Jazyk C#

O půl tónu výš

Po definitivním rozchodu s Javou firmy Sun Microsystems přichází Microsoft s návrhem nového programovacího jazyka. Jeho název C#, čtený jako C sharp, snad v angličtině může znamenat cosi jako "chytré C", autoři si však přitom neodpustili slovní hříčku – anglosaský hudebník si pod tímto označením představí jediné: tón cis.

V současné době je k dispozici první, ještě ne zcela úplná referenční příručka C#. Už z ní je však zřejmé, že nový jazyk obsahuje některé velice zajímavé rysy a vlastnosti, a proto se zde pokusím alespoň o krátkou exkurzi do jeho syntaxe a sémantiky. Znovu připomínám, že následující odstavce vznikly na základě velice rané verze referenční příručky, a tak se některé detaily ještě mohou v budoucnosti změnit.

Jako Java…

Při úplně prvním pohledu nás napadne, že jazyk se značně podobá Javě. Například tradiční "pozdrav světu" vypadá v C# takto:

```
using System;
class Hello {
  static void Main() {
        Console.WriteLine("Hello, world");
  }
}
```

C# je stejně jako Java založen na syntaxi jazyka C; nalezneme zde obvyklé třídy, rozhraní a metody, správa paměti je stejně jako v Javě založena na garbage collectoru. Tento fakt vedl k mnohým komentářům o tom, že C# je tupá a nekompatibilní nápodoba Javy, kterou Microsoft vymyslel jenom proto, aby odvrátil zájem vývojářů od Sunu. Pokud se ovšem do normy začteme podrobněji, zjistíme dosti podstatné rozdíly, které většinou vyznívají ve prospěch C#.

Hodnoty a odkazy

C# rozlišuje dvě základní skupiny typů – typy hodnotové a referenční. Proměnné hodnotových typů přímo obsahují svá data, zatímco proměnné odkazových typů obsahují odkaz na data uložená jinde. Odkazové typy jsou tedy především třídy a rozhraní a dále také pole a tzv. zástupci (delegate), což jsou víceméně odkazy na metody. Mezi hodnotové typy patří základní typy, záznamy (struct) a výčty (enum).

Rozdíl mezi odkazovými a hodnotovými typy ilustruje následující příklad:

```
class Class1 {
  public int Value = 0;
}
  ...
  int val1 = 0;
  int val2 = val1;
  val2 = 123;
  Class1 ref1 = new Class1();
  Class1 ref2 = ref1;
  ref2.Value = 123;
...
```

Po skončení tohoto prográmku bude val1 obsahovat 0, zatímco v ref1.Value bude 123.

Třídy a záznamy

C# má jeden velice pragmatický rys – kromě tříd (class), které jsou referenčními typy, obsahuje také záznamy (struct), což jsou typy hodnotové. Záznamy přitom mohou obsahovat metody, dokonce mohou implementovat rozhraní; jediné omezení pro ně spočívá v tom, že se nemohou stát členem typové hierarchie.

C# podporuje přetěžování operátorů (na rozdíl od Javy), a tak je v principu možné nadefinovat například plnohodnotný typ komplexních čísel, jejichž použití bude k nerozeznání od použití

základního typu int. Ve skutečnosti se základní typy liší od takto nadefinovaných nových typů pouze přímou podporou pro překladač, a tedy efektivnějším vygenerovaným kódem.

Sjednocení typů

Významnou vlastností C# je tzv. sjednocení typů. Libovolný typ lze, bez ohledu na to, zda je hodnotový, nebo odkazový, převést na třídu object a zpět:

```
int i = 777;
object o = i;
int j = (int) o;
```

Zpětný převod se pochopitelně provede pouze v případě, že proměnná typu object obsahuje hodnotu (nebo odkaz v případě, že byl přiřazen referenční typ) požadovaného typu (jinak dojde k výjimce).

Převodu hodnotového typu na objekt sémanticky odpovídá vytvoření nové instance třídy, která obsahuje jediný atribut požadovaného typu a přiřazení hodnoty do tohoto atributu. Schopnost pracovat s hodnotovými typy jako s objekty tak vyplňuje mezeru, která existuje ve většině objektově orientovaných jazyků, jako je Java nebo Objective C. Například kontejnerové typy (pole, zřetězené seznamy apod.) stačí implementovat pro instance třídy object, a přitom je lze bez dalšího úsilí používat pro skladování základních typů nebo záznamů.

Zástupci

Datový typ zástupce (delegate) přibližně odpovídá ukazateli na funkci, ovšem místo funkce ukazuje na metodu určité konkrétní instance. Deklarace datového typu zástupce určuje počet a typ parametrů a typ návratové hodnoty:

```
delegate void ZpracujInt(int i);
```

Do proměnné takto definovaného typu ZpracujInt lze přiřadit libovolnou metodu, která má jediný parametr int a nic nevrací. Zástupce tak vlastně představuje zapouzdření volatelné entity, což je mimo jiné ideálním základem pro implementaci zpracování událostí v GUI.

Mimochodem, jazyk Java řeší tuto oblast pomocí poněkud kostrbatého systému vnitřních a anonymních tříd. Zástupci byli jedním z rozšíření J++ Microsoftu oproti Javě, se kterými Sun ostře nesouhlasil, což nakonec vedlo k soudnímu sporu a opuštění Javy Microsoftem. Nevím, mně osobně se zástupci zdají jednodušší a přehlednější.

Vlastnosti a indexy

Vlastnosti (property) představují mechanismus dobře známý z Visual Basicu, který volání metod “maskuje" jako přiřazení atributu. Například

```
public class Button {
  private string caption;
  public string Caption {
        get { return caption; }
        set {     caption = value; Repaint(); }
  }
...
}
```

definuje vlastnost Caption.

Přiřazením

```
Button b = new Button();
b.Caption = "ABC";          // set
string s = b.Caption;       // get
b.Caption += "DEF”;         // get & set
```

se volají příslušné metody get a set.

Jiný mechanismus dokáže zajistit, aby se objekt choval jako “chytré pole":

```
public class List {
  private string[] items;
  public string this[int index] {
        get { return items[index]; }
        set {
              items[index] = value;
              Repaint();
```

```
            }
        }
        ...
    }

    List list;
    list[0] = "hello";
```

Takové řešení je výhodné dokonce i oproti jinak velmi otevřenému systému přetěžování operátorů v C++, kde někdy bývá problém s rozlišením zápisu do “pole" a čtení z “pole".

Nebezpečná zóna

Ačkoliv se C#, stejně jako Java, většinou celkem rozumně tváří jako jazyk poměrně vysoké úrovně, který neumožňuje například přímo pracovat s obsahem paměti, na rozdíl od Javy autory neopustil pragmatický duch a zavedli jakousi “nebezpečnou zónu”, tedy kus kódu, ve kterém existují ukazatelé jako v C a v němž se smí cokoliv – třeba tohle:

```
class Dirty {
  unsafe static void Set(byte[] dst, byte val, int count) {
  fixed (byte *d = dst)
while(count --> 0)
*d++ = val;
  }
}
```

Výhodou je, že takovéto “divoké" části kódu jsou klíčovým slovem unsafe dobře izolovány od ostatního “mírumilovného" okolí.

Další zajímavé vlastnosti

- C# obsahuje víceméně klasický céčkový preprocesor. To je další důkaz zdravého rozumu autorů, protože i když nejsou v C# potřeba hlavičkové soubory pro vyjádření rozhraní mezi moduly, stále je ještě hodně situací, které se bez preprocesoru řeší zbytečně složitě.
- Kromě klasické řídicí struktury, vesměs převzaté z C, je k dispozici příkaz foreach, který docela pěkným způsobem zobecňuje procházení mezi prvky kontejnerů.
- C# se na rozdíl od Javy nebrání dokonce ani zprofanovanému goto.
- Pro synchronizaci procesů lze použít klíčové slovo lock.
- Klíčovými slovy checked a unchecked lze stanovit, zda se má kontrolovat aritmetické přetečení výrazu.
- Autoři nezapomněli ani na výčtové typy (enum), jejichž absence v javovském repertoáru někdy vadí.

Souvislosti

Ačkoliv se o tom referenční příručka příliš nešíří, je evidentní, že jazyk C# je vytvořen tak, aby “zapadl" do celkové strategie Microsoftu, zejména v souvislosti s novou platformou .NET.

A tak je například zřejmé, že rozhraní (interface) v C# přímo korespondují s rozhraními COM, takže každý objekt v C# může být vlastně automaticky COM objektem.

V souvislosti s oznámením C# byl uveden také nový formát CEF, nezávislý na procesoru. Jedná se tedy o jakousi obdobu Java Bytecode, ovšem na rozdíl od něj je tento formát plně použitelný i pro jiné jazyky (Visual Basic, C++, Pascal) a od počátku je určen pro překlad do strojového kódu, nikoliv pro interpretování.

Závěr

Jazyk C# je na jedné straně nepochybně protitahem Microsoftu na nástup Javy. Jeho autoři měli jistě výhodu v tom, že za pět let existence Javy mohli nasbírat praktické zkušenosti s programovacím jazykem takového typu. Při bližším prozkoumání se C# jeví jako jazyk podstatně flexibilnější a pragmatičtější a práce v něm bude s velkou pravděpodobností pohodlnější než v Javě. Připočteme-li úzkou integraci s dominantním objektovým systémem (a tady mám na mysli spíš COM než samotné Windows), je pravděpodobné, že C# má slušnou šanci stát se v dohledné době jedním z nejrozšířenějších programovacích jazyků.

Miroslav Fídler (cxl@volny.cz)

(Ing. Miroslav Fídler je programátorem u firmy Cybex Development, s. r. o.)