KAPITOLA

5

Uživatelské rozhraní a události

Témata kapitoly:

- **Kontejnery na rozmístění prvků**
- **Příklad – Definice rozložení prvků**
- **Příklad – Spuštění jiné aktivity**
- **Ladění aplikace**

Každý prvek v uživatelském rozhraní aplikace pro Android je podtřídou třídy `View`, přesněji `android.view.View`. Android SDK obsahuje sadu předpřipravených prvků, které se nejčastěji používají k tvorbě uživatelského rozhraní. Typickým příkladem jsou prvky typu `TextView`, `Button`, `Checkbox`, `Progressbar` a mnohé další. Objekty (prvky) uživatelského rozhraní se často označují i jako widgety nebo komponenty. Kromě předdefinovaných standardních prvků, které jsou součástí Android SDK, můžete vytvářet vlastní komponenty. Můžete je vytvářet jako rozšíření existujících tříd nebo vytvořit komponenty úplně nové.

Každý prvek odvozený od třídy `View` zabírá v uživatelském rozhraní odpovídající obdélníkovou plochu. Prvek je zodpovědný za to, co se v tomto obdélníku zobrazuje, a také za to, aby adekvátně reagoval na události, ke kterým dochází v rámci této části obrazovky, například na dotykové události. Obrazovka uživatelského rozhraní se skládá z hierarchie prvků `View`. Na nejvyšší úrovni hierarchie je kořenový element a podřízené (nebo v tomto případě vnořené), prvky jsou umístěny na nižších větvích hierarchického stromu.

Pohled (`view`) může být i kompozitní, to znamená složený z několika pohledů. Takovéto pohledy jsou podtřídy třídy `Android ViewGroup` (`Android.view.ViewGroup`) a tato třída je zase podtřídou `View`. Dobrým příkladem takovéhoto pohledu je `RadioGroup`, prvek, který vizuálně i funkčně zapouzdřuje více objektů `RadioButton`. Je to přesná analogie reálného světa, protože prvek `RadioButton` je ekvivalentem jednoho z více tlačítek, která se na starých radiopřijímačích používala k přepínání vlnových rozsahů. V poloze „zapnuto“ mohlo být vždy pouze jedno ze skupiny tlačítek.

Ale vraťme se k prvkům uživatelského rozhraní aplikace pro Android. Z hlediska struktury kompozitních prvků se tyto skládají z jednoho nadřazeného pohledu odvozeného od třídy `ViewGroup`, který nazýváme také kontejner, nebo výstižněji

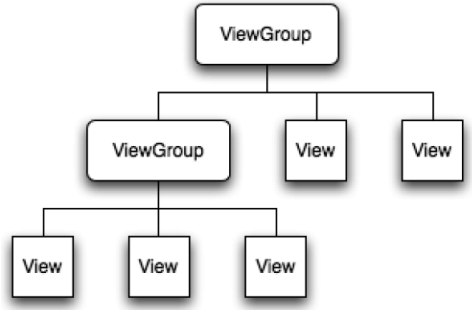
vizuální kontejner, jelikož v sobě zapouzdřuje prvky uživatelského rozhraní, případně tyto mohou být různě hierarchicky sdružené ve vnořených kontejnerech. Pro vnořené komponenty z hlediska syntaxe XML je vizuální kontejner kořenovým elementem.

Návrh uživatelského rozhraní

Základem tvorby interaktivního uživatelského rozhraní jsou pohledy, skupiny pohledů a aktivity.

- `Views` (pohledy) jsou základní třídy pro tvorbu vizuálních prvků uživatelského rozhraní známé i pod názvem widgety. Všechny ovládací prvky uživatelského rozhraní a třídy uspořádání jsou odvozeny od `Views`.
- `ViewGroups` jsou rozšířením pohledů pro tvorbu složených prvků, které, jak vyplývá z názvu, obsahují více `Views`.
- `Activities` (aktivity) jsou ekvivalentem formulářů nebo dialogových oken s ovládacími prvky, které znáte z jiných platforem. Je to nejmenší logická jednotka aplikace s uživatelským prostředím. Může obsahovat propojení na jiné aktivity.

Skupiny pohledů umožňují libovolné uspořádání hierarchie uživatelského rozhraní, takže na rozdíl od jiných platforem má vývojář při definování architektury volnou ruku. Při vytvoření projektu podle šablony **Blank Activity** si v souboru *fragment_main.xml* všimněte, že prvek



`<TextView>` je umístěn uvnitř skupiny pohledů `<RelativeLayout>`.

Obrázek 5.1: Hierarchie objektů VievGroup

Nejčastěji se k návrhu uživatelského rozhraní využívají následující prvky:

- `TextView` – prvek umožní zobrazení textového výpisu, například hodnoty nějaké textové proměnné a podobně. Základním parametrem, který nastavujeme u tohoto prvku, je text.
- `EditText` – zatímco prvek `TextView` slouží k jednosměrné komunikaci od aplikace k uživateli, to znamená, že jen zobrazuje výpisy, `EditText` je jednoduché nebo víceřádkové editační okno pro zadávání textových údajů. Prvek podporuje i zarovnávání slov.
- `ImageView` – prvek na zobrazení obrázku.
- `Spinner` – kompozitní prvek na výběr více položek přizpůsobený výběru na malé ploše dotykového displeje telefonu. Seznam možností je možné rozbalit a některou z nich dotykem vybrat.
- `Button` – standardní tlačítko aktivované dotykem.

- `CheckBox` – tento dvojstavový ovládací prvek umožňuje přepínat mezi dvěma hodnotami – pravda/nepravda (true/false). Pokud je symbol prvku zaškrtnut, nabývá hodnotu true.
- `RadioButton` – ovládací prvek `RadioButton` sám o sobě velký smysl nemá, je potřeba použít těchto prvků několik, potom fungují jako přepínač (odtud analogie `RadioButton` – tlačítková sada na přepínání rozsahů rozhlasového přijímače).
- `ProgressBar` – prvek, pomocí něhož aplikace indikuje uživateli probíhající činnost. Pomocí parametru `android:indeterminate` určujete, zda se jedná o `ProgressBar` konečný, který ukazuje průběh v procentech, nebo nekonečný, kde nelze určit, v jakém stadiu dokončení se prováděná akce nachází.

Kontejnery na rozmístění prvků

Jak už bylo uvedeno, k definici rozmístění prvků uživatelského rozhraní slouží takzvané vizuální kontejnery, v původní terminologii `Layouts`. Jsou odvozeny od třídy `ViewGroup`.

Rozmístění prvků musí být flexibilní. Za všechny uvedeme jen dva důvody: různé rozměry obrazovek a změnu orientace zařízení.

Při pootočení zařízení z režimu „na výšku" do režimu „na šířku", případně naopak, je zpravidla zapotřebí nejen geometrická transformace, ale i změna rozmístění prvků, aby co nejlépe využily plochu obrazovky v daném režimu.

LinearLayout

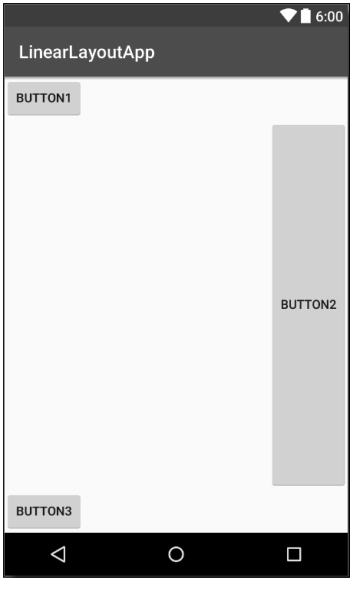
Umožňuje uspořádat vnořené prvky (`views`) horizontálně do jednoho řádku nebo vertikálně do jednoho sloupce. Klíčové parametry pro `LinearLayout` jsou:

- `orientation` – směr uspořádání vnořených prvků.
- `layout_gravity` – takzvané těžiště vizuálního kontejneru, které definuje centrální bod zarovnání vnořených `view`.
- `Layout_weight` – priorita vnořeného `view` v rámci layoutu. Čím vyšší hodnota, tím vyšší priorita. Pokud máte v rámci layoutu tři prvky, přičemž dva z nich mají nastavený parametr `weight` na `0` a třetí na hodnotu `1`, dva prvky s nižší prioritou zaberou prostor daný jejich obsahem a třetí prvek s vyšší prioritou vyplní zbytek plochy layoutu.

V následujícím ilustračním příkladu je uživatelské rozhraní využívající `LinearLayout` s vertikální orientací sestávající ze tří tlačítek. První a třetí nemá nastavený parametr `weight`, což v praxi znamená, že je implicitně nastavený na hodnotu 0. Prostřední tlačítko má nastavený parametr `weight` na hodnotu `1`, takže má vyšší prioritu a zabere na výšku zbylou plochu. Parametr `gravity` je nastaven střídavě na hodnoty `left` a `right`.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  tools:context="sk.pcrevue.llacko.linearlayoutapp.MainActivity">

  <Button
```

```
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="left"
    android:text="Button1" />

  <Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:layout_gravity="right"
    android:layout_weight="1"
    android:text="Button2" />


  <Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="left"
    android:text="Button3" />

</LinearLayout>
```



Obrázek 5.2: Ilustrace fungování parametrů grafity a weight

RelativeLayout

Relativní layouty umožňují definování vzájemných pozičních vztahů nejen mezi jednotlivými vizuálními prvky uživatelského rozhraní (views), ale i vůči rodičovskému layoutu. Aby bylo možné definovat vzájemné vazby, musí mít každý prvek jednoznačný identifikátor – atribut ID.

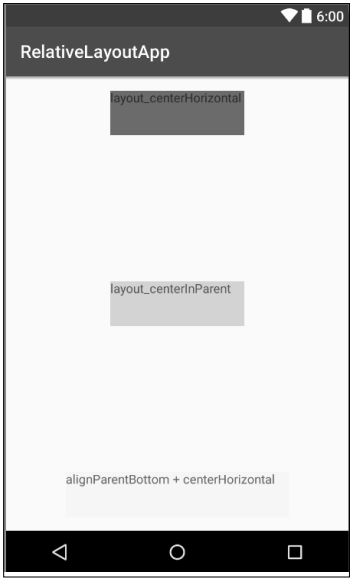
Příklady možností definování polohy:

- layout_above – vůči prvku se zadaným ID
- layout_alignLeft – vůči prvku se zadaným ID
- layout_alignParentLeft – true/false
- layout_centerHorizontal – true/false
- layout_toLeftOf – vůči prvku se zadaným ID

Prvky jsou implicitně umísťovány do levého horního rohu.

V ilustračním příkladu ukážeme nejprve určení polohy views vůči rodičovskému layoutu.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context="sk.pcrevue.llacko.relativelayoutapp.MainActivity">

  <TextView
    android:layout_width="150dp"
    android:layout_height="50dp"
    android:layout_centerInParent="true"
    android:background="@color/green"
    android:text="layout_centerInParent" />

  <TextView
    android:layout_width="150dp"
    android:layout_height="50dp"
    android:layout_centerHorizontal="true"
    android:background="@color/red"
    android:text="layout_centerHorizontal" />

  <TextView
    android:layout_width="250dp"
    android:layout_height="50dp"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:background="@color/yellow"
    android:text="alignParentBottom + centerHorizontal" />
```



Obrázek 5.3: Relativní pozicování prvků vůči rodičovskému view a relativnímu layoutu

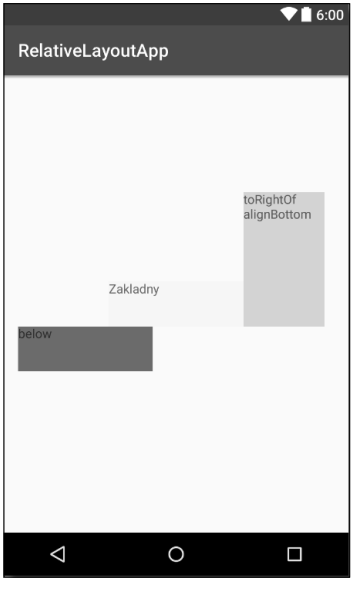
Pomocí atributů

- `android:layout_alignLeft`
- `android:layout_alignTop`
- `android:layout_alignRight`
- `android:layout_alignBottom`
- `android:layout_toLeftOf`
- `android:layout_above`
- `android:layout_toRightOf`
- `android:layout_below`

můžete definovat pozice prvků vůči jiným prvkům. V následujícím příkladu jsme jeden prvek ukotvili vůči rodičovskému `view` a druhé dva prvky jsou ukotveny relativně vůči tomuto prvku.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context="sk.pcrevue.llacko.relativelayoutapp.MainActivity">

  <TextView
    android:id="@+id/zakladny"
    android:layout_width="150dp"
    android:layout_height="50dp"
```

```
    android:layout_centerInParent="true"
    android:background="@color/yellow"
    android:text="Zakladny"

  <TextView
    android:layout_width="150dp"
    android:layout_height="50dp"
    android:layout_below="@id/zakladny"
    android:background="@color/red"
    android:text="below" />

  <TextView
    android:layout_width="90dp"
    android:layout_height="150dp"
    android:layout_alignBottom="@id/zakladny"
    android:layout_toRightOf="@id/zakladny"
    android:background="@color/green"
    android:text="toRightOf\nalignBottom" />

</RelativeLayout>
```

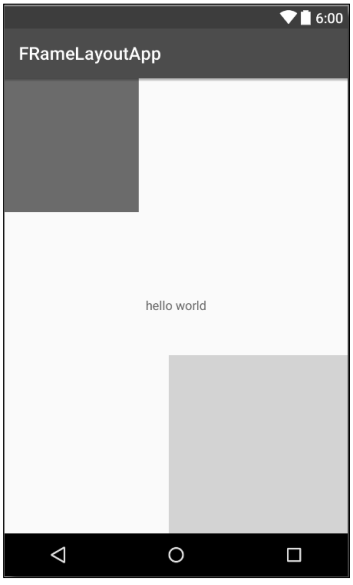


Obrázek 5.4: Relativní pozicování prvků vůči jiným prvkům

Relativní layouty jsou velmi flexibilní, ale pouze v případě, že definujete mezi prvky jen minimální počet vazeb. V opačném případě se stává návrh nepřehledným a vzniknou takzvané kruhové vazby a rozpory, například mezi definováním rozměrů prvků a okrajů vzhledem k rozměru rodičovského layoutu, který zabírá celou šířku obrazovky. Pokud se třeba součet šířky prvku a jeho levého a pravého okraje nerovná šířce rodičovského kontejneru, nastane problém.

FrameLayout

Tento layout umožňuje zobrazit jen jeden prvek (`view`). Využívá se hlavně k zobrazování dynamicky vytvářených fragmentů pro verze operačního systému starší než 3.0 a komplexnějších prvků, například `ScrollView`. Pokud byste do `FrameLayoutu` vložili více prvků, zobrazí se na tom samém místě, přičemž budou zarovnané podle levého horního rohu. Umístění prvků je možné ovlivňovat pomocí atributu `android:layout_gravity`.



Obrázek 5.5: Příklad rozmístění prvků s využitím kontejneru FrameLayout

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="sk.pcrevue.llacko.framelayoutapp.MainActivity">

  <View
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:background="@color/red" />

  <View
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_gravity="bottom|right"
    android:background="@color/green" />

  <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
```

```
    android:text="@string/hello_world" />

</FrameLayout>
```

TableLayout

Do verze API 14 (Android 4.0) se na organizování prvků uživatelského rozhraní do formy tabulek, tedy řádků a sloupců, využíval `TableLayout`. Ke konstrukci řádků tabulky se využívá kontejner `TableRow`. Při vytváření složitějších tabulek oceníte možnost roztáhnout pole přes několik buněk tabulky. Od verze 4.0 se mnohem více využívá intuitivní kontejner `GridLayout`.

V příkladu je tabulka denních maximálních a minimálních teplot. Kvůli zkrácení výpisu jsou odstraněné identifikátory jednotlivých buněk tabulky, text je vložen přímo, a nikoliv pomocí zdrojů jako v reálných aplikacích a pro každý řádek je uvedena jen jedna hodnota.

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:collapseColumns=""
  android:shrinkColumns="*"
  android:stretchColumns="*"
  tools:context="sk.pcrevue.llacko.tablelayoutapp.MainActivity">

  <TableRow
    android:id="@+id/tableRowNadpis"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_horizontal" >

    <TextView
      android:id="@+id/textViewNadpis"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:gravity="center"
      android:padding="8dp"
      android:text="Denné maximá a minimá teplôt"
      android:textSize="13dp"
      android:textStyle="bold">
    </TextView>
  </TableRow>

  <TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
      android:text="" >
    </TextView>

    <TextView
      android:gravity="center"
```

```
        android:text="1.5">
      </TextView>

    ...
    </TableRow>

    <TableRow
      android:id="@+id/tableRowHttp"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" >

      <TextView
        android:id="@+id/textView10"
        android:text="MIN"
        android:textStyle="bold" >
      </TextView>

      <TextView
        android:id="@+id/textView11"
        android:gravity="center_horizontal"
        android:text="12" >
      </TextView>

    ...
    </TableRow>

    <TableRow
      android:id="@+id/tableRowHttps"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" >

      <TextView
        android:id="@+id/textView20"
        android:text="MAX"
        android:textStyle="bold" >
      </TextView>

      <TextView
        android:id="@+id/textView21"
        android:gravity="center_horizontal"
        android:text="26" >
      </TextView>

    ...
    </TableRow>

  </TableLayout >
```