Zpravodaj moderníProgramování 3/2013: Dynamické vytváření XML

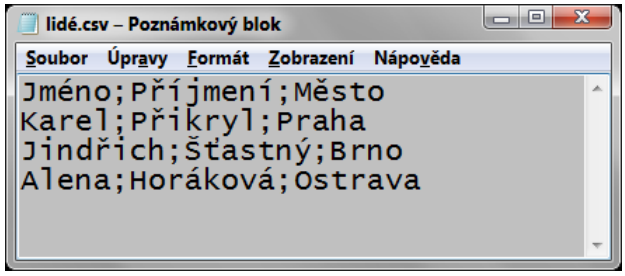
Obtížnost: středně pokročilí

V posledním čísle Zpravodaje jsme se seznámili s ukládáním strukturovaných textových dat v souborech XML. Seznámili jsme se rovněž s vytvářením XML souboru pomocí tříd technologie LINQ to XML - `XDocument`, `XElement`, `XDeclaration`, `XAttribute`. Slíbil jsem pokračování tématu v dalším, tj. tomto čísle.

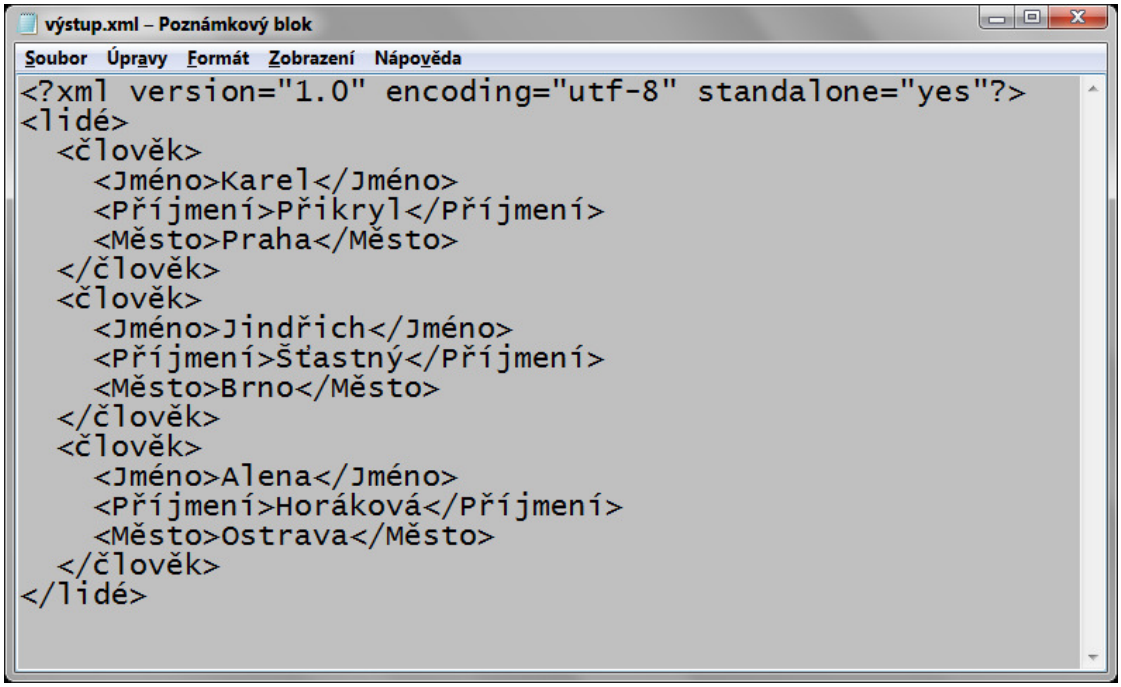
Posledně jsme vytvářeli XML soubor, jehož data jsme znali v době psaní programu. To samozřejmě není typická situace, smyslem ale bylo začít na jednoduchém příkladu. Nyní se podíváme na typičtější situaci, kdy aplikace získá data za běhu.

Zadání

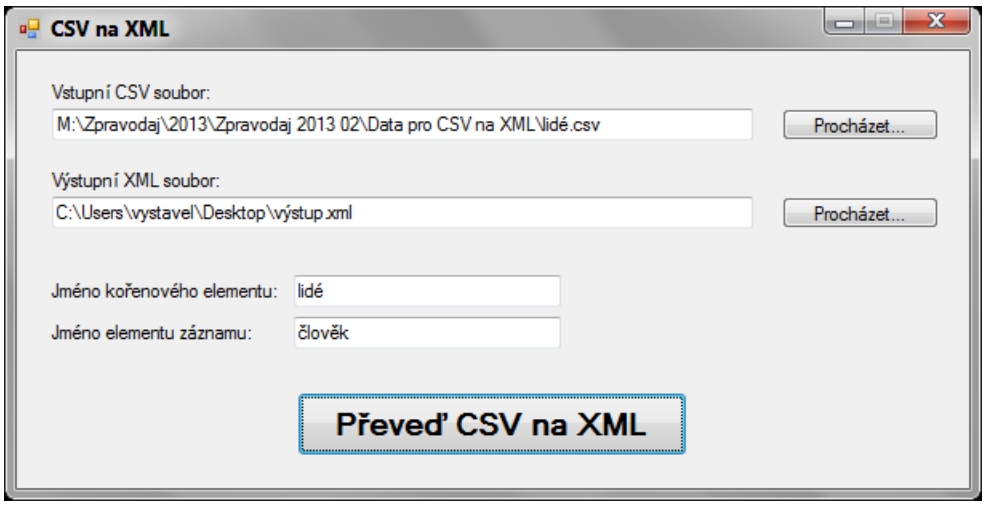
Připravíme program, který data ze souboru CSV přetransformuje do podoby XML. Vstupní soubor bude mít hlavičkový řádek s názvy sloupců, sloupců může být libovolný počet. Soubor může vypadat např. takto:



Náš program ho převede na XML:



Názvy elementů obsahující data ze sloupců CSV souboru program převezme z hlavičkového řádku. Název elementu, který zapouzdří celý záznam (řádek), stejně jako název kořenového elementu ale v CSV datech není, musí je tedy zadat uživatel, z čehož vyplývá uživatelské rozhraní:



Program

Podívejme se rovnou na výpis, hlavní místa si poté vysvětlíme.

```
using System;
using System.Text;
using System.Windows.Forms;
using System.IO; // přidáno
using System.Xml.Linq; // přidáno

namespace CSVnaXML
{
  public partial class oknoProgramu : Form
  {
    public oknoProgramu()
    {
      InitializeComponent();
    }

    private void tlačítkoProcházetVstupní_Click(object sender, EventArgs e)
    {
      if (oknoOtevřeníSouboru.ShowDialog() == DialogResult.OK)
        poleVstupníCSVsoubor.Text = oknoOtevřeníSouboru.FileName;
    }

    private void tlačítkoProcházetVýstupní_Click(object sender, EventArgs e)
    {
      if (oknoUloženíSouboru.ShowDialog() == DialogResult.OK)
        poleVýstupníXMLsoubor.Text = oknoUloženíSouboru.FileName;
    }
```

```
    private void tlačítkoPřeveď_Click(object sender, EventArgs e)
    {
      // Neřeší se žádné chybové stavy (přístup k souborům, chyba
      //   v názvech elementů, chybná struktura vstupního souboru ap.)

      const char oddělovač = ';';

      // Připrav XML dokument
      XDocument xml = new XDocument(
        new XDeclaration("1.0", "utf-8", "yes"),
        new XElement(poleJménoKořenovéhoElementu.Text));

      // Otevři CSV soubor (předp. implic. CP Windows)
      StreamReader vstupníSoubor =
        new StreamReader(poleVstupníCSVsoubor.Text, Encoding.Default);

      // Přečti hlavičku
      string hlavička = vstupníSoubor.ReadLine();
      string[] názvyElementů = hlavička.Split(oddělovač);

      // Čti CSV záznam po záznamu
      string řádek;
      while ((řádek = vstupníSoubor.ReadLine()) != null)
      {
        string[] hodnoty = řádek.Split(oddělovač);
        XElement záznam = new XElement(poleJménoElementuZáznamu.Text);
        for (int index = 0; index < názvyElementů.Length; index++)
        {
          string název = názvyElementů[index];
          string hodnota = hodnoty[index];
          XElement vkládanýElement = new XElement(název, hodnota);
          záznam.Add(vkládanýElement);
        }
        xml.Root.Add(záznam);
      }

      // Zavři vstupní soubor a zapiš XML
      vstupníSoubor.Close();
      xml.Save(poleVýstupníXMLsoubor.Text);
      MessageBox.Show("Hotovo");
    }
  }
}
```

Poznámky k programu

- Práce se soubory včetně použití standardních dialogových oken je vysvětlena v páté kapitole učebnice pro středně pokročilé;
- Práce se soubory CSV je vysvětlena v sedmé kapitole učebnice pro středně pokročilé. Především jde o použití metody `Split`, která řetězec rozseká na pole dílčích řetězců (podle oddělovače, zde středníku);

- Pokud známe obsah (hodnotu) elementu v okamžiku jeho vytváření, lze použít dvouparametrický konstruktor `XElement`. V našem programu jde o elementy `Jméno`, `Příjmení` a `Město`, které se vkládají řádkem:

```
XElement vkládanýElement = new XElement(název, hodnota);
```

- U kořenového elementu `lidé` a elementu `člověk` ale jejich obsah dopředu neznáme. Vytváříme je tedy z počátku prázdné pomocí jednoparametrického konstruktoru:

```
new XElement(poleJménoKořenovéhoElementu.Text)
```

resp.

```
XElement záznam = new XElement(poleJménoElementuZáznamu.Text);
```

Tyto prázdné elementy poté plníme přidáváním vnořených elementů. K tomu slouží metoda `Add`;

- Instance třídy `XDocument` mají vlastnost `Root`, která odkazuje na kořenový element dokumentu. Díky tomu nemusíme kořenový element ukládat do nějaké proměnné.

Závěr

Doufám, že jste nyní v práci s XML soubory postoupili zase o kousek dál. Zbývá ještě naučit se v nich vyhledávat pomocí LINQ dotazů. To ale až někdy jindy.

Radek Vystavěl, 23. dubna 2013

Pokud Vám Zpravodaje moderníProgramování připadají užitečné, doporučte jejich odběr svým známým. Mohou se přihlásit na webu www.moderniProgramovani.cz.