"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:verticalSpacing="40dip"
    android:horizontalSpacing="5dip"
    android:numColumns="auto_fit"
    android:columnWidth="100dip"
    android:stretchMode="columnWidth"
    android:gravity="center"
    />
</LinearLayout>
```

Tato mřížka zabírá celou obrazovku kromě místa potřebného pro popisek výběru. Počet sloupců mřížky určuje systém (`android:numColumns = "auto_fit"`) na základě rozteče sloupců (`android:horizontalSpacing = "5dip"`) a šířky sloupců (`android:columnWidth = "100dip"`) a sloupce „absorbují“ jakékoliv zbytky volného místa (`android:stretchMode = "columnWidth"`).

Zdrojový kód jazyka Java konfigurující widget `GridView` vypadá takto:

```
package com.commonsware.android.grid;

import android.app.Activity;
import android.content.Context;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.GridView;
import android.widget.TextView;

public class GridDemo extends Activity
  implements AdapterView.OnItemSelectedListener {
  private TextView selection;
  private static final String[] items={"lorem", "ipsum", "dolor",
          "sit", "amet",
          "consectetuer", "adipiscing", "elit", "morbi", "vel",
          "ligula", "vitae", "arcu", "aliquet", "mollis",
          "etiam", "vel", "erat", "placerat", "ante",
          "porttitor", "sodales", "pellentesque", "augue", "purus"};

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    selection=(TextView)findViewById(R.id.selection);

    GridView g=(GridView) findViewById(R.id.grid);
    g.setAdapter(new ArrayAdapter<String>(this,
                        R.layout.cell,
                        items));
    g.setOnItemSelectedListener(this);
  }

  public void onItemSelected(AdapterView<?> parent, View v,
                             int position, long id) {
    selection.setText(items[position]);
  }

  public void onNothingSelected(AdapterView<?> parent) {
    selection.setText("");
  }
}
```

Zobrazení buněk mřížky jsou definována samostatným souborem definice rozložení `res/layout/cell.xml`, na který se v adaptéru `ArrayAdapter` odkazujeme jako na `R.layout.cell`:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:textSize="14dip"
/>
```

Díky definici rozteče řádků v definici rozložení XML (`android:verticalSpacing = "40dip"`) mřížka přesahuje přes hranice obrazovky – viz obrázky 12.5 a 12.6.

Obrázek 12.5 Vzorová aplikace GridDemo po spuštění

Obrázek 12.6 Stejná aplikace zobrazující poslední řádek mřížky

O 35 % méně psaní na klávesnici při práci s textovými poli

Widget `AutoCompleteTextView` je jakýmsi hybridem mezi widgetem `EditText` (textové pole) a `Spinner` (rozevírací nabídka). Díky nástroji na automatické dokončování zadávaného textu se se zadávaným textem nakládá jako s filtrem předpon a text se porovnává se seznamem možných kandidátů. Nalezení kandidáti (řetězce znaků), kteří začínají písmeny shodujícími se se zadávaným řetězcem, se rozbalují v nabídce směrem dolů z textového pole, stejně jako v případě rozevírací nabídky. Uživatel pak buď může dokončit zadávání vstupu na klávesnici (například řetězce, který není na seznamu), nebo vybrat odpovídající položku z rozbalené nabídky a zadat ji tak jako hodnotu textového pole.

Třída `AutoCompleteTextView` je podtřídou třídy `EditText`, takže můžete nastavit všechny standardní vizuální a ovládací vlastnosti widgetu, například jeho font a barvu. Kromě těchto atributů má widget `AutoCompleteTextView` ještě atribut `android:completionThreshold`, jehož hodnota určuje minimální počet znaků, které musí uživatel zadat, než se zahájí filtrování nabídky kandidátů.

Widgetu `AutoCompleteTextView` můžete prostřednictvím metody `setAdapter()` přidělit adaptér obsahující seznam kandidátů. Avšak protože uživatel může zadat hodnotu, která na seznamu kandidátů není, nepodporuje třída `AutoCompleteTextView` naslouchací objekt provedeného výběru. Místo toho můžete stejně jako v případě jakéhokoliv jiného widgetu `EditText` zaregistrovat rozhraní `TextWatcher`, které vás upozorní na změnu obsahu textového pole. Tato událost je pak buď výsledkem ručního zadání vstupu, nebo výběru z rozevírací nabídky kandidátů.

Následující známá definice rozložení ve formátu XML tentokrát obsahuje definici widgetu `AutoCompleteTextView` (najdete jej ve složce vzorové aplikace `Selection/AutoComplete`):

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  >
  <TextView
    android:id="@+id/selection"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
  <AutoCompleteTextView android:id="@+id/edit"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:completionThreshold="3"/>
</LinearLayout>
```

Příslušný zdrojový kód jazyka Java pak vypadá takto:

```
package com.commonsware.android.auto;
import android.app.Activity;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.TextView;

public class AutoCompleteDemo extends Activity
  implements TextWatcher {
  private TextView selection;
  private AutoCompleteTextView edit;
```

```
  private static final String[] items={"lorem", "ipsum", "dolor",
          "sit", "amet",
          "consectetuer", "adipiscing", "elit", "morbi", "vel",
          "ligula", "vitae", "arcu", "aliquet", "mollis",
          "etiam", "vel", "erat", "placerat", "ante",
          "porttitor", "sodales", "pellentesque", "augue", "purus"};

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    selection=(TextView)findViewById(R.id.selection);
    edit=(AutoCompleteTextView)findViewById(R.id.edit);
    edit.addTextChangedListener(this);

    edit.setAdapter(new ArrayAdapter<String>(this,
                        android.R.layout.simple_dropdown_item_1line,
                        items));
  }

  public void onTextChanged(CharSequence s, int start, int before,
                            int count) {
    selection.setText(edit.getText());
  }

  public void beforeTextChanged(CharSequence s, int start,
                                int count, int after) {
    // Potřeba kvůli rozhraní, ale nepoužívá se.
  }

  public void afterTextChanged(Editable s) {
    // Potřeba kvůli rozhraní, ale nepoužívá se.
  }
}
```

Aktivita tentokrát implementuje rozhraní `TextWatcher`, takže používáme zpětně volané metody `onTextChanged()`, `beforeTextChanged()` a `afterTextChanged()`. V tomto případě však používáme pouze metodu `onTextChanged()` a aktualizujeme popisek výběru tak, aby odpovídal aktuálnímu obsahu widgetu `AutoCompleteTextView`. Výsledek si můžete prohlédnout na obrázcích 12.7, 12.8 a 12.9.