KAPITOLA 7

Widgety nabídek

V této kapitole:

- Přizpůsobení se okolnostem
- Výčty nabídek
- Rozbalovací menu
- Mřížka nabídek
- O 35 % méně psaní na klávesnici při práci s textovými poli
- Galerie

V 5. kapitole jste se dočetli, jak lze na textová pole aplikovat omezení specifikující formát dat, která do nich lze vložit – například pouze čísla. Tento druh omezení pomáhá uživateli vložit do pole data ve správném formátu, a to zejména na mobilních zařízeních s maličkou klávesnicí.

Omezit pak lze samozřejmě i výběr z určité množiny položek, například ze skupiny přepínačů (popis skupiny přepínačů najdete také v 5. kapitole). Klasické sady nástrojů pro tvorbu uživatelského rozhraní za tímto účelem nabízejí výčty, rozbalovací výčty, drop-down menu a podobné ovládací prvky. Android podporuje mnoho stejných druhů widgetů a navíc ještě některé zvláštní druhy, které se hodí zejména pro použití na mobilním zařízení (například třídu `Gallery` pro prohlížení uložených fotografií).

Kromě toho Android nabízí také flexibilní prostředí pro určení položek, z nichž lze prostřednictvím těchto widgetů vybírat. Konkrétně se jedná o prostředí datových adaptérů, které zajišťují společné rozhraní pro výčty nabídek a umí nabídkám dodat data z nejrůznějších zdrojů, od statických polí až po obsah databází. Náhledy nabídek – widgety pro prezentaci výčtu možností – tedy mají svůj adaptér, který jim dodává jejich obsah.

Na začátku této kapitoly se nejprve společně podíváme na adaptéry systému Android a poté si řekneme více o widgetech nabídek.

Přizpůsobení se okolnostem

Z teoretického hlediska adaptéry zajišťují společné rozhraní pro více různých nesourodých API. V případě systému Android se pak konkrétněji jedná o zajištění společného rozhraní pro datový model, který stojí za widgety nabídek, jako je například výčet. Tento způsob využití Java rozhraní je běžný (viz například Java/Swing adaptéry pro třídu `JTable`) a Java není jediným prostředím, které tento druh abstrakce nabízí (XML data-binding prostředí jazyka Flex například přijímá XML dokumenty v inline podobě nebo stažené z Internetu).

Adaptéry systému Android jsou zodpovědné za dodávání soupisu dat widgetům nabídek a za konvertování jednotlivých datových položek na specifické náhledy, které se ve widgetech nabídek zobrazují. Tato druhá funkce systému adaptérů vám možná připadá poněkud zvláštní, ale ve skutečnosti se příliš neliší od způsobů, kterými nahrazují výchozí zobrazovací chování i jiné sady nástrojů pro tvorbu uživatelského rozhraní. Pokud například chcete v jazyce Java nebo Swing konvertovat výčet založený na třídě `JList` na výčet se zaškrtnutelnými políčky, skončíte nevyhnutelně u volání metody `setCellRenderer()`, abyste mohli dodat vaši vlastní instanci třídy `ListCellRenderer`, která pak konvertuje řetězce výčtu na složené widgety tvořené instancemi tříd `JCheckBox` a `JLabel`.

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

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

Konstruktor třídy `ArrayAdapter` přijímá tři parametry:

- Kontext (instanci třídy `Context`), který se má použít (to bude obvykle instance vaší aktivity).
- ID prostředku náhledu, který se má použít (například ID prostředku zabudovaného do systému, jako v předchozím příkladu).
- Samotné pole nebo výčet položek, které se mají zobrazit.

Ve výchozím nastavení zavolá třída `ArrayAdapter` pro objekty na výčtu metodu `toString()` a obalí každý z řetězců náhledem dodaným jako prostředek. `android.R.layout.simple_list_item_1` jednoduše konvertuje tyto řetězce na objekty typu `TextView`. Tyto widgety typu `TextView` se následně zobrazí na výčtu, v rozbalovacím menu nebo jakémkoliv jiném widgetu, který tento adaptér typu `ArrayAdapter` používá. V kapitole 8 se dočtete, jak vytvořit podtřídu třídy `Adapter` a nahradit operaci vytvoření řádku tak, abyste měli větší kontrolu nad jeho výslednou podobou.

Zde jsou další dva adaptéry systému Android, které by se vám mohly hodit:

- `CursorAdapter`: Konvertuje instanci třídy `Cursor`, obvykle dodanou dodavatelem obsahu, na něco, co lze zobrazit v náhledu nabídky. (Podrobnější rozbor třídy `CursorAdapter` najdete v kapitole 22, která je věnovaná databázím.)
- `SimpleAdapter`: Konvertuje data nalezená v XML prostředcích.

Výčty nabídek

Klasický widget výčtu nabídek je v systému Android třídy `ListView`. Vložte instanci této třídy do vašeho návrhu, zavolejte metodu `setAdapter()`, abyste jí dodali data a náhledy, které budou jejími potomky, přidělte posluchač prostřednictvím metody `setOnItemSelectedListener()`, abyste byli upozorněni na provedenou volbu, a máte plně funkční výčet nabídek.

Avšak tvoří-li vaši aktivitu jediný výčet nabídek, můžete také zvážit její vytvoření jako podtřídy třídy `ListActivity` místo podtřídy obvyklé základní třídy `Activity`. Pokud je váš hlavní náhled pouhým výčtem, nemusíte dokonce ani dodat jeho návrh; třída `ListActivity` pro vás zkonstruuje výčet přes celou obrazovku. Chcete-li jeho podobu upravit, můžete to udělat, pokud svoji instanci třídy `ListView` identifikujete jako `@android:id/list`, aby třída `ListActivity` věděla, který widget je hlavním výčtem aktivity.

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

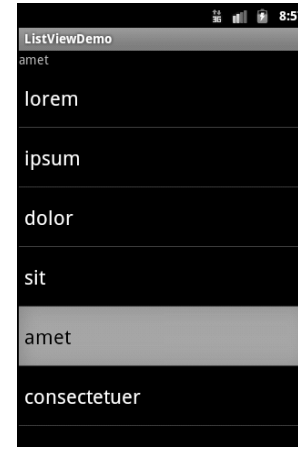
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent" >
  <TextView
    android:id="@+id/selection"
    android:layout_width="fill_parent"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

A zde je zdrojový kód jazyka Java, který konfiguruje výčet a propojuje jej s popiskem:

```
public class ListViewDemo extends ListActivity {
  TextView selection;
  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 vaší aktivity od třídy `ListActivity`, můžete nastavit adaptér výčtu prostřednictvím metody `setListAdapter()` – které v tomto případě předáte instanci třídy `ArrayAdapter` obalující pole řetězců. Abyste zjistili, že se změnil provedený výběr, nahraďte metodu `onListItemClick()` a proveďte příslušné kroky na základě dodaného náhledu potomka a pozice na výčtu – v tomto případě se jedná o aktualizaci textu popisku textem této pozice. Výsledná aplikace je na obrázku 7.1.

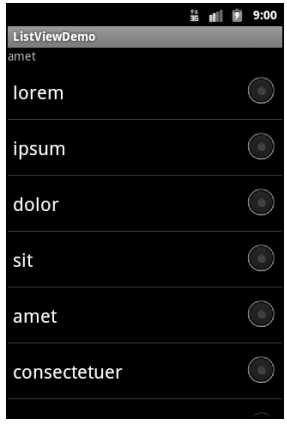


Obrázek 7.1: Vzorová aplikace ListViewDemo

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

Ve výchozím nastavení instance třídy `ListView` jednoduše informuje o klepnutí na záznamy ve výčtu. Potřebujete-li použít výčet, který si pamatuje provedený výběr nebo umožňuje výběr více položek najednou, můžete také použít instanci třídy `Listview`, ale musíte provést několik úprav:

- Zavolat ve zdrojovém kódu jazyka Java metodu instance třídy `ListView` `setChoiceMode()` a předat jí jednu z následujících hodnot: `CHOICE_MODE_SINGLE`, nebo `CHOICE_MODE_MULTIPLE`. Instanci třídy `ListView` můžete získat z instance třídy `ListActivity` prostřednictvím volání její metody `getListView()`.
- Místo toho, abyste specifikovali podobu řádků na výčtu prostřednictvím hodnoty `android.R.layout.simple_list_item_1`, použijte buď hodnotu `android.R.layout.simple_list_item_single_choice` pro výběr jediného řádku (viz obrázek 7.2), nebo `android.R.layout.simple_list_item_multiple_choice` pro výběr více různých řádků najednou (viz obrázek 7.3).
- Chcete-li určit, které řádky uživatel zaškrtnul, zavolejte metodu `getCheckedItemPositions()` vaší instance třídy `ListView`.



Obrázek 7.2: Výběr jediného řádku

Obrázek 7.3: Výběr více řádků najednou

Rozbalovací menu

Třída `Spinner` je v systému Android ekvivalentem rozbalovacího menu, které najdete i v jiných sadách nástrojů pro tvorbu uživatelského rozhraní (například třída `ComboBox` v sadě nástrojů Swing pro tvorbu uživatelského rozhraní pro Java aplikace). Stisknete-li prostřední tlačítko na směrovém ovladači telefonu, menu se rozbalí a uživatel z něj může vybrat nějakou položku. Získáte tak v zásadě možnost výběru z výčtu možností, aniž byste zabrali celou obrazovku instancí třídy `ListView`, avšak za cenu dalšího klepnutí či poklepání na obrazovku.

Stejně jako v případě třídy `ListView`, i instancím třídy `Spinner` dodáváte adaptér pro jejich data a náhledy potomků prostřednictvím metody `setAdapter` a prostřednictvím metody `setOnItemSelectedListener()` přiřazujete objekt posluchače, který bude upozorněn na provedenou změnu.

Chcete-li si ušít náhled používaný při zobrazování rozbaleného menu na míru, musíte nastavit adaptér, a ne widget typu `Spinner`. Pro dodání ID prostředku náhledu, který se má použít, použijte metodu `setDropDownViewResource()`.

Zde je jednoduchý příklad náhledu s rozbalovacím menu, 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>
```

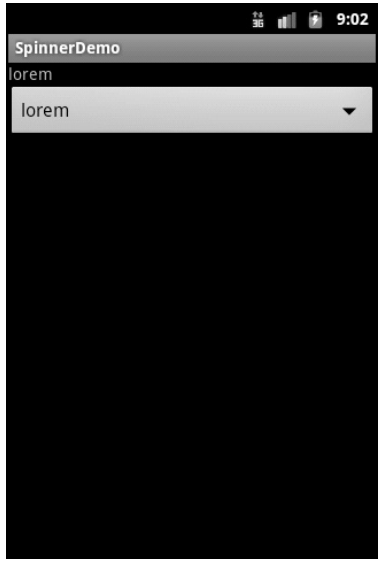
Jedná se o stejný náhled jako v předchozí sekci, ale tentokrát v něm místo widgetu typu `ListView` používáte widget typu `Spinner`. Atribut widgetu typu `Spinner`, `android:drawSelectorOnTop`, specifikuje, zda se na ovládacím tlačítku na pravé straně widgetu vykreslí šipka.

Abyste rozbalovací menu zaplnili položkami a mohli jej používat, musíte napsat také nějaký zdrojový kód jazyka Java:

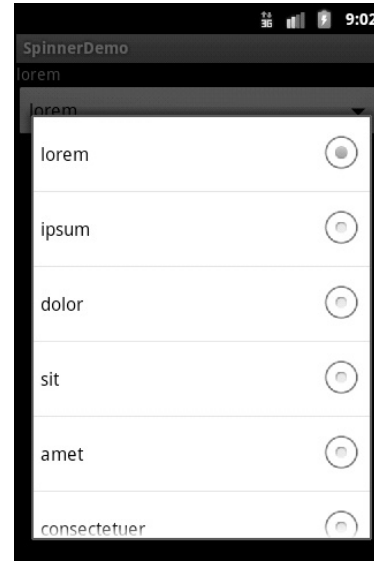
```
public class SpinnerDemo extends Activity
  implements AdapterView.OnItemSelectedListener {
  TextView selection;
  String[] items={"lorem", "ipsum", "dolor", "sit", "amet",
          "consectetuer", "adipiscing", "elit", "morbi", "vel",
          "ligula", "vitae", "arcu", "aliquet", "mollis",
          "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 přiřazujete jako posluchače provedeného výběru samotnou aktivitu (`spin.setOnItemSelectedListener(this)`). To můžete udělat, protože aktivita implementuje rozhraní `OnItemSelectedListener`. Adaptér nakonfigurujete nejen výčtem řetězců, ale i konkrétním prostředkem, který má pro náhled rozbalovacího menu používat (prostřednictvím metody `aa.setDropDownViewResource()`). Všimněte si také použití hodnoty `android.R.layout.simple_spinner_item`, specifikující zabudovaný náhled pro zobrazování položek menu.

Nakonec implementujete callback metody vyžadované rozhraním `OnItemSelectedListener` pro nastavení popisku, který reflektuje provedený výběr. Výsledná aplikace je na obrázcích 7.4 a 7.5.



Obrázek 7.4: Vzorová aplikace SpinnerDemo po spuštění



Obrázek 7.5: Vzorová aplikace SpinnerDemo s rozbaleným menu

Mřížka nabídek

Jak již napovídá název jeho třídy, widget typu `GridView` vám dodává dvoudimenzionální 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, anebo, pokud tento atribut nastavíte na hodnotu `auto_fit`, že má vypočítat počet sloupců na základě volného místa na obrazovce a hodnot následujících atributů systému.
- `android:verticalSpacing` a `android:horizontalSpacing`: Specifikuje 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.

Představte si například, že má displej 320 pixelů na šířku a vy máte atribut `android:columnWidth` nastavený na hodnotu `100px` a atribut `android:horizontalSpacing` na hodnotu `5px`. Tři sloupce zaberou 310 pixelů (tři sloupce po 100 pixelech a dvě mezery po 5 pixelech). Nastavíte-li atribut `android:stretchMode` na hodnotu `columnWidth`, roztáhnou se všechny tři sloupce o 3 až 4 pixe-

ly, aby využily zbývajících 10 pixelů. Nastavíte-li atribut `android:stretchMode` na hodnotu `spacingWidth`, roztáhnou se o 5 pixelů a zaberou tak zbývající volné místo obě mezery.

Jinak funguje widget typu `GridView` v podstatě stejně jako jakýkoliv jiný widget nabídky: používá metodu `setAdapter()` k dodání dat a náhledů potomků, metodu `setOnItemSelectedListener()` k registraci posluchačů výběru atd.

Zde je jednoduchý příklad XML návrhu, 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="35px"
    android:horizontalSpacing="5px"
    android:numColumns="auto_fit"
    android:columnWidth="100px"
    android:stretchMode="columnWidth"
    android:gravity="center"
    />
</LinearLayout>
```

Tato mřížka zabírá celou obrazovku kromě místa potřebného pro popisek reflektující provedený výběr. Počet sloupců mřížky počítá systém (`android:numColumns = "auto_fit"`) na základě rozměru horizontálních mezer 5 pixelů (`android:horizontalSpacing = "5px"`) a šířky sloupců 100 pixelů (`android:columnWidth = "100px"`). Sloupce pak absorbují jakékoliv zbytky volného prostoru v horizontálním směru (`android:stretchMode = "columnWidth"`).

Zdrojový kód jazyka Java potřebný k nakonfigurování mřížky vypadá takto:

```
public class GridDemo extends Activity
  implements AdapterView.OnItemSelectedListener {
  TextView selection;
  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 FunnyLookingAdapter(this,
                        android.R.layout.simple_list_item_1,
                        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("");
  }

  private class FunnyLookingAdapter extends ArrayAdapter {
    Context ctxt;

    FunnyLookingAdapter(Context ctxt, int resource,
                        String[] items) {
      super(ctxt, resource, items);

      this.ctxt=ctxt;
    }

    public View getView(int position, View convertView,
                        ViewGroup parent) {
      TextView label=(TextView)convertView;

      if (convertView==null) {
        convertView=new TextView(ctxt);
        label=(TextView)convertView;
      }

      label.setText(items[position]);

      return(convertView);
    }
  }
}
```

Tentokrát používáte pro buňky mřížky místo automaticky generovaných widgetů typu `TextView` jako v předchozích náhledech vaše vlastní náhledy tak, že vytváříte podtřídu třídy `ArrayAdapter` a nahrazujete metodu `getView()`. V tomto případě obalujete řetězce do vašich vlastních widgetů typu `TextView`, aby vypadaly trochu jinak. Pokud metoda `getView` obdrží instanci třídy `TextView`, pouze resetujete její text; jinak vytvoříte novou instanci třídy `TextView` a zaplníte ji.

S vertikálními mezerami o šířce 35 pixelů nastavenými XML souboru návrhu (`android:verticalSpacing="35"`) mřížka přesahuje hranice obrazovky emulátoru – viz obrázky 7.6 a 7.7.

Obrázek 7.6: Vzorová aplikace GridDemo po spuštění

Obrázek 7.7: Vzorová aplikace GridDemo zobrazující spodní část mřížky

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

Třída `AutoCompleteTextView` je jakýmsi hybridem mezi třídou `EditText` (textové pole) a `Spinner` (rozbalovací menu). Díky nástroji pro automatické dokončování zadávaného textu se s zadávaným textem nakládá jako s filtrem předpon a text se porovnává s výčtem možných kandidátů. Nalezené kandidáty (řetězce znaků), které začínají písmeny shodujícími se se zadávaným řetězcem, se rozbalují v menu směrem dolů z textového pole, stejně jako v případě rozbalovacího menu. Uživatel pak buď může dokončit zadávání vstupu na klávesnici (například řetězce, který není na výčtu), anebo vybrat odpovídající položku z rozbaleného menu a nastavit 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 třída `AutoCompleteTextView` ještě vlastnost `android:completionThreshold`, jejíž hodnota specifikuje minimální počet znaků, které musí uživatel zadat, než se zahájí filtrování nabídky kandidátů.

Widgetu typu `AutoCompleteTextView` můžete prostřednictvím metody `setAdapter()` přidělit adaptér obsahující výčet hodnot kandidátů. Avšak protože uživatel může zadat hodnotu, která na výčtu kandidátů není, nepodporuje třída `AutoCompleteTextView` posluchače provedeného výběru. Místo toho můžete stejně jako v případě jakéhokoliv jiného widgetu typu `EditText` zaregistrovat rozhraní `TextWatcher`, které vás upozorní na změnu obsahu textového pole. Tato událost je pak buď výsledkem ručně vloženého vstupu, nebo výběru z rozbalovací nabídky kandidátů.

Následující povědomý XML návrh tentokrát obsahuje definici widgetu typu `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>
```

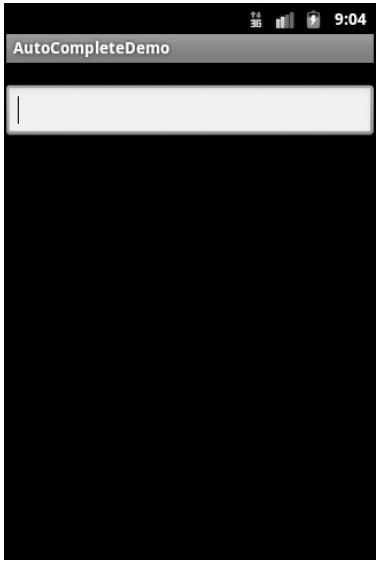
Příslušný zdrojový kód vypadá takto:

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

Vaše aktivita tentokrát implementuje rozhraní `TextWatcher`, což znamená, že vaše zpětná volání představují metody `onTextChanged()` a `beforeTextChanged()`. V tomto případě vás zajímá pouze ta první, ve které aktualizujete popisek reprezentující provedený výběr tak, aby odpovídal aktuálnímu obsahu widgetu typu `AutoCompleteTextView`.

Výsledná aplikace je na obrázcích 7.8, 7.9 a 7.10.



Obrázek 7.8: Vzorová aplikace AutoCompleteDemo po spuštění

Obrázek 7.9: Stejná aplikace po zadání několika prvních písmen, která odpovídají nějakému kandidátu, zobrazující rozbalovací menu využívající nástroj pro automatické dokončování vkládaného textu

Galerie

Widget typu `Gallery` obvykle v sadách nástrojů pro tvorbu uživatelského rozhraní nenajdete. Ve skutečnosti se totiž jedná o horizontálně uložený výčet nabídek. Jedna položka výčtu leží vedle druhé v horizontální ose a aktuálně vybraná položka je zvýrazněná. V systému Android pak uživatelé položkami procházejí prostřednictvím tlačítek „vlevo“ a „vpravo“ na směrovém ovladači.

Ve srovnání s widgetem typu `ListView` zabírá widget typu `Gallery` méně prostoru na obrazovce, a přesto zobrazuje v jednu chvíli více různých položek (za předpokladu že mají odpovídající rozměry). Ve srovnání s rozbalovacím menu pak galerie vždy zobrazuje více než jednu položku.

Typický příkladem použití widgetu typu `Gallery` je náhled kolekce obrázků. Předáte-li widgetu typu `Gallery` kolekci fotografií nebo ikon, widget zobrazí jejich náhledy a umožní uživateli, aby si některý z nich vybral.

Obrázek 7.10: Stejná aplikace po vybrání kandidátu

Z hlediska zdrojového kódu funguje galerie velmi podobně jako rozbalovací menu nebo mřížka. Ve své XML návrhu pak můžete pro galerii použít několik atributů:

- `android:spacing`: Specifikuje počet pixelů mezi záznamy na výčtu.
- `android:spinnerSelector`: Specifikuje způsob označení provedeného výběru. Může se jednat buď o odkaz na instanci typu `Drawable` (viz 20. kapitola), nebo o RGB hodnotu ve formátu `#AARRGGBB` nebo jiném podobném formátu.
- `android:drawSelectorOnTop`: Specifikuje, zda se má selektor označující provedený výběr (nebo odkaz na `Drawable`) zobrazit před (`false`), anebo po (`true`) zobrazení vybraného potomka. Zvolíte-li hodnotu `true`, ujistěte se, že je váš selektor dostatečně transparentní na to, aby za ním byl zobrazovaný potomek vidět; jinak uživatel vybranou položku neuvidí.