Zpravodaj moderníProgramování 2/2015: Formální typové konverze

Obtížnost: pokročilí

Navazuji na předchozí Zpravodaj, který řešil skutečné typové konverze. Konverze formální se liší v tom, že **nedochází k žádnému přepočtu** (transformaci) hodnoty. Konverzí formální pouze říkáme, že **chceme s danou hodnotou pracovat v jiném datovém typu**.

Implicitní konverze

K formálním implicitním konverzím, tj. formálním konverzím bez našeho přičinění, dochází při přiřazování objektu do proměnné typu předka.

Například jakýkoli ovládací prvek Windows Forms lze přiřadit do proměnné typu `Control`. To využíváme, pokud např. chceme jednotným způsobem zpracovat všechny ovládací prvky v okně, třeba je všechny posunout ap. Při tomto jednotném zpracování máme k dispozici vše, co je definováno ve třídě `Control`, což je docela dost - písma, barvy, poloha, velikost...

Podobně lze **cokoli** (jakýkoli objekt, čísla, řetězce...) v .NETu přiřadit do proměnné typu `object` (odpovídající třídě `System.Object`). Pak ale s hodnotou pracujeme jako s hodnotou třídy `Object`, to znamená, že ji můžeme tak leda převést na řetězec voláním `ToString`.

Obojí je rozebráno ve žluté a zelené učebnici, přičemž zvláště druhý případ je extrémně důležitý, neboť se využívá u velmi mnoha obecně fungujících komponent a bude se mu věnovat zbytek výkladu tohoto Zpravodaje.

Konverze z typu object

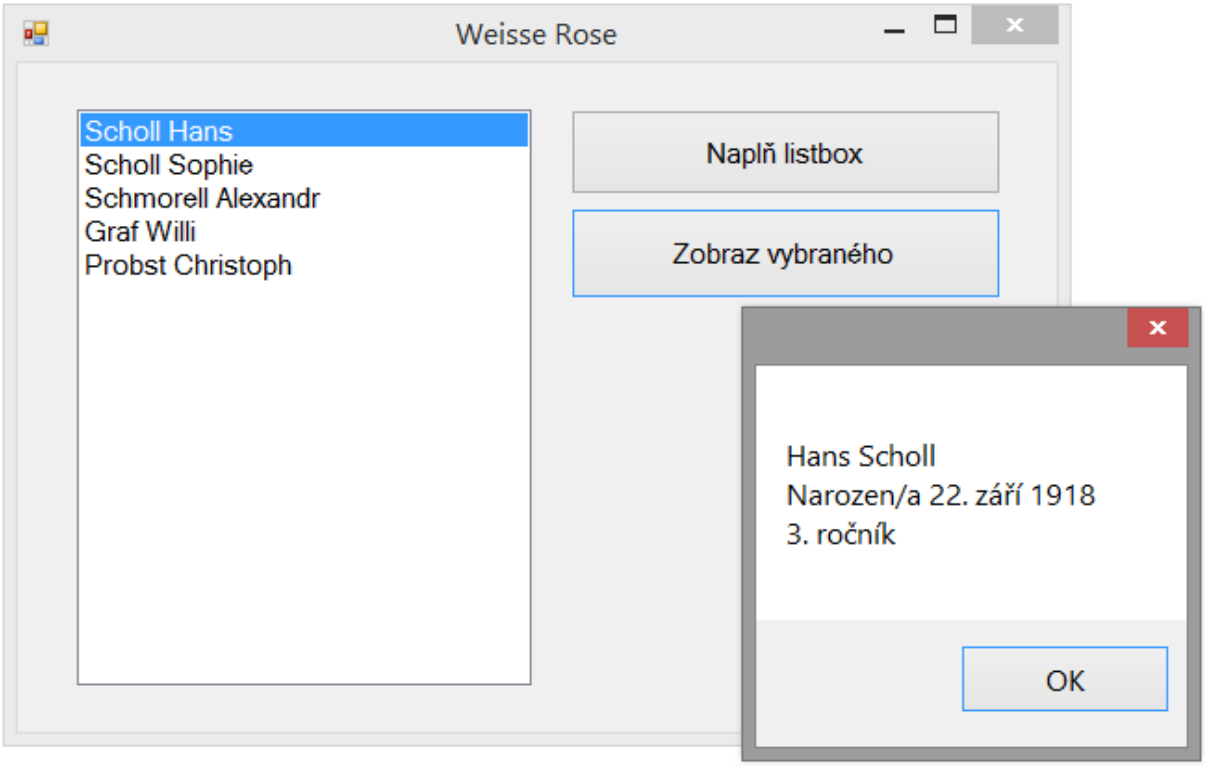
Uvažujme listbox. Do listboxu můžete strčit cokoli - řetězce, čísla, jakékoli objekty atd. Vložené hodnoty se zobrazují pomocí své metody `ToString`, příp. nastavením vlastnosti `DisplayMember`. Do listboxu můžete strčit cokoli, neboť vnitřně se každá hodnota uloží jako hodnota typu `object`.

Druhou stránkou této univerzality ale je vytahování vložených hodnot z listboxu. Na tyto hodnoty se dostaneme buď přes `listbox.Items[index]` nebo `listbox.SelectedItem`, kde `listbox` je název instance ovládacího prvku `ListBox`. Vzhledem k výše zmíněné univerzalitě ale **vytaženou hodnotu dostaneme v nejobecnějším typu object**, což znamená, že s ní prakticky nic nemůžeme dělat!

Abychom s vytaženou hodnotou mohli nějak rozumně pracovat, potřebujeme ji mít zase v původním datovém typu. Když jsme tam strčili řetězec, potřebujeme zase zpátky získat řetězec, když jsme tam strčili číslo, potřebujeme zase zpátky získat číslo, když jsme tam strčili objekt třídy `Student`, potřebujme zase zpátky objekt třídy `Student`. Jinými slovy, vytaženou hodnotu potřebujeme **formálně konvertovat** na původní typ. Jak naznačuje slovo, jedná se pouze o splnění formálních požadavků jazyka C# a jeho překladače, nikoli o nějakou transformaci (změnu) dat samotných - ta zůstávají nedotčena.

Příklad

Problematiku formálních konverzí si ukážeme na praktickém příkladu. Do listboxu budeme vkládat instance vlastní třídy `Student`. Po stisknutí tlačítka se zobrazí detail vybraného studenta.



Příprava třídy Student

Pro objekty studentů potřebujeme do projektu přidat třídu `Student`:

```
class Student
{
    public string Jméno { get; set; }
    public string Příjmení { get; set; }
    public int Ročník { get; set;}
    public DateTime DatumNarození { get; set; }

    public override string ToString()
    {
        return Příjmení + " " + Jméno;
    }
}
```

Plnění listboxu

Vytvoříme několik instancí třídy `Student` a vložíme je do kolekce `Items` listboxu, abychom je zobrazili:

```
private void tlačítkoNaplňListbox_Click(object sender, EventArgs e)
{
    // Mnichov, LMU, 1942/43, Weisse Rose

    // Vytvoření instancí
    Student hans = new Student()
    {
        Jméno = "Hans",
        Příjmení = "Scholl",
        DatumNarození = new DateTime(1918, 9, 22),
        Ročník = 3
    };

    Student sophie = new Student()
    {
        Jméno = "Sophie",
        Příjmení = "Scholl",
        DatumNarození = new DateTime(1921, 5, 9),
        Ročník = 1
    };

    Student alex = new Student()
    {
        Jméno = "Alexandr",
        Příjmení = "Schmorell",
        DatumNarození = new DateTime(1917, 9, 16),
        Ročník = 3
    };

    Student willi = new Student()
    {
        Jméno = "Willi",
        Příjmení = "Graf",
        DatumNarození = new DateTime(1918, 1, 2),
        Ročník = 2
    };

    Student chris = new Student()
    {
        Jméno = "Christoph",
        Příjmení = "Probst",
        DatumNarození = new DateTime(1918, 11, 6),
        Ročník = 3
    };

    // Vložení instancí do listboxu
    listBoxStudenti.Items.Clear();
    listBoxStudenti.Items.Add(hans);
    listBoxStudenti.Items.Add(sophie);
    listBoxStudenti.Items.Add(alex);
    listBoxStudenti.Items.Add(willi);
    listBoxStudenti.Items.Add(chris);
}
```

Jádro úlohy – práce s vybraným záznamem

Nyní se dostáváme k tomu hlavnímu – **abychom mohli pracovat s vybraným záznamem, potřebujeme provést formální typovou konverzi na typ, který jsme do listboxu dříve vložili**:

```
private void tlačítkoZobrazVybraného_Click(object sender, EventArgs e)
{
    // Nelze dále, pokud není nikdo vybrán
    if (listBoxStudenti.SelectedIndex < 0)
        return;
    // První pokus o práci s vybranou položkou
    object vybranýZáznam = listBoxStudenti.SelectedItem;
    // MessageBox.Show(vybranýZáznam.Příjmení);
    // Nelze! V typu object neexistuje žádné Příjmení...
    // Nutná formální typová konverze na původní typ
    // Použijeme unární operátor typové konverze
    //   (název typu v závorkách)
    Student vybranýStudent = (Student)listBoxStudenti.SelectedItem;
    MessageBox.Show(
        vybranýStudent.Jméno + " " + vybranýStudent.Příjmení +
        Environment.NewLine +
        "Narozen/a " + vybranýStudent.DatumNarození.ToLongDateString() +
        Environment.NewLine +
        vybranýStudent.Ročník.ToString() + ". ročník"
        );
}
```

Závěr

Listbox je univerzální komponenta schopná uživateli zobrazovat cokoli. Této univerzality se dosahuje použitím obecného typu `object`. Důsledkem je nutnost formálních typových konverzí při další práci s daty listboxu.

Možná si říkáte, proč je to nutné. Proč překladač neví, že jsme tam dali studenty, a ne brambory? No, neví, je to tak. Vědět musíme my, konverze je naše riziko, a proto ji musíme výslovně zapsat. Když do listboxu strčíme studenty a budeme je posléze chtít konvertovat na brambory, program spadne. Student není ertepla!

Silně typový jazyk C# nám poskytuje komfort našeptávání Visual Studia, před lecčím nás chrání atd. Daní za to jsou formální typové konverze z typu `object`.

Setkáte se s nimi na mnoha místech, proto je důležité jim porozumět:

- Všechny seznamové prvky ve Windows Forms, WPF, Windows Store/Phone;
- Zpracování parametru `sender` u událostí;
- Prostředky řízení stavu webové aplikace ASP.NET jako jsou stav stránky, strav relace, keš;
- Čtení dat z databáze v připojeném režimu ADO.NET;
- Čtení objektů ve formátu JSON a tak dále.

Radek Vystavěl, 9. února 2015

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