

Jeffrey Richter

Microsoft .NET Framework- Programmierung

Microsoft[®]
Press

Inhaltsverzeichnis

Danksagungen	XIII
Einführung	XV
Bestandteile der Microsoft .NET-Initiative	XVIII
Ein Betriebssystem als Basis: Windows	XIX
Nützliche Produkte: Die .NET Enterprise-Server	XIX
Microsoft XML-Webdienste: .NET My Services	XX
Die Entwicklungsplattform: Das .NET Framework	XX
Die Entwicklungsumgebung: Visual Studio .NET	XXIV
Ziel dieses Buchs	XXV
Systemvoraussetzungen	XXVI
Dieses Buch ist perfekt	XXVI
Support	XXVI
Teil I	
Grundlagen des Microsoft .NET Frameworks	1
1 Die Architektur der .NET Framework-Entwicklungsplattform	3
Quellcode in verwaltete Module kompilieren	3
Verwaltete Module in Assemblys kombinieren	6
Laden der Common Language Runtime	8
Ausführen des Codes in der Assembly	10
IL und der Schutz geistigen Eigentums	11
Standardisierung von .NET Framework	12
IL und Überprüfung	16
Ist Ihr Code sicher?	17
Die .NET Framework-Klassenbibliothek	18
Das Common Type System	21
Die Common Language Specification	23
Interoperabilität mit nicht verwaltetem Code	27
2 Anwendungen und Typen erstellen, installieren und administrieren	31
Ziele der .NET Framework-Weitergabe	32
Typen erstellen und in Modulen zusammenfassen	33
Module zu einer Assembly zusammenfassen	40
.NET Assemblys in Visual Studio zu einem Projekt hinzufügen	45
Der Assemblylinker	46
Ressourcendateien in die Assembly aufnehmen	47

Versionsinformationen über eine Assembly	49
Versionsnummern	51
Kultur	52
Einfache Anwendungsweitergabe	53
Einfache Administration	55
Suchen nach Assemblydateien	57
3 Gemeinsam genutzte Assemblys	61
Zwei Arten von Assemblys, zwei Arten der Weitergabe	62
Assemblys einen starken Namen geben	63
Der globale Assemblycache	68
Die interne Struktur des globalen Assemblycaches	72
Eine Assembly erstellen, die auf eine Assembly mit starkem Namen verweist	74
Assemblys mit starken Namen sind fälschungssicher	76
Verzögertes Signieren	77
Assemblys mit starken Namen privat weitergeben	81
Parallelausführung	82
Wie die CLR Typverweise auflöst	83
Fortgeschrittene Administration	85
Das .NET Framework-Konfigurationstool	88
Herausgeberrichtlinien	88
Eine fehlerhafte Anwendung reparieren	91
Teil II	
Arbeiten mit Typen und der Common Language Runtime	95
4 Grundlagen zu Typen	97
Alle Typen sind von <i>System.Object</i> abgeleitet	97
Typkonvertierung	99
Typkonvertierung mit den C#-Operatoren <i>is</i> und <i>as</i>	101
Namespaces und Assemblys	102
Der Zusammenhang zwischen Namespaces und Assemblys	106
5 Primitive Typen, Verweistypen, Werttypen	107
Primitive Typen in der Programmiersprache	107
Überprüfte und nicht überprüfte Operationen mit primitiven Typen	110
Verweistypen und Werttypen	113
Wie die CLR die Anordnung der Felder eines Typs steuert	117
Schachteln von Werttypen	117
6 Grundlegende Operationen von Objekten	129
Gleichheit von Objekten und Identität	129
<i>Equals</i> für Verweistypen, deren Basisklassen die Implementierung von <i>Object</i> nicht überschreiben	130
<i>Equals</i> für Verweistypen, deren Basisklassen die Implementierung von <i>Object</i> überschreiben	132
<i>Equals</i> für Werttypen	133
Zusammenfassung zur Implementierung von <i>Equals</i> und den Operatoren <i>==</i> / <i>!=</i>	135

Identität	135
Hashcodes von Objekten	136
Klonen von Objekten	138
Teil III	
Typen entwerfen	141
7 Typmember und Zugriffssteuerung	143
Typmember	143
Zugriffsmodifizierer und vordefinierte Attribute	146
Vordefinierte Attribute für Typen	147
Vordefinierte Attribute für Felder	148
Vordefinierte Attribute für Methoden	148
8 Konstanten und Felder	151
Konstanten	151
Felder	152
9 Methoden	155
Instanzkonstruktoren	155
Typkonstruktoren	161
Methoden zum Überladen von Operatoren	163
Operatoren und die Interoperabilität von Programmiersprachen	165
Methoden für Konvertierungsoperatoren	167
Jeffs Meinung über Microsofts Namensregeln für Operatormethoden	168
Argumente als Verweise an eine Methode übergeben	171
Methoden mit einer variablen Anzahl von Parametern	176
Wie virtuelle Methoden aufgerufen werden	178
Versionen von virtuellen Methoden	179
10 Eigenschaften	183
Parameterlose Eigenschaften	183
Eigenschaften mit Parametern	187
Die primäre Eigenschaft auswählen	191
11 Ereignisse	193
Einen Typ entwerfen, der ein Ereignis anbietet	195
Einen Typ entwerfen, der ein Ereignis empfängt	198
Das Anmelden von Ereignisempfängern explizit steuern	200
Typen entwickeln, die viele Ereignisse definieren	202
Der Typ <i>EventHandlerSet</i>	206
Teil IV	
Wichtige Typen	209
12 Arbeiten mit Text	211
Zeichen	211
Der Typ <i>System.String</i>	214

Strings anlegen	214
Strings sind unveränderlich	216
Vergleichen von Strings	217
Japanische Schriftzeichen	221
String-Interning	222
String-Pooling	225
Die Zeichen innerhalb eines Strings untersuchen	225
Sonstige Stringoperationen	228
Einen String effizient zusammenstellen	229
Ein <i>StringBuilder</i> -Objekt anlegen	229
Die Member von <i>StringBuilder</i>	230
Stringdarstellungen für ein Objekt	232
Bestimmte Formate und Kulturen	233
Mehrere Objekte in einem einzigen String formatieren	236
Benutzerdefinierte Formatierungsmethoden	238
Ein Objekt aus dessen Stringdarstellung extrahieren	240
Zeichenkodierung: Konvertieren zwischen Zeichen und Bytes	243
Kodieren/Dekodieren von Zeichen- und Bytestreams	249
Kodieren und Dekodieren von Base-64-Strings	250
13 Enumerationstypen und Bitflags	251
Enumerationstypen	251
Bitflags	255
14 Arrays	259
Alle Arrays sind implizit von <i>System.Array</i> abgeleitet	262
Typkonvertierung bei Arrays	263
Arrays als Parameter und Rückgabewerte	265
Arrays mit einem Startindex ungleich null	266
Schneller Arrayzugriff	267
Die Größe eines Arrays ändern	270
15 Schnittstellen	273
Schnittstellen und Vererbung	273
Eine Anwendung entwerfen, die Plug-In-Komponenten unterstützt	278
Felder in einem geschachtelten Werttyp mit Hilfe von Schnittstellen ändern	279
Mehrere Schnittstellen implementieren, die dieselbe Methode haben	282
Schnittstellenmember explizit implementieren	283
Vorsicht bei expliziten Schnittstellenmemberimplementierungen	286
16 Benutzerdefinierte Attribute	289
Benutzerdefinierte Attribute einsetzen	289
Eigene Attribute definieren	292
Datentypen für Konstruktor, Felder und Eigenschaften von Attributen	295
Benutzerdefinierte Attribute auswerten	296
Zwei Attributinstanzen miteinander vergleichen	300
Pseudo-benutzerdefinierte Attribute	303

17 Delegaten	305
Ein erster Blick auf Delegaten	305
Rückrufe in statische Methoden	308
Rückrufe in Instanzmethoden	309
Delegaten enträtselt	309
<i>System.Delegate</i> und <i>System.MulticastDelegate</i>	313
Delegaten vergleichen	315
Delegatenketten	315
Delegatenketten in C#	319
Einzelne Objekte einer Delegatenkette aufrufen	320
Delegaten und Reflektion	322
Teil V	
Typen verwalten	325
18 Ausnahmen	327
Die Evolution der Ausnahmebehandlung	328
Die Funktionsweise der Ausnahmebehandlung	330
Der <i>try</i> -Block	331
Der <i>catch</i> -Block	331
Der <i>finally</i> -Block	333
Was genau ist eine Ausnahme?	334
Implizite Annahmen, die Entwickler fast immer vergessen	336
Die Klasse <i>System.Exception</i>	338
Ausnahmeklassen in der FCL	339
Eine eigene Ausnahmeklasse definieren	341
Wie Sie Ausnahmen richtig einsetzen	345
Sie können gar nicht genug <i>finally</i> -Blöcke haben	346
Fangen Sie nicht alles ab	347
Nach dem Abfangen einer Ausnahme die Situation bereinigen	348
Eine teilweise ausgeführte Operation abbrechen	349
Implementierungsdetails verstecken	350
Was stimmt nicht mit der FCL?	352
Leistungsfragen	354
Catch-Filter	356
Unbehandelte Ausnahmen	358
Steuern, wie die CLR auf eine unbehandelte Ausnahme reagiert	363
Unbehandelte Ausnahmen und Windows Forms	364
Unbehandelte Ausnahmen und ASP.NET Web Forms	365
Unbehandelte Ausnahmen und ASP.NET XML-Webdienste	366
Stackabbilder nach Ausnahmen	366
Stackabbilder an Remoterechner übertragen	368
Ausnahmen debuggen	369
Die Art des debuggten Codes festlegen	372
19 Automatische Speicherverwaltung	375
Arbeiten auf einer Plattform mit automatischer Speicherverwaltung	375
Der Garbage Collection-Algorithmus	379
Objektauflösung	382

Wann werden <i>Finalize</i> -Methoden aufgerufen?	388
Interna des Objektabschlusses	389
Ein Objekt zum Aufräumen zwingen	392
Einen Typ benutzen, der das Beseitigungsverfahren implementiert	397
Die C#-Anweisung <i>using</i>	401
Ein interessantes Abhängigkeitsproblem	402
Schwache Verweise	403
Funktionsweise der schwachen Verweise	405
Wiederbelebung	406
Ein Objektpool auf der Basis der Wiederbelebung	408
Generationen	410
Den Garbage Collector vom Programm aus steuern	414
Weitere Leistungsaspekte des Garbage Collectors	416
Synchronisationsfreie Objekterstellung	418
Skalierbare parallele Garbage Collections	418
Garbage Collection im Hintergrund	418
Große Objekte	420
Garbage Collections überwachen	420
20 CLR-Hosting, AppDomains und Reflektion	423
Metadaten: Die Basis des .NET Frameworks	424
CLR-Hosting	425
AppDomains	426
Über AppDomain-Grenzen auf Objekte zugreifen	429
AppDomain-Ereignisse	430
Wie Anwendungen die CLR hosten und AppDomains verwalten	430
»Yukon«	432
Grundlagen der Reflektion	432
Reflektion über Typen einer Assembly	434
Reflektion über die Assemblys einer AppDomain	436
Reflektion über die Member eines Typs: Bindung	436
Assemblys explizit laden	438
Assemblys als Datendateien laden	440
Eine Hierarchie der von <i>Exception</i> abgeleiteten Typen erstellen	441
Assemblys explizit schließen: Eine AppDomain schließen	443
Einen Verweis auf ein <i>System.Type</i> -Objekt holen	445
Reflektion über die Member eines Typs	448
Eine Instanz eines Typs anlegen	450
Methoden eines Typs aufrufen	452
Einmal binden, mehrmals aufrufen	455
Reflektion über die Schnittstellen eines Typs	459
Geschwindigkeit der Reflektion	461
Stichwortverzeichnis	463
Der Autor	489