3 31. března – Windows Forms Applications a pár drobností

Ačkoliv jsem se domníval, že postačí studijní materiál, nakonec jsem si řekl, že přidám ještě pár drobností. Začněte však, prosím, prostudováním materiálu, následující text si přečtěte až pak.

První v materiálu chybějící informací je, že řídicí prvky můžete přidávat na formulář při běhu programu. Ony i ty ovládací prvky jsou jen instance nějaké třídy (a to té, podle které s nimi pracujeme, když jim nastavujeme atributy). Tedy tlačítko je instance třídy `Button`. Chceme-li se vytvořit nové tlačítko, zavoláme konstruktor této třídy. Tím vznikne prázdné tlačítko. Následně tlačítku vyplníme vhodné atributy (ty si prohlédněte samostatně) a především musíme tlačítko *umístit na formulář*. Věc si demonstrujeme následujícím příkladem. Naklikejte si formulář s tlačítkem, při jehož stisknutí přidáme na formulář další tlačítko, kterému nastavíme týž ovladač události `Click`:

```
int pocet=0;

private void button1_Click(object sender, EventArgs e)
{   Button b = new Button();//Volame konstruktor
    b.Text = "Cudlik";//Nastavujeme atributy
    b.Location = new Point(70, 70+20*i++);//aby se knofliky nezakryvaly
    b.Size = new Size(50, 30);
    b.Click += button1_Click;
    this.Controls.Add(b);//pridavame na formular. Tohle je dulezite!
}
```

Atribut `pocet` definujeme, aby námi tvořená tlačítka nešla přes sebe. Takto se sice mírně překrývat nejspíše budou, ale současně bude vidět, kolik jich je. Ostatní komponenty můžete za běhu tvořit podobným způsobem.

Druhou věcí je, že materiál poněkud odbývá kreslení. Samozřejmě můžete kreslit tak, jak je tam popsáno. Ovšem ve chvíli, kdy okno vystrčíte přes okraj obrazovky a následně zatáhnete zpět, Windows zapomenou údaje o části okna, která nebyla vidět a nechají si buďto celé okno, nebo alespoň příslušnou část překreslit. A po překreslení se vaše malůvky již neobjeví. Chcete-li tedy, aby vaše umělecká díla namalovaná na formulář po překreslení nemizela, musíte je nechat po každém překreslení prvku, na kterém jsou, nakreslit znovu. Uděláme to tak, že dotyčnému prvku (který jsme pomalovali) přidáme další ovladač události `Paint`:

```
void pomaluj(object sender, PaintEventArgs e)
{       //Zde budeme delat cmariky, cmariky...
}
...
        prvek_k_pomalovani.Paint+=pomaluj;
```

Takto dosáhneme toho, že se funkce `pomaluj` zavolá při každém překreslení pomalovaného objektu.

Třetí a vlastně i čtvrtou věcí, kterou si ještě rozebereme, je mluvený výstup a s tím související přidávání referencí k projektu. Motivováni mluveným výstupem se pustíme do přidávání referencí:

V programu chceme (z důvodu, který bude brzy jasný) použít `System.Speech.Synthesis`. Zkusíte-li najít tutorialy po webu, uvidíte, že tento namespace by měl být dostupný již od .NET Frameworku 4.0 (přičemž vy užíváte verzi alespoň 4.6). Přesto když se pokusíte říct:
`using System.Speech.Synthesis;`
neprojde to. Překladač totiž ke každému projektu přikompilovává standardní knihovny pro daný typ projektu typické. Windows Forms Applications (které jsou tento týden předmětem našeho zájmu) však obvykle nepoužívají hlasový výstup. Aby aplikace nebyla příliš velká, dotyčná knihovna se k ní nepřikompilovává. Chceme-li tento namespace použít, musíme *přidat příslušnou referenci.* To uděláme takto: `Projekt` → `Add Reference...`, ve vyskočivším okně vpravo nahoře dáme najít `Speech` a všimneme si, že z původní změti možných referencí nám zbyla jediná v tuto chvíli nevybraná reference: `System.Speech`, kterou zaklineme (a výběr potvrdíme). Následně si všimněte v *Project manageru*, že v sekci `References` tuto přidanou referenci vidíte. Nyní můžete do zdrojového textu přidat `using System.Speech.Synthesis;` a program půjde přeložit.

Asi nám už je všem jasné, že minulý rituál jsme dělali proto, že abychom mohli aplikaci obohatit o mluvený výstup. Tento mluvený výstup ostatně můžete používat nejen ve formulářových aplikacích (pro které navíc není úplně typický), ale jinam do výkladu se nehodí. Mluvený výstup se ovládá velmi jednoduše. Stručně řečeno, postavíme si objekt typu `SpeechSynthesizer`, nastavíme mu atributy a pak už jen té správné metodě strkáme stringy, které má přečíst. Opět si vše ukážeme snadným příkladem. Využijeme minulý příklad, kde máme tlačítko jménem `button1` a definujeme mu ovladač události `Click` takto:

```
using System.Speech.Synthesis;
...//zapomete-li pridat tento using, budete vsude psat\\
   //System.Speech.Synthesis.SpeechSynthesizer

        private void button1_Click(object sender, EventArgs e)
        {   SpeechSynthesizer s = new SpeechSynthesizer();
            s.SetOutputToDefaultAudioDevice();
            //s.SelectVoice()
            s.Speak("Nazdar!");
        }
```

Máte-li dojem, že syntetizátor na vás mluví nějak divně - nejspíše jako Američan z Vysočan, ten dojem může být správný. Jaký hlas na vás mluví, totiž závisí na tom, jak máte nastavené jazyky v nastavení systému. Máte-li Windows lokalizované do angličtiny, čte se text podle anglických pravidel. Máte-li Windows nastavené na češtinu, předčítá vám Jakub. Anglicky vám mohou předčítat:

James, Catherin, Richard, Linda, George, Hazel, Susan, Ravi, Heera, Sean, David, Mark nebo Zira. Kdo má předčítat, si právě můžete zkusit ponastavovat taháním za vhodné metody a atributy. Jednou takovou velmi vhodnou metodou je ve zdrojáku zakomentovaná metoda `SelectVoice`.

Cvičení: Chybí-li vám nafilmované přednášky z Programování II, napište si Windows Forms Application, která načte zadaný soubor, rozdělí ho na jednotlivé odstavce, které bude postupně zobrazovat v `Labelu`. Zobrazovaný odstavec nechte vždycky přečíst a po přečtení pokračujte následujícím odstavcem. Program vybavte ovládacími prvky dle vlastního uvážení (odstavec dopředu, odstavec dozadu, na start – kde se zobrazí odkaz na nějaké sekce či dokonce jednotlivé odstavce) a uvidíte prezentaci ještě o něco lepší, nežli by byla nafilmovaná přednáška. :-)