

Stručný obsah

Předmluva	19
1. Úvod do jazyka C#	23
2. C# a CLR	33
3. Přehled syntaxe jazyka C#	41
4. Třídy, struktury a objekty.....	65
5. Rozhraní a smluvní vztahy	159
6. Přetěžování operátorů	187
7. Zabezpečení zdrojového kódu proti výskytu výjimek a jejich ošetřování.....	203
8. Práce s řetězci.....	237
9. Pole, kolekce a iterátory	265
10. Delegáty, anonymní funkce a události	301
11. Genericita	327
12. Práce s podprocesy v jazyku C#	379
13. Hledání kanonických tvarů v jazyce C#	445
14. Rozšiřující metody	503
15. Lambda-výrazy	529
16. LINQ: Dotazy integrované do jazyka	553
17. Dynamické typy	587

Obsah

O autorovi.....	17
Odborný korektor.....	17
Poděkování.....	18
Předmluva	19
O této knize.....	20
Kapitola 1	
Úvod do jazyka C#.....	23
Rozdíly mezi jazyky C# a C++.....	23
C#.....	23
C++.....	24
Automatická správa paměti (garbage collector) v CLR.....	25
Příklad programu v jazyce C#.....	26
Přehled nových vlastností C# 2.0.....	27
Přehled nových vlastností C# 3.0.....	28
Přehled nových vlastností C# 4.0.....	30
Shrnutí.....	30
Kapitola 2	
C# a CLR.....	33
Překladač JIT a CLR.....	34
Sestavení a zavaděč sestavení.....	35
Minimalizace množiny stránek aplikace.....	36
Názvy sestavení.....	37
Načítání sestavení.....	37
Metadata.....	38
Kompatibilita napříč jazyky.....	39
Shrnutí.....	40

Kapitola 3

Přehled syntaxe jazyka C# 41

C# je silně typový jazyk	41
Výrazy.....	42
Příkazy a výrazy	44
Typy a proměnné.....	44
Hodnotové typy	46
Referenční typy	49
Výchozí inicializace proměnných.....	50
Implicitně typované lokální proměnné.....	51
Konverze typů.....	53
Operátory as a is	55
Genericita	57
Jmenné prostory.....	58
Deklarace jmenných prostorů	59
Používání jmenných prostorů.....	60
Řízení toku programu	62
if-else, while, do-while a for.....	62
switch.....	62
foreach.....	63
break, continue, goto, return a throw.....	64
Shrnutí	64

Kapitola 4

Třídy, struktury a objekty 65

Definice tříd	67
Složky	68
Konstruktory.....	71
Metody	72
Vlastnosti	74
Zapouzdření	78
Přístupnost.....	82
Rozhraní	84
Dědění	85
Zapečetěné třídy.....	92
Abstraktní třídy.....	93
Vnořené třídy	94
Indexery	97
Částečné třídy	99

Částečné metody.....	100
Statické třídy.....	101
Vyhrazené názvy členů.....	103
Definice hodnotových typů.....	104
Konstruktory.....	105
Význam klíčového slova this	107
Finalizéry.....	109
Rozhraní	110
Anonymní typy	110
Inicializátory objektů.....	113
Balení a vybalování hodnotových typů	116
Kdy dojde k zabalení	120
Efektivita a zmatek.....	122
System.Object	122
Rovnost a její význam	124
Rozhraní IComparable	124
Vytváření objektů	125
Klíčové slovo new	125
Inicializace složek.....	126
Statické konstruktory (konstruktory tříd).....	127
Konstruktor instance a pořadí operací při jejím vytváření.....	130
Destrukce objektů	134
Finalizéry.....	134
Deterministická destrukce	136
Ošetřování výjimek	137
Jednorázové objekty	137
Rozhraní IDisposable.....	137
Klíčové slovo using.....	140
Typy parametrů metod.....	141
Hodnotové parametry.....	142
Parametry předávané odkazem (ref)	142
Výstupní parametry (out)	144
Proměnný počet parametrů (params)	144
Přetěžování metod.....	145
Volitelné parametry	145
Pojmenované parametry.....	146
Dědění a virtuální metody.....	150
Virtuální a abstraktní metody.....	150
Metody s modifikátory override a new	150
Zapečetěné metody (sealed)	152

Několik posledních slov o virtuálních metodách v jazyce C#	153
Dědění, skládání a delegace	153
Volba mezi použitím rozhraní a děděním tříd	153
Delegace a kompozice vs. dědění	155
Shrnutí	157

Kapitola 5

Rozhraní a smluvní vztahy..... 159

Rozhraní definují typy	160
Definice rozhraní.....	161
Co může v rozhraní být	162
Dědičnost rozhraní a skrývání členů	162
Implementace rozhraní	165
Implicitní implementace rozhraní	165
Explicitní implementace rozhraní.....	165
Překrývání implementací rozhraní v odvozené třídě	167
Pozor na vedlejší účinky hodnotových typů, které implementují rozhraní.....	171
Pravidla přiřazování členům rozhraní	172
Explicitní implementace rozhraní a hodnotové typy	176
Práce s verzemi	178
Kontrakty	179
Kontrakty implementované třídami	179
Kontrakt rozhraní.....	181
Výběr mezi rozhraními a třídami.....	181
Shrnutí	185

Kapitola 6

Přetěžování operátorů..... 187

To, že můžete, ještě neznamená, že byste měli	187
Typy a formáty přetížených operátorů	188
Operátory by neměly měnit své operandy	189
Záleží na pořadí parametrů?.....	189
Přetížení operátoru pro sčítání	190
Operátory, které lze přetěžovat.....	191
Porovnávací operátory	192
Konverzní operátory	195
Logické operátory	198
Shrnutí	201

Kapitola 7

Zabezpečení zdrojového kódu proti výskytu výjimek a jejich ošetřování 203

Jak CLR nakládá s výjimkami.....	204
Mechanismus ošetřování výjimek v CLR.....	204
Vyvolávání výjimek.....	204
Změny ve způsobu zpracování neošetřených výjimek ve verzi .NET 2.0.....	205
Přehled syntaxe příkazů try, catch a finally.....	206
Opětovné vyvolávání výjimek a překlad výjimek.....	208
Výjimky vyvolané v blocích finally.....	210
Výjimky vyvolané ve finalizérech.....	211
Výjimky ve statických konstruktorech.....	212
Kdo by měl výjimky ošetřovat?.....	213
Vyhnete se využívání výjimek k řízení běhu programu.....	214
Jak docílit neutrálního chování při výskytu výjimek.....	214
Základní struktura zdrojového kódu, který se chová při výskytu výjimek neutrálně.....	215
Omezené prováděcí oblasti.....	221
Kritické finalizéry a třída SafeHandle.....	223
Vytváření vlastních tříd výjimek.....	227
Práce s alokovanými prostředky a výjimkami.....	229
Implementace zrušení změn („rollback“).....	233
Shrnutí.....	236

Kapitola 8

Práce s řetězci 237

Základní údaje o typu String.....	237
Řetězcové literály.....	238
Specifikátory formátu a globalizace.....	239
Object.ToString, IFormattable a CultureInfo.....	240
Vytváření a registrace uživatelských typů CultureInfo.....	241
Formátovací řetězce.....	243
Metody Console.WriteLine a String.Format.....	244
Příklady formátování řetězců v uživatelských typech.....	245
Rozhraní ICustomFormatter.....	247
Porovnávání řetězců.....	249
Práce s řetězci z vnějších zdrojů.....	250
StringBuilder.....	253
Prohledávání řetězců pomocí regulárních výrazů.....	254
Vyhledávání regulárních výrazů.....	255

Vyhledávání a seskupování.....	256
Nahrazení textu regulárním výrazem	260
Možnosti vytváření regulárních výrazů	262
Shrnutí	264

Kapitola 9

Pole, kolekce a iterátory 265

Úvod do problematiky polí	265
Implicitně typovaná pole.....	267
Přetypování a kovariance	269
Třídění a prohledávání.....	270
Synchronizace.....	271
Vektory vs. pole	271
Pravouhlá multidimenzionální pole	273
Nepravidelná multidimenzionální pole.....	275
Typy kolekcí	276
Srovnání rozhraní ICollection<T> a ICollection.....	277
Synchronizace kolekcí.....	278
Seznamy.....	279
Slovníky.....	280
Množiny.....	281
Jmenný prostor System.Collections.ObjectModel.....	281
Efektivita	284
Rozhraní IEnumerable<T>, IEnumerator<T>, IEnumerable a IEnumerator	285
Typy, které produkují kolekce	289
Iterátory.....	289
Dopředné, zpětné a obousměrné iterátory.....	294
Inicializátory kolekcí	298
Shrnutí	299

Kapitola 10

Delegáty, anonymní funkce a události 301

Přehled problematiky delegátů.....	301
Vytváření a používání delegátů.....	303
Samostatný delegát.....	303
Řetězení delegátů.....	304
Iterování řetězcem delegátů	306
Otevřené delegáty (unbound, open instance).....	308
Události	310

Anonymní metody	314
Zachycené proměnné a uzávěry	317
Pozor na překvapení, které vám může přichystat zachycená proměnná	319
Anonymní metody jako nástroj pro vazbu parametrů delegátů	322
Návrhový vzor strategie	325
Shrnutí	326

Kapitola 11

Genericita.....327

Rozdíl mezi generickými typy a šablonami v C+.....	328
Efektivita a typová bezpečnost generických konstrukcí	329
Definice generických typů a konstruované typy	331
Generické třídy a struktury	332
Generická rozhraní	335
Generické metody	335
Generické delegáty	337
Konverze generických typů	341
Výrazy výchozích hodnot	342
Nulovatelné typy	343
Přístupnost konstruovaných typů	345
Genericita a dědičnost	345
Omezení	347
Omezení netřídových typů	351
Kovariance a kontravariance	352
Kovariance	354
Kontravariance	357
Invariance	359
Variance a delegáty	359
Generické systémové kolekce	363
Generická systémová rozhraní	365
Vybrané problémy a jejich řešení	366
Konverze a operátory v generických typech	367
Dynamické vytváření konstruovaných typů	376
Shrnutí	378

Kapitola 12

Práce s podprocesy v jazyku C#379

Práce s podprocesy v jazyce C# a v prostředí .NET	380
Spouštění podprocesů	380

Předávání dat novým podprocesům.....	382
Použití delegátového typu ParameterizedThreadStart	383
Návrhový vzor IOU a asynchronní volání metod	384
Stavy podprocesů.....	384
Ukončování podprocesů.....	387
Zastavování podprocesů a probouzení spících podprocesů	389
Čekání na ukončení podprocesu	390
Hlavní a vedlejší podprocesy.....	390
Lokální úložiště podprocesu	391
Jak k sobě pasují neřízené podprocesy a apartmány COM.....	395
Synchronizace činnosti mezi podprocesy	396
Odlehčená synchronizace s třídou Interlocked.....	397
Třída SpinLock.....	403
Třída Monitor.....	405
Dejte si pozor na zabalování	409
Metody Pulse a Wait	410
Zámky	414
Třída ReaderWriterLock	414
Třída ReaderWriterLockSlim	417
Mutex	418
Semafor	419
Události	422
Synchronizační objekty rozhraní Win32 a třída WaitHandle	423
Použití třídy ThreadPool	425
Asynchronní volání metod.....	426
Časovače	434
Souběžné programování	435
Třída Task	436
Třída Parallel	437
Snadný vstup do fondu podprocesů.....	441
Kolekce bezpečné vzhledem k vícepodprocesovému zpracování.....	443
Shrnutí	444

Kapitola 13

Hledání kanonických tvarů v jazyce C# 445

Kanonické formy pro referenční typy.....	446
Třídy standardně zapečetujte	446
Používejte vzor Nevirtuální rozhraní (NVI)	447
Lze objekt klonovat?	450
Lze objekt odstranit?.....	456

Potřebuje objekt finalizační metodu?	458
Co pro objekt znamená rovnost?	466
Referenční typy a identita	466
Hodnotová rovnost.....	469
Překrytí metody Object.Equals pro referenční typy	470
S metodou Equals překryjte také metodu GetHashCode	473
Podporuje třída řazení?	476
Lze objekt formátovat?	479
Je objekt převoditelný?	483
Vždy dávejte přednost typové bezpečnosti	485
Použití neměnných referenčních typů	489
Kanonické tvary hodnotových typů	492
Překryjte metodu Equals kvůli lepšímu výkonu.....	492
Podporuje daný typ nějaká rozhraní?	496
Implementujte typově bezpečné formy složek rozhraní a odvozených metod	497
Shrnutí	500
Kontrolní seznam pro referenční typy	500
Kontrolní seznam pro hodnotové typy	502

Kapitola 14

Rozšiřující metody..... 503

Seznámení s rozšiřujícími metodami.....	503
Jak najde překladač rozšiřující metodu?	504
Pod pokličkou	507
Čitelnost kódu versus jeho srozumitelnost	508
Praktická doporučení	509
Zvažte použití rozšiřujících metod místo dědění	509
Vyčleňte rozšiřující metody do samostatného jmenného prostoru	511
Změna kontraktu typu může rozbít rozšiřující metody	512
Transformace	512
Řetězení operací.....	516
Vlastní iterátory	518
Vypůjčeno z funkcionálního programování	519
Vzor Návštěvník	525
Shrnutí	528

Kapitola 15

Lambda-výrazy 529

Seznámení s lambda-výrazy	529
Lambda-výrazy a uzávěry	530

Uzávěry v jazyku C# 1.0	533
Uzávěry v jazyce C# 2.0	535
Lambda-příkazy	535
Výrazové stromy	536
Operování na výrazech	539
Funkce jako data	539
Užitečné aplikace lambda-výrazů	540
Revize iterátorů a generátorů	540
Uzávěry (zachycení proměnných) a tabulace	544
Vazba parametrů (curryfikace)	548
Anonymní rekurze	550
Shrnutí	552

Kapitola 16

LINQ: Dotazy integrované do jazyka 553

Most k datům	554
Dotazové výrazy	554
Revize rozšiřujících metod a lambda-výrazů	556
Standardní dotazové operátory	557
Dotazová klíčová slova jazyka C#	559
Klauzule from a rozsahové proměnné	559
Klauzule join	561
Klauzule where a filtry	563
Klauzule orderby	563
Klauzule select a projekce	564
Klauzule let	566
Klauzule group	567
Klauzule into a pokračování	570
Přednosti lenosti	571
Iterátory jazyka C# podporují lenost	572
Ničitelé lenosti	573
Okamžité provádění dotazů	574
Revize výrazových stromů	575
Techniky z funkcionálního programování	575
Vlastní standardní dotazové operátory a líné vyhodnocování	576
Nahrazení příkazu foreach	583
Shrnutí	585

Kapitola 17**Dynamické typy 587**

Co se skrývá za klíčovým slovem dynamic	587
Jak dynamické typy fungují.....	590
Veliké sjednocení.....	592
Místa volání	592
Objekty s vlastním dynamickým chováním.....	594
Efektivita	597
Zabalování s typem dynamic.....	598
Dynamické převody.....	599
Implicitní převod dynamických výrazů	600
Dynamický výběr přetížené varianty.....	601
Dynamické dědění	603
Od typu dynamic nelze odvozovat nové třídy	603
Nelze implementovat rozhraní s typem dynamic	604
Nové třídy lze odvozovat od dynamických bazových typů.....	606
Kachní typování v jazyku C#	607
Omezení typu dynamic	610
Dynamické vytváření objektů pomocí třídy ExpandableObject.....	610
Shrnutí	614

Rejstřík 615

O autorovi

Trey Nash pracuje jako starší technik v týmu Platforms Global Escalation Services společnosti Microsoft na vývoji operačních systémů Windows a nejrůznějších dalších produktů. Když zrovna horečnatě nepracuje na útrokách operačního systému, vyučuje ladění programů pro platformu .NET a ladění programů pro uživatelský i kernel mód platformy Windows. Ve svém předchozím zaměstnání pracoval jako vedoucí softwarový vývojář bezpečnostních řešení ve společnosti Credant Technologies, čelní společnosti zabývající se vývojem bezpečnostního softwaru. Absolvoval také pracovní stáž ve velké firmě vyvíjející software pro komunikační technologii Bluetooth, kde vyvíjel řešení pro systém Windows Vista. A předtím mu byla po pět let domovem společnost Macromedia Inc., ve které pracoval několik let v týmu zabývajícím se vývojem různých produktů a navrhoval řešení různých problémů pro mnoho aplikací, včetně aplikací pro Flash, Fireworks a Dreamweaver. Až do revolučního nástupu prostředí .NET se specializoval na technologii COM/DCOM a jazyky C/C++ a knihovnu ATL. Počítačům věnuje veškerý svůj čas už od té doby, co jako třináctiletý dostal svůj první počítač TI-99/4A a k úžasu a nelibosti svých rodičů proměnil svoji mladistvou posedlost v dobře placenou práci. Titul bakaláře a inženýra elektrotechniky získal na Texas A&M University. Když zrovna neseďdí u počítače, kutí něco v garáži, hraje na klavír, piluje cizí jazyky (aktuálně ruštinu a islandštinu), nebo hraje lední hokej.

Odborný korektor

Damien Foggan je vývojář, autor a odborný korektor knih o nejnovějších technologiích. Podílel se již na více než padesáti knihách o prostředí .NET, programovacích jazycích C# a Visual Basic a prostředí ASP.NET. Je držitelem několika certifikátů MCPD odbornosti v prostředí .NET 2.0 a .NET 3.5. Najdete jej také online na adrese <http://blog.littlepond.co.uk>.

Poděkování

Psaní knihy je dlouhý a vysilující proces, v němž mě velmi podporovala moje rodina a moji přátelé, čehož si velmi vážím. Bez jejich podpory by byl celý tento proces mnohem náročnější a bezesporu také méně plodný.

Rád bych poděkoval zejména následujícím lidem, kteří se podíleli na prvních dvou vydáních této knihy (nejsou uvedeni v žádném konkrétním pořadí): Davidu Wellerovi, Stephenu Toubovi, Rexi Jaeschkeovi, Vladimíru Levinovi, Jerryemu Marescaovi, Chrise Pelsovi, Christopheru T. McNabbovi, Bradu Wilsonovi, Peteru Partchovi, Paulu Stubbovi, Rufusi Littlefieldovi, Tomasi Restrepi, Johnu Lambertovi, Joan Murrayové, Sheri Cainové, Jessice D'Amico, Karen Gettmanové, Jimu Huddlestonovi, Richardu Dal Portovi, Gary Cornellovi, Bradu Abramsovi, Ellie Fountainové, Nicole Abramowitzové, celému týmu nakladatelství Apress a konečně také Shelley Nashové, Michaelu Pulkovi, Shawn Wildermuthové, Sofii Marchantové, Jimu Comptonovi, Dominicu Shakeshaftovi, Wes Dyerové, Kelly Winquistové a Lauře Cheuové.

Za pomoc s vývojem třetího vydání bych pak rád poděkoval následujícím lidem (opět nejsou uvedeni v žádném konkrétním pořadí): Jonathanu Hassellovi, Mary Tobinové, Damienu Foggonovi a Maite Cerverové.

Jestli jsem na někoho zapomněl, je to jenom moje chyba a neměl jsem to v úmyslu. Bez vás bych to nedokázal. Díky vám všem!

Předmluva

Visual C#.NET (C#) je jazyk snadno pochopitelný pro každého, kdo zná jiný objektově orientovaný jazyk. Jazyk C# bude jednoduchý dokonce i pro někoho, kdo zná Visual Basic 6.0 a chce se naučit také nějaký objektově orientovaný jazyk. Avšak i když jazyk C# spolu s aplikačním rámcem .NET Framework umožňují rychlý vývoj jednoduchých aplikací, chcete-li psát v jazyce C# robustní rafinované aplikace, které nejsou náchylné k chybám, stále je nutné, abyste měli velký objem znalostí a věděli, jak jazyk správně používat. V této knize vás naučím vše, co potřebujete vědět, a vysvětlím vám, jak své vědomosti nejlépe využít k tomu, abyste se rychle stali opravdovým odborníkem na jazyk C#.

Idiomy a návrhové vzory jsou při nabývání a aplikaci odborných znalostí nedocenitelné a já vám ukážu, jak mnohé z nich používat k vytváření robustních efektivních aplikací, které jsou odolné vůči chybám a zabezpečené proti výskytu výjimek. Ačkoliv programátoři pracující s jazyky Java a C++ budou mnohé z nich znát, některé jsou jedinečné v aplikačním rámci .NET a jeho běhovém prostředí Common Language Runtime (CLR). Předvedu vám, jak tyto nepostradatelné idiomy a techniky aplikovat při ucelené integraci vašich C# aplikací do runtime prostředí aplikačního rámce .NET s důrazem na nové schopnosti verze jazyka C# 3.0.

Návrhové vzory ukazují nejlepší postupy při navrhování aplikací, které mnoho programátorů objevilo a opět oživilo v průběhu času. Samotný aplikační rámec .NET ve skutečnosti implementuje mnoho velmi dobře známých návrhových vzorů. A za tu dobu, co zde jsou poslední tři verze rámce .NET a poslední dvě verze jazyka C#, se také mnoho nových idiomů a postupů objevilo. V této knize je podrobně rozeberu. Navíc je třeba brát v potaz, že nedocenitelná zásoba těchto vývojových technik se v průběhu času stále vyvíjí.

Od verze jazyka C# 3.0 můžete začít jednoduše využívat techniky funkcionálního programování prostřednictvím lambda-výrazů, rozšiřujících metod a jazyka LINQ. Lambda-výrazy zjednodušují deklaraci a konkretizaci delegátů funkcí v jednom místě. Prostřednictvím lambda-výrazů lze navíc jednoduše vytvářet funkcionály, což jsou funkce, které přijímají jako své vstupní parametry jiné funkce a obvykle vrací další funkci. Dokonce i když jste mohli implementovat funkcionální programovací techniky v jazyce C# již dříve (ačkoliv to nebylo úplně snadné), nové vlastnosti verze 3.0 tvoří prostředí, ve kterém může funkcionální programování vzkvétat v úzkém spojení s typicky imperativním programovacím stylem jazyka C#. Jazyk LINQ vám pak umožňuje prostřednictvím nativní syntaxe jazyka C# vyjadřovat operace datových dotazů (které jsou ve své podstatě typicky funkcionální). Jakmile zjistíte, jak jazyk LINQ funguje, uvědomíte si, že s jeho pomocí toho můžete dělat mnohem více než pouhé dotazy na data a že jej můžete použít k implementaci komplexních funkcionálních programů.

Rámec .NET a CLR představují jedinečné běhové prostředí, které lze použít na různých platformách. C# je pouze jedním z jazyků, které se na toto výkonné běhové prostředí zaměřují, a mnoho technik, které v této knize popisují, proto lze využít i v jakémkoliv jiném jazyku zaměřeném na vývoj aplikací pro běhové prostředí rámce .NET.

Ti z vás, kteří mají velké zkušenosti s jazykem C++ a znají takové ideje, jako jsou kanonické tvary, zabezpečení vůči výjimkám, získávání prostředků při inicializaci (Resource Acquisition Is Initialization, RAII) a korektnost konstant (const-correctness), v této knize najdou popis způsobu aplikace těchto technik v jazyce C#. Vývojáři pracující v jazycích Java nebo Visual Basic, kteří strávili léta vývojem svých vlastních postupů, zde pak najdou popis způsobu jejich efektivní aplikace v jazyce C#.

Jak uvidíte, abyste se stali profesionály v C#, nemusíte strávit léta pokusy a hledáním chyb. Stačí, když získáte správné vědomosti a naučíte se je odpovídajícím způsobem využívat. Proto jsem pro vás napsal tuto knihu.

O této knize

Předpokládám, že již máte nějaké pracovní zkušenosti s některým z objektově orientovaných programovacích jazyků, například s jazykem C++, Java nebo Visual Basic .NET. Protože je syntaxe jazyka C# odvozena od syntaxe jazyků C++ a Java, nevěnuji se problematice syntaxe příliš podrobně a více do detailu ji rozebírám pouze v případech, ve kterých se od syntaxe jazyků C++ a Java naprosto liší. Pokud již o C# něco víte, můžete kapitoly 1 - 3 pouze prolistovat nebo dokonce úplně přeskočit.

1. kapitola, „Úvod do jazyka C#“, je krátkým úvodem, ve kterém rozebírám, jak vypadá aplikace napsaná v jazyce C# a jaké jsou základní rozdíly mezi programovacím prostředím jazyka C# a nativním prostředím jazyka C++.

2. kapitola, „C# a CLR“, navazuje na kapitolu první a věnuji se v ní rychlému rozboru spravovaného prostředí, ve kterém C# aplikace běží. V této kapitole dále najdete úvod do problematiky sestavení, základních stavebních kamenů aplikací, do kterých se soubory se zdrojovým kódem jazyka C# kompilují. Nakonec vysvětlím, jak jsou sestavení – díky metadatům – sebecopisná.

3. kapitola, „Přehled syntaxe jazyka C#“, mapuje syntaxi jazyka C#. Seznámím vás v ní se dvěma základními druhy typů v CLR: s typy hodnotovými a referenčními. V této kapitole také popisují jmenné prostory a způsob, jak můžete jejich prostřednictvím logicky organizovat typy a funkcionalitu vašich aplikací.

Kapitoly 4. - 13. obsahují detailní popis způsobu využití užitečných idiomů, návrhových vzorů a nejlepších postupů pro vaše programy a návrhy. Velmi jsem se snažil tyto kapitoly logicky seřadit, ale někdy se v některých z nich používají techniky nebo témata popsané v jedné z kapitol následujících. Tomu se téměř nelze vyhnout, ale snažil jsem se, aby k tomu docházelo co nejméně.

4. kapitola, „Třídy, struktury a objekty“, obsahuje detailní informace o vytváření datových typů v jazyce C#. Dozvíte se v ní více o hodnotových a referenčních typech v CLR a také se v ní dotknu nativní podpory rozhraní v prostředí CLR a jazyce C#. Dočtete se, jak v jazyce C# funguje typová dědičnost a že každý objektový typ je odvozen od typu `System.Object`. Tato kapitola také obsahuje množství informací o spravovaném prostředí a o tom, co musíte vědět, abyste definovali typy, které jsou v něm užitečné. Mnohých z těchto témat se pouze dotýkám a později je podrobně rozebírám v kapitolách následujících.

5. kapitola, „Rozhraní a smluvní vztahy“, nabízí podrobný rozbor rozhraní a jejich role v jazyce C#. Rozhraní zajišťují funkcionální smluvní vztahy, které se typy mohou rozhodnout implementovat. Ukážu vám nejrůznější způsoby, jakými může typ implementovat rozhraní, a jak se během prostředí rozhoduje, kterou metodu zavolá, když dojde k volání metody rozhraní.

6. kapitola, „Přetěžování operátorů“, obsahuje informace o tom, jak můžete přiřadit vaši vlastní funkcionalitu zabudovaným operátorům jazyka C# při jejich aplikaci na vaše vlastní typy. Dozvíte se v ní, jak operátory přetěžovat zodpovědně, protože ne všechny spravované jazyky, které kompilují kód pro CLR, mohou přetížené operátory používat.

7. kapitola, „Zabezpečení zdrojového kódu proti výskytu výjimek a jejich ošetřování“, je věnována popisu schopností ošetřování výjimek jazyka C# a prostředí CLR. Ačkoliv je samotná syntaxe výjimek podobná jejich syntaxi v jazyce C++, psaní zdrojového kódu bezpečného při výjimkách a zdrojového kódu, který se chová při výskytu výjimek neutrálně, je náročné - dokonce ještě náročnější než psaní zdrojového kódu bezpečného při výskytu výjimek v nativním C++. V této kapitole se dovíte, že lze psát zdrojový kód odolný vůči výskytu chyb a zdrojový kód, který se chová při výskytu výjimek neutrálně, aniž byste používali konstrukce příkazů `try`, `catch` nebo `finally`. V této kapitole také rozebírám některé nové schopnosti přidané do běhového modulu prostředí .NET 2.0, které vám umožňují vytvářet zdrojový kód odolný vůči výskytu chyb.

8. kapitola, „Práce s řetězci“, popisuje prvořadou pozici řetězcových typů v CLR a způsob jejich efektivního používání v jazyce C#. Velkou část této kapitoly věnuji možnostem formátování řetězců, které mají nejrůznější typy v .NET, a ukazují, jak docílit prostřednictvím implementace rozhraní `IFormattable`, aby se vaše vlastní typy chovaly podobně. Seznámím vás také s globalizačními schopnostmi aplikačního rámce a ukážu vám, jak vytvářet vaše vlastní instance typu `CultureInfo` pro kultury a regiony, o kterých rámec .NET zatím neví.

9. kapitola, „Pole, kolekce a iterátory“, je věnována popisu nejrůznějších typů polí a kolekcí, které můžete v jazyce C# použít. Můžete vytvářet dva typy vícerozměrných polí i své vlastní typy kolekcí a využít při tom pomocné třídy kolekcí. Ukážu vám, jak definovat dopředné, zpětné a obousměrné iterátory prostřednictvím nové syntaxe iterátorů zavedené ve verzi jazyka C# 2.0., takže vaše typy kolekcí budou dobře fungovat s příkazy `foreach`.

10. kapitola, „Delegáty, anonymní funkce a události“, popisuje mechanismus, který se v jazyce C# používá pro zajištění zpětných volání. Všechny životaschopné aplikační rámce vždy nabízely mechanismus zajišťující možnost implementace zpětných volání. Jazyk C# jde v tomto ohledu ještě dále a zapouzdřuje zpětná volání do objektů, které lze volat, tzv. *delegátů*. Od verze jazyka C# 2.0 navíc můžete vytvářet delegáty prostřednictvím zkrácené syntaxe tzv. *anonymních funkcí*. Anonymní funkce se podobají lambda-funkcím ve funkcionálním programování. Ukážu vám také, jak aplikační rámec delegáty využívá k tomu, aby zajistil notifikační mechanismus událostí založený na návrhovém vzoru pozorovatel (observer; používá se také označení vydavatel-předplatitel, publisher-subscriber), který vám umožňuje oddělit v návrhu zdroj události od jejího spotřebitele.

11. kapitola, „Genericita“, je úvodem do pravděpodobně nejzajímavější nové vlastnosti přidané do verze jazyka C# 2.0 a do CLR. Těm z vás, kteří mají zkušenosti s šablonami v C++, budou genericita povědomá, i když se od šablon v několika ohledech diametrálně liší. Prostřednictvím generických konstrukcí můžete dodat funkcionalitu, ve které se definují specifitější typy za chodu programu. Generické typy jsou nejužitečnější při použití s typy kolekcí a kolekce jsou díky nim v porovnání se kolekcemi v předchozích verzích prostředí .NET mnohem efektivnější. Počínaje verzí jazyka C# 4.0 je způsob jejich použití ještě intuitivnější díky podpoře kovariance a kontravariance. Přiřazení jednoho generického typu jinému, jestliže dává v rámci typového systému intuitivní smysl, je nyní možné, a díky tomu klesá dřívější potřeba množství konverzních metod.

12. kapitola, „Aplikace s více podprocesy“, rozebírá tvorbu aplikací o více podprocesech (vláknech) ve spravovaném virtuálním prováděcím prostředí jazyka C#. Máte-li zkušenosti s problematikou podprocesů v nativním prostředí Win32, zjistíte, že mezi prostředím Win32 a spravovaným virtuálním běhovým prostředím jazyka C# existují v tomto ohledu signifikantní rozdíly. Spravované prostředí vám navíc nabízí mnohem propracovanější infrastrukturu, která dělení do podprocesů značně zjednodušuje. Dozvíte se, že delegáty zajišťují při použití v souladu s návrhovým vzorem „Dlužník“ (I Owe You - IOU) výbornou přístupovou cestu do fondu vláken. Při současném provádění různých proprocesů je nepochybně nejdůležitější správná synchronizace. V této kapitole proto rozeberu nejrůznější nástroje pro synchronizaci, které mají vaše aplikace k dispozici. Souběžný způsob provádění zdrojového kódu je v současnosti jedním z nejdůležitějších témat, protože hardwarový průmysl se aktuálně místo extrémně časově náročného vývoje rychlejších procesorů zaměřuje na vývoj procesorů s více jádry. Proto ve 12. kapitole rozeberu paralelizací koncepty Parallel Extensions a knihovnu Task Parallel Library (TPL).

13. kapitola, „Hledání kanonických tvarů v C#“, je dizertací na téma nejlepších postupů při definování nových typů tak, aby je bylo možné co nepřirozeněji používat a aby je jejich spotřebitelé nemohli nechtěně zneužívat. Tomuto tématu se letmo věnuji i v jiných kapitolách, v 13. kapitole pak podrobně. Na jejím konci najdete také kontrolní seznam problémů, které byste měli brát při definování nových typů v potaz.

14. kapitola, „Rozšiřující metody“, je věnována této nové vlastnosti jazyka zavedené ve verzi 3.0. Protože můžete rozšiřující metody volat jako metody instancí typu, který rozšiřují, lze jejich prostřednictvím rozšiřovat smluvní vztahy, k jejichž dodržování se typy zavazují. Avšak tím možnosti použití rozšiřujících metod rozhodně nekončí. V této kapitole vám předvedu, jak rozšiřující metody začínají otevírat svět funkcionálního programování v C#.

15. kapitola, „Lambda-výrazy“, se věnuje další nové vlastnosti jazyka C# přidané do jeho třetí verze. Díky lambda-výrazům můžete deklarovat a konkretizovat delegáty prostřednictvím syntaxe, která je velmi stručná a vizuálně deskriptivní. Ačkoliv lze ke stejnému účelu využít i anonymní metody, jejich aplikace vyžaduje mnohem více psaní a není tak syntakticky elegantní. Počínaje verzí jazyka C# 3.0 také můžete lambda-výrazy konvertovat do stromů. Jazyk má tedy zabudovanou schopnost konverze zdrojového kódu do datových struktur. Tato schopnost je užitečná i sama o sobě, ale nesrovnatelně užitečnější ve spojení s jazykem LINQ. Lambda-výrazy ve spojení s rozšiřujícími metodami pak završují implementaci funkcionálního programování do jazyka C#.

16. kapitola, „LINQ: Integrovaný dotazovací jazyk“, je završením rozboru nových vlastností zavedených ve verzi jazyka C# 3.0. Díky LINQ-výrazům vytvořeným prostřednictvím klíčových slov orientovaných na tento zabudovaný dotazovací jazyk můžete konzistentním způsobem integrovat do vašeho zdrojového kódu datové dotazy. LINQ tvoří spojnicí mezi typicky imperativním světem programování v jazyce C# a funkcionálním světem programování datových dotazů. LINQ-výrazy můžete použít při manipulaci s běžnými objekty i s daty pocházejícími z SQL databází, Datasetů a XML (mimo jiné).

17. kapitola, „Dynamické typy“, rozebírá nové dynamické typy přidané do C# 4.0. S dynamickými typy je spojená jednodušší integrace s dynamickými jazyky v prostředí .NET, včetně objektů COM Automation. Pryč jsou dny psaní nepřirozeného a obtížně čitelného kódu ve snaze integrovat tyto komponenty - implementace dynamických typů dělá všechnu tuto mechanickou práci za vás. Implementace dynamických typů využívá prostředí Dynamic Language Runtime (DLR), které je také základem pro dynamické jazyky jako IronRuby a IronPython (mimo jiné). A když použijete dynamické typy s typy v DLR, jako je například typ `ExpandableObject`, můžete v jazyce C# vytvářet a implementovat typy, které jsou opravdu dynamické.