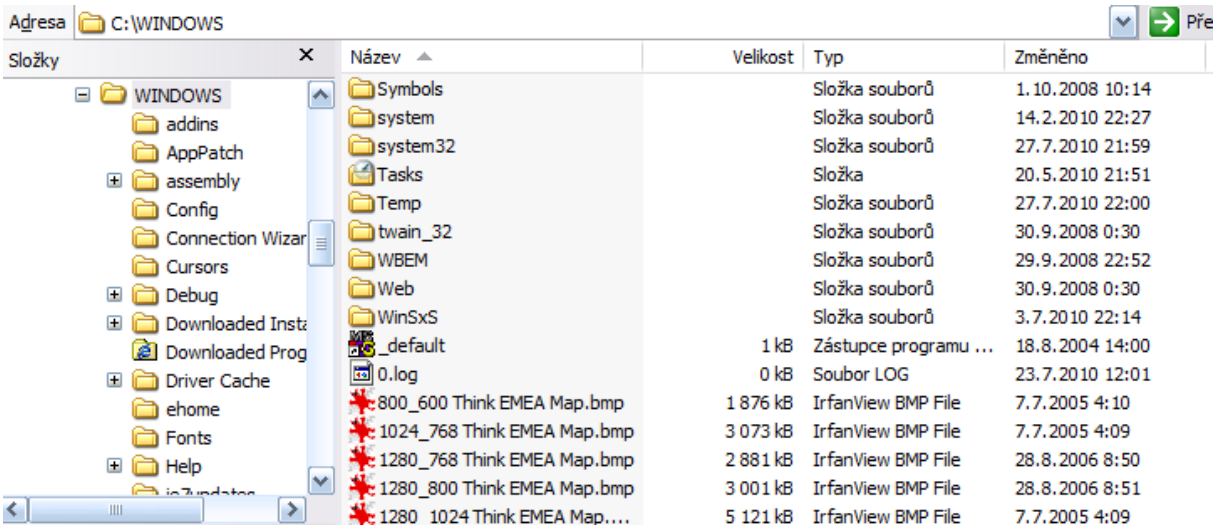
Delphi – lekce 5

TListView a TTreeView

Pomalu přecházíme na složitější komponenty a na složitější příklad (uvedeme jen důležité komponenty, tj. ne všechny). Uvedené dvě komponenty jsou velmi užitečné a často používané. Typické použití je přímo v průzkumníku Windows.



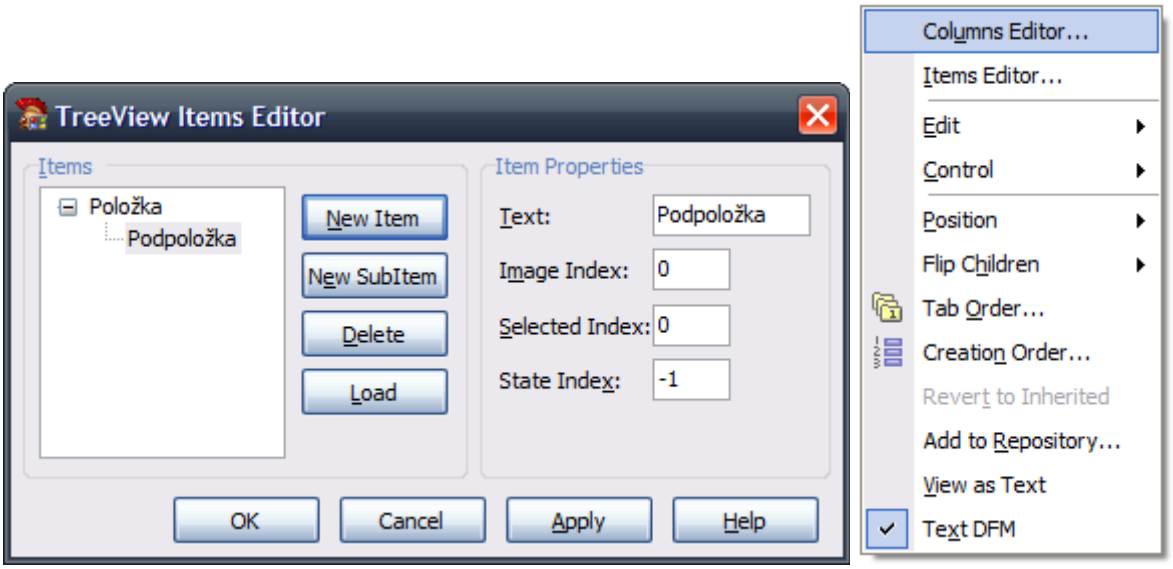
TTreeView je ten strom vlevo (vzhled se dá upravovat) a TListView je ten seznam vpravo (včetně volitelných módů zobrazení jako ikony).

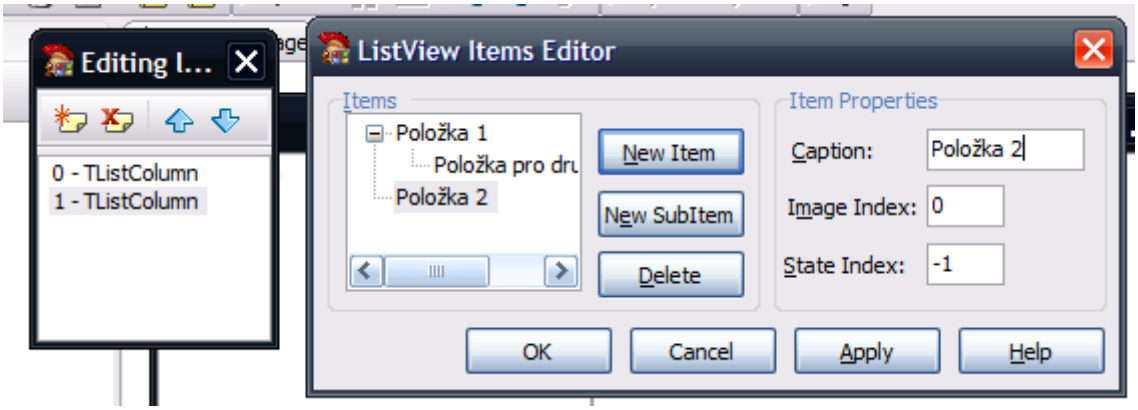
TTreeView se používá k zobrazení stromové struktury (obsah disku, vztahy nadřízený podřízený, položky a podpoložky atd.), kdežto TListView k zobrazení lineárnímu seznamu (výpis adresáře, výpis položek objednávky atd.). Jak je vidět na obrázku, často se používají v kombinaci, kdy ve stromě vybereme nadřízený objekt a v listu pak jeho položky.

Obě komponenty se často spojují s TImageList (z lekce 4) pro udržování kolekce ikon (v případě TListView i ikon různých velikostí) zobrazovaných u jednotlivých položek (na obrázku např. ikona složky ve stromě).

Obě komponenty mají vlastní editory položek, kde lze definovat položky, ale častěji se používá plnění daty až za běhu (např. obsahem disku) a v IDE se definují jen záhlaví sloupců (v ukázce např. Název, Velikost).

Editor položek pro strom se spouští dvojklikem nebo výběrem z popmenu na komponentě. U seznamu se dá spustit editor sloupců a editor položek.





Důležité vlastnosti TListView

Důležitou vlastností je **ViewStyle**, který určuje mód zobrazení:

- vsIcon – zobrazení v módu ikon
- vsList –jednoduchý seznam
- vsSmallIcon – podobně jako vsIcon, jen malé ikony
- vsReport – nejpoužívanější mód, seznam se sloupci

Propojení s TImageList zajišťují property **SmallImages, LargeImages a StateImages**, kdy nejdůležitější je SmallImages, které jsou použity ve všech módech kromě vsIcon.

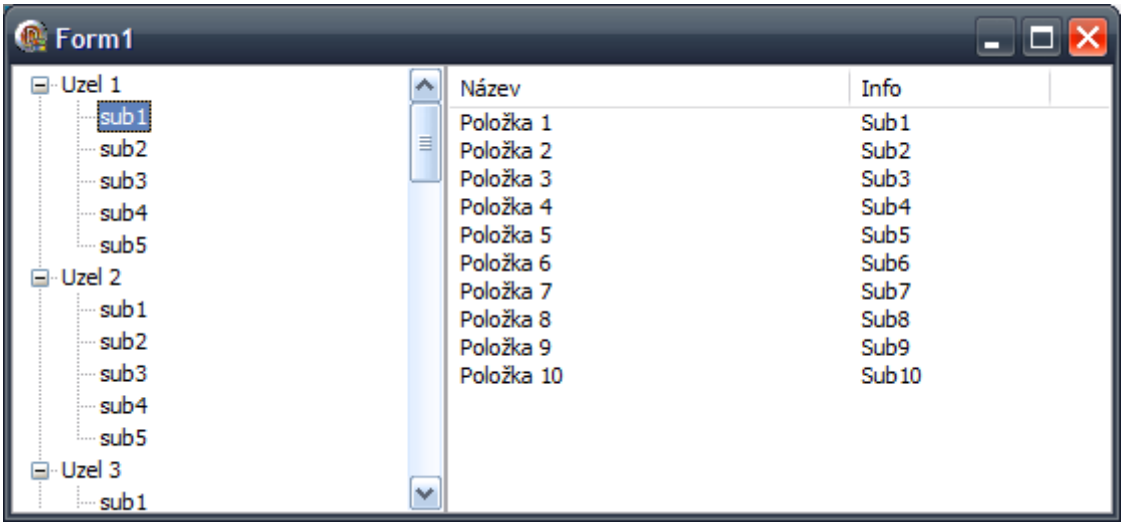
Důležitou vlastností je **RowSelect**, která určuje zda výběr bude celý řádek nebo jen položka. Ale nejdůležitější vlastností je **Items**, která udržuje seznam položek (viz část o editorech položek).

Za běhu lze přidat položky např. takto:

```
procedure TForm1.FormCreate(Sender: TObject);
var
  lvItem: TListItem; // položka TListView
  i: Integer;
  tvItem: TTreeNode;  // položka TTreeView
  j: Integer;
begin
  lv1.Items.BeginUpdate;
  for i := 1 to 10 do //10 polozek do TListView
  begin
    lvItem := lv1.Items.Add;
    lvItem.Caption := 'Položka '+ IntToStr(i);
    lvItem.SubItems.Add('Sub'+IntToStr(i)); // druhý sloupec
  end;
  lv1.Items.EndUpdate;

  tv1.Items.BeginUpdate;
  for i := 1 to 10 do // 10 položek první úrovně
  begin
    tvItem := tv1.Items.Add(nil, 'Uzel '+IntToStr(i));
    for j := 1 to 5 do // pridame druhou uroven – 5 položek
        tv1.Items.AddChild(tvItem, 'sub'+ IntToStr(j));
    tvItem.Expanded := true; // rozbal uzel
  end;
  tv1.Items.EndUpdate;
end;
```

Výsledek běhu je následující:



Volání BeginUpdate / EndUpdate není povinné, ale v případě aktualizace více položek to výrazně urychluje vykonání, jelikož to zakáže překreslování a vyvolávání různých událostí. Ideálně by volání mělo být chráněno try .. finally, tj.

```
BeginUpdate
try
…
finally
   EndUpdate
end
```
Protože v případě nějaké chyby by se nepovolilo překreslování.

Praktický příklad

Zkusíme napodobit průzkumníka, tj. v našem programu bude vlevo TTreeView a vpravo TListView. Naplnění stromu ukáži, zobrazení položek podle vybrané položky bude domácí úkol.

Program je v příloze. Zde klíčový kód.

Po startu se strom naplnění adresáři od konstanty csStart. Ve skutečném projektu se jednotlivé úrovně načtou při rozbalení, tady to načteme najednou – i když to bude trvat déle.

```
procedure TForm1.FormCreate(Sender: TObject);
var
  lvItem: TListItem;
  i: Integer;
  tvItem: TTreeNode;
  j: Integer;
const
  csStart = 'c:\Windows\System32';
begin
  tv1.Items.BeginUpdate;
//  mFillTreeLevel(nil, 'c:');
  mFillTreeLevel(tv1.Items.Add(nil, csStart), csStart );
  tv1.Items.EndUpdate;
end;
```

Klíčová metoda na načtení jedné úrovně. Jedná se o rekurzívní metodu – volá sama sebe s jinými parametry postupně, tak jak prochází diskem. Kombinace FindFirst, FindNext a FindClose se použije při čtení adresáře.

```
procedure TForm1.mFillTreeLevel(oParentNode: TTreeNode; const sPath:
string);
var
  sr: TSearchRec;
  oItem: TTreeNode;

begin
  if FindFirst(sPath+'\'+'*.*', faAnyFile, sr) = 0 then
  begin
    repeat
      if (sr.Attr and faDirectory) > 0 then
      begin
```

```
        if (sr.name <> '.') and (sr.Name <> '..') then
        begin
          oItem := tv1.Items.AddChild(oParentNode, sr.Name);
          mFillTreeLevel(oItem, sPath + '\'+sr.Name );
        end;
      end;
    until FindNext(sr) <> 0;
  end;
  FindClose(sr);
end;
```

Nápověda pro domácí úkol:

- Je třeba obsloužit OnChange stromu a tam vždy volat plnění ListView.
- ListView se vyprázdní voláním Items.Clear (za BeginUpdate!)
- Program lze rozšířit libovolně (ikony, lepší detaily souborů…)