

.NET BIBLIOTHEK
hrsg. von dr. holger SCHWICHTENBERG

www.IT-Visions.de



manfred STEYER
holger SCHWICHTENBERG
matthias FISCHER
jörg KRAUSE

VERTEILTE SYSTEME UND SERVICES MIT .NET 4.5

**KONZEPTE UND LÖSUNGEN MIT WCF 4.5
UND ASP.NET WEB-API**

2. Auflage



**Zusammenspiel von WCF, ASP.NET Web API, SignalR,
Workflow Foundation, Identity Foundation, Entity
Framework und Azure Service Bus**



EXTRA: Mit kostenlosem E-Book

HANSER

Steyer/Schwichtenberg/Fischer/Krause

Verteilte Systeme und Services mit .NET 4.5



Bleiben Sie auf dem Laufenden!

Der Hanser Computerbuch-Newsletter informiert Sie regelmäßig über neue Bücher und Termine aus den verschiedenen Bereichen der IT. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter

www.hanser-fachbuch.de/newsletter

Manfred Steyer
Holger Schwichtenberg
Matthias Fischer
Jörg Krause

Verteilte Systeme und Services mit .NET 4.5

Konzepte und Lösungen
für WCF 4.5 und ASP.NET Web-API

2., überarbeitete und erweiterte Auflage

HANSER

Die Autoren:

Manfred Steyer, IT-Visions.de, FH CAMPUS 02, Graz

Holger Schwichtenberg (Herausgeber und Autor), IT-Visions.de, Essen

Matthias Fischer, Rathenow bei Berlin

Jörg Krause, Berlin

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autoren und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autoren und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.



Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2013 Carl Hanser Verlag München, www.hanser.de

Lektorat: Sieglinde Schärli

Copy editing: Sandra Gottmann, Münster-Nienberge

Herstellung: Irene Weilhart

Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München

Umschlagrealisation: Stephan Rönigk

Gesamtherstellung: Kösel, Krugzell

Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702

Printed in Germany

print-ISBN: 978-3-446-43443-1

e-book-ISBN: 978-3-446-43565-0

Inhalt

Geleitwort des Herausgebers	XVII
Vorwort	XIX
1 Serviceorientierung	1
1.1 Konzeptionelle Ebene	1
1.1.1 Betriebswirtschaftliche Sicht	1
1.1.2 Technische Sicht	2
1.1.3 Was ist ein Service?	3
1.2 Technische Realisierung	4
1.2.1 SOAP	4
1.2.2 Web Service Description Language (WSDL)	6
1.2.3 Universal Description, Discovery and Integration	7
1.2.4 WS-I	7
1.2.5 WS-*	8
1.2.6 RESTful Web Services als Gegenbewegung zu SOAP	8
1.2.7 POX-Services und Web APIs	11
1.2.8 SOAP und REST im Vergleich	12
1.3 WCF vs. ASP.NET Web API	13
2 WCF im Überblick	15
2.1 Architektur	15
2.2 Standard-Bindings	16
2.3 Hosting von Services	18
2.4 Erste Schritte mit WCF	19
2.4.1 Erstellen eines Web-Service-Projektes	19
2.4.2 Web-Service mit Client konsumieren	27
2.4.3 Mit Laufzeit-Proxy auf Service zugreifen	31
2.4.4 Service zur Verwendung von ws2007HttpBinding konfigurieren	32
2.4.5 NetTcpBinding und Self-Hosting	33

3	Services mit WCF erstellen	39
3.1	Verträge	39
3.1.1	Serviceverträge	39
3.1.2	Datenverträge	40
3.1.3	Nachrichtenverträge	43
3.1.4	SOAP-Binding festlegen	44
3.2	Instanziierung von Services	44
3.2.1	PerCall	45
3.2.2	Single	46
3.2.3	PerSession	46
3.3	Nebenläufigkeit	51
3.4	Asynchrone Service-Operationen	51
3.5	WCF konfigurieren	52
3.5.1	WCF deklarativ konfigurieren	52
3.5.2	WCF programmatisch konfigurieren	55
3.5.3	In IIS gehostete Services programmatisch konfigurieren (ab .NET 4.5)	55
3.5.4	Benutzerdefinierte Bindings	57
3.5.5	Einschränkungen für Bindings festlegen	58
3.5.6	Drosselung	59
3.5.7	Port-Sharing bei TCP-basierten Services	60
3.5.8	Konfiguration des Proxy-Servers	61
3.6	Metadaten	62
3.7	Services diagnostizieren	64
3.7.1	Protokollierung konfigurieren	64
3.7.2	Leistungsindikatoren	66
3.8	One-Way-Operationen	67
3.9	Duplex-Operationen	67
3.9.1	Unterstützte Bindings	68
3.9.2	Implementierung von Duplex-Szenarien	69
3.9.3	Konfigurieren von Duplex-Szenarien	70
3.9.4	Callbacks mit WebSockets (ab .NET 4.5)	72
3.9.5	Grenzen von Callbacks	73
3.10	UDP und Multicasts (ab .NET 4.5)	74
3.11	Umgang mit binären Daten	75
3.11.1	MTOM	76
3.11.2	Streaming	80
3.12	Fehlerbehandlung und FaultContracts	83
3.13	ASP.NET-Kompatibilität	85
3.14	Zuverlässige Sitzungen	87
3.14.1	Verlässliche Sitzungen konfigurieren	88
3.14.2	Verlässliche Sitzungen erzwingen	89
3.14.3	Idempotente Services als Alternative zu WS-ReliableMessaging	89
3.15	Transaktionale Services	90
3.15.1	Zwei-Phasen-Commit	90
3.15.2	Plug-in für WS-AtomicTransactions einrichten	90

3.15.3	Distributed Transaction Coordinator (DTC) einrichten	91
3.15.4	Transaktionen konfigurieren und nutzen	92
3.15.5	Transaktionsisolationslevel	94
3.15.6	Transaktion am Client starten	95
3.15.7	Transaktionen und Sessions	95
3.16	Queued Services	96
3.16.1	Microsoft Message Queuing Services (MSMQ)	96
3.16.2	Implementierung von Queued Services	97
3.16.3	Queued Services konfigurieren	98
3.17	REST-basierte Services mit WCF	100
3.17.1	REST-Services mit WCF implementieren und konsumieren	101
3.17.2	Antwortformat dynamisch festlegen	104
3.17.3	Hilfe-Seiten (Help Pages)	105
3.17.4	ASP.NET Cache Profiles	107
3.17.5	REST-Services über ASP.NET-Routen	108
3.17.6	Ausnahmen auf HTTP-Statuscodes abbilden	108
3.17.7	Conditional GET und ETag-Unterstützung	109
3.18	WCF und Windows 8	110
4	Sicherheit von WCF-Diensten	111
4.1	Überblick über WCF-Security	111
4.1.1	Transport- und Nachrichtensicherheit	111
4.1.2	Festlegen der zu verwendenden Credentials	113
4.1.3	Authentifizierung und Autorisierung	114
4.1.4	Service-Identitäten	116
4.1.5	Verschlüsseln und Signieren	116
4.2	Windows-Security	118
4.2.1	Impersonation	118
4.2.2	Kerberos vs. NTLM	119
4.3	Web-Security mit SSL und IIS	121
4.3.1	IIS für die Verwendung von SSL konfigurieren	121
4.3.2	Konfiguration des Service	125
4.3.3	Aufruf der Service-Operation	126
4.3.4	Benutzerdefinierte Authentifizierung und Autorisierung	126
4.4	Web-Security mit SSL ohne IIS	133
4.5	Nachrichtenbasierte Sicherheit	134
4.5.1	Zertifikate einrichten	134
4.5.2	Konfiguration	135
4.5.3	Aufruf der Service-Operation	137
4.6	Client-Zertifikate	137
4.7	Federated und Claims-based Security mit Windows Identity Foundation (WIF)	139
4.7.1	Architektur	140
4.7.2	Web-Service-Standards	141
4.7.3	Implementierung in .NET 4.5	141
4.7.4	Implementierung für Versionen vor 4.5	155

5	Lose Kopplung mit WCF	167
5.1	Routing (System.ServiceModel.Routing)	167
5.1.1	Architektur	168
5.1.2	Routerarten	169
5.1.3	Beispielanwendung (Routing Testclient)	170
5.1.4	Filter	171
5.1.5	Erstellen des Routers mit WCF 4.x	174
5.1.6	Konfiguration eines Routers	174
5.1.7	Router für das Routing-TestszENARIO	175
5.1.8	Entwicklung des Clients	180
5.1.9	Dynamische Filter entwickeln	181
5.1.10	Dynamische Filter mit eigener Filtertabelle	183
5.1.11	Leistungsverlust durch Routing	187
5.2	Discovery	188
5.2.1	Dienst mit Discovery	188
5.2.2	Client mit Discovery	189
5.2.3	Mögliche Services mittels Scopes einschränken	190
5.2.4	Clients für die Verwendung mit Discovery deklarativ konfigurieren	191
5.2.5	Ankündigungen (Announcements)	192
5.3	Ereignisse und Publish-/Subscribe-Szenarien	194
5.3.1	Service-Verträge	194
5.3.2	Implementierung eines Publish-/Subscribe-Service	195
5.3.3	Konfiguration	197
5.3.4	Implementierung des zu benachrichtigenden Clients	198
5.3.5	Weiterführende Überlegungen	199
5.4	ESB-Lösungen mit WCF entwickeln	199
5.4.1	Catch-All-Verträge	199
5.4.2	Nachrichtentransformation	200
5.5	Freie ESB-Implementierung	204
6	ASP.NET Web API	205
6.1	REST-Services mit ASP.NET Web API erstellen	205
6.1.1	Parameter und Rückgabewerte	206
6.1.2	Dynamische Parameter	208
6.1.3	REST-Services konfigurieren	209
6.1.4	REST-Services mit Fiddler testen	210
6.1.5	Mehr Kontrolle über HTTP	211
6.1.6	REST-Services über HttpClient konsumieren	214
6.1.7	Hilfe-Seiten	217
6.2	Tracing	221
6.2.1	Standard-Implementierung von ITraceWriter	221
6.2.2	Eigenen TraceWriter implementieren	222
6.3	OData-Unterstützung	223
6.4	Self-Hosting mit Web APIs	223
6.5	Querschnittsfunktionen an globalen Stellen platzieren	224

6.5.1	Querschnittsfunktionen mit Message-Handler implementieren	225
6.5.2	Handler mit HttpClient verwenden	226
6.5.3	Querschnittsfunktionen mit Filter realisieren	228
6.6	Erweiterte Konfigurationsmöglichkeiten	232
6.6.1	Benutzerdefinierte Routen	232
6.6.2	Controller-basierte Konfiguration	232
6.6.3	Routen-basierte Konfiguration	233
6.7	Deklaratives Validieren von Parametern	234
6.7.1	Verwenden von Data-Attributen	234
6.7.2	Auswerten von Validierungsattributen	236
6.7.3	Benutzerdefinierte Validierungsattribute	237
6.8	Benutzerdefinierte Formate unterstützen	238
6.8.1	Formatter implementieren	238
6.8.2	Formatter serverseitig registrieren und testen	240
6.8.3	Formatter mit HttpClient verwenden	240
6.8.4	Binäre Serialisierung mit BSON	241
6.9	Serialisierung beeinflussen	241
6.9.1	JSON-Serializer konfigurieren	242
6.9.2	XML-Serializer konfigurieren	243
6.9.3	Eigenschaften von der Serialisierung ausnehmen	243
6.9.4	Zirkuläre Referenzen serialisieren	243
6.10	Streaming	246
6.10.1	Action-Methoden für Streaming vorbereiten	246
6.10.2	Streaming in Self-Hosting-Szenarien konfigurieren	247
6.10.3	Streaming für IIS konfigurieren	247
6.10.4	Streams über HttpClient verwenden	249
6.11	Fortschritt ermitteln	250
6.12	Web API und HTML-Formulare	251
6.12.1	Einfache Formular-Felder übermitteln	251
6.12.2	Dateiupload via HTML-Formular	252
6.13	ASP.NET Web API erweitern	254
6.13.1	Abhängigkeiten auflösen mit benutzerdefiniertem DependencyResolver	254
6.13.2	Zusätzliche Assemblies mit AssemblyResolver laden	255
6.13.3	Service-Operationen über HttpActionSelector auswählen	256
6.13.4	Controller über HttpControllerSelector auswählen	257
6.13.5	Methodenparameter auf benutzerdefinierte Weise mit HttpParameterBinding	259
7	ASP.NET Web API Security	263
7.1	Verschlüsselte Übertragung	263
7.1.1	SSL mit IIS	263
7.1.2	SSL in Self-Hosting-Szenarien	263
7.1.3	Diskussion über Nachrichtensicherheit	264
7.2	Authentifizierung und Autorisierung	264

7.2.1	Operationen absichern	264
7.2.2	HTTP-Authentifizierung mit IIS	266
7.2.3	HTTP-Authentifizierung in Self-Hosting-Szenarien	267
7.2.4	Benutzer mit HttpClient authentifizieren	268
7.2.5	Benutzerdefinierte Security mit Handler	268
7.2.6	Benutzerdefinierte Security mit HTTP-Module	270
7.2.7	Mit Client-Zertifikaten arbeiten	273
7.3	Single Sign-On mit OAuth2 und DotNetOpenAuth	277
7.3.1	OAuth2	277
7.3.2	OAuth2 und REST-Services	278
7.3.3	Implementieren eines Authorization Servers mit DotNetOpenAuth ...	278
7.3.4	Client für OAuth2-Szenario	284
7.3.5	Service mittels OAuth2 absichern	285
8	ASP.NET SignalR	287
8.1	Long-Polling	287
8.2	Web Sockets	288
8.3	ASP.NET SignalR	288
8.4	PersistentConnection	289
8.4.1	Erste Schritte mit SignalR und PersistentConnection	289
8.4.2	Lifecycle-Methoden	290
8.4.3	URL-Mapping für PersistentConnection	291
8.4.4	Einfacher Client für eine PersistentConnection	291
8.4.5	Einfacher JavaScript-Client für eine PersistentConnection	292
8.4.6	Daten über serverseitige Prozesse an Connection senden	294
8.5	Hubs	294
8.5.1	Methoden und Callbacks mit SignalR und Hubs	294
8.5.2	URL-Mapping für Hubs	296
8.5.3	Lifecycle-Methoden	296
8.5.4	Hubs konsumieren	296
8.5.5	Hubs über JavaScript konsumieren	299
8.5.6	Gruppen	302
8.5.7	Hubs über serverseitige Prozesse benachrichtigen	303
8.6	Pipeline-Modules für Querschnittsfunktionen	303
8.7	SignalR konfigurieren	305
8.8	SignalR skalieren	305
9	Hosting von WCF- und Web API-Diensten	307
9.1	Hosting in Windows-Systemdiensten	307
9.1.1	Erstellen eines Windows-Systemdienstes	308
9.1.2	Ergänzen eines WCF-Dienstes in einem Systemdienst	310
9.1.3	Ergänzen eines Web API-Dienstes in einem Systemdienst	311
9.1.4	Trick für das Debugging eines Systemdienstes	311
9.1.5	Vorbereiten der Installation eines Systemdienstes	314
9.1.6	Installation eines Systemdienstes	315

9.2	Hosting im Internet Information Server (IIS)/Windows Server AppFabric	317
9.2.1	Von IIS über WAS zu AppFabric	317
9.2.2	IIS-Websites	318
9.2.3	IIS-Anwendungen	324
9.2.4	IIS-Anwendungspools	325
9.2.5	Autostart	331
9.3	WCF-Projekte für den IIS	334
9.3.1	.svc-Dateien	335
9.3.2	Test des Dienstes	335
9.3.3	Abruf der Metadaten	336
9.3.4	Konfigurationsdatei	337
9.3.5	WCF ohne SVC-Datei	339
9.3.6	Eigene ServiceHostFactory	339
9.4	ASP.NET Web API-Projekte für den IIS	340
9.5	Installieren von WCF- und Web API-Diensten im IIS	340
9.5.1	Manuelles Verbreiten von Diensten (XCopy-Deployment)	340
9.5.2	Verbreiten mit der Funktion „Build/Publish“ in Visual Studio direkt auf einen IIS	341
9.5.3	Verbreiten von Diensten mit dem IIS Web Deployment Tool (MSDeploy)	342
9.6	Konfiguration und Monitoring mit den „AppFabric“-Erweiterungen	352
9.6.1	Installation	353
9.6.2	AppFabric-Ansichten	354
9.6.3	Überwachungsfunktionen	356
9.6.4	Weitere Konfigurationsmöglichkeiten	358
10	Service Bus und Access Control für Windows Azure und Windows Server	361
10.1	Überblick	361
10.2	Namespace einrichten	362
10.3	Relaying	363
10.3.1	Relay-Bindings	364
10.3.2	Service über Relaying bereitstellen	364
10.3.3	Service über Relaying konsumieren	367
10.4	Lose Kopplung mit Queues und Topics	368
10.4.1	Zugriff auf Queues und Topics	370
10.4.2	Transaktionen	370
10.4.3	Sitzungen	371
10.4.4	Metadaten, Filtern und Stamping	371
10.4.5	Queues über die .NET-API ansprechen	371
10.4.6	Queues programmatisch erzeugen	372
10.4.7	Nachrichten an Queues senden	373
10.4.8	Nachrichten aus Queues abrufen	374
10.4.9	Mit Sitzungen arbeiten	376
10.4.10	Aus Dead Letter Queue lesen	377

10.4.11	Topics und Abonnenten einrichten	378
10.4.12	Nachrichten unter Verwendung von Topics senden und abrufen	380
10.4.13	Queues über WCF ansprechen	382
10.4.14	WCF-Sitzungen realisieren	385
10.4.15	Topics via WCF abonnieren	386
10.5	Windows Azure Access Control	387
10.5.1	AC konfigurieren	387
10.5.2	Service mit AC absichern	389
10.5.3	Azure Service Bus mit AC absichern	393
10.6	Windows Server Service Bus	394
10.6.1	Namespaces administrieren	395
10.6.2	Auf Queues und Topics zugreifen	395
10.6.3	Zugriff auf Queues und Topics einschränken	396
11	Datenbasierte Services	397
11.1	Was sind datenbasierte Services?	397
11.2	Services und ADO.NET Entity Framework	397
11.2.1	Vom DataSet zum Entity Framework	397
11.2.2	Grundlagen der Self-Tracking Entities	399
11.2.3	Beispielanwendung	403
11.3	ADO.NET Entity Framework Code First (Code Only)	412
11.3.1	Die drei Ebenen des ADO.NET Entity Framework	412
11.3.2	Vorbereitungen	413
11.3.3	Entitätsklassen	413
11.3.4	Kontextklasse	418
11.3.5	Nutzen der Kontextklasse	419
11.3.6	Einsatzbeispiel	420
11.3.7	Konvention vor Konfiguration	423
11.3.8	Codebasierte Konfiguration	425
11.3.9	Schemamigrationen	426
11.3.10	Entity Framework Power Tools	427
11.3.11	Code Only und Webservices	429
11.4	Kritik an der Implementierung eines datenbasierten Service mit WCF	430
11.5	WCF Data Services (Open Data Protocol)	431
11.5.1	Lizenz und Standardisierung	431
11.5.2	Bezug der WCF Data Services	431
11.5.3	Andere Bibliotheken	432
11.5.4	Rahmenbedingungen	433
11.5.5	OData-Beispiel	433
11.5.6	Architektur	435
11.5.7	Abfragesyntax	436
11.5.8	Einen WCF Data Service erstellen	437
11.5.9	Zugriffsrechte	437
11.5.10	Fehlermeldungen aktivieren	439
11.5.11	Einen WCF Data Service im Webbrowser testen	440
11.5.12	Abruf der Metadaten	442

11.5.13	Einen WCF Data Service mit Fiddler testen	443
11.5.14	Einen .NET-basierten Client erstellen	443
11.5.15	Tipps und Tricks	448
11.6	WCF RIA Services	457
11.6.1	Architektur der RIA Services	458
11.6.2	Einen RIA Service erstellen	459
11.6.3	Einen RIA Service nutzen	463
11.7	Datenbasierte REST-Services mit ASP.NET Web API und dem Open Data Protocol (OData)	466
11.7.1	Daten mit OData flexibel abfragen	466
11.7.2	Mögliche OData-Abfragen einschränken	468
11.7.3	OData-Abfragen global aktivieren	470
11.7.4	OData-Abfragen manuell auswerten	470
11.7.5	Daten mit OData verwalten	471
11.8	Vergleich und Fazit	475
12	Workflows und Workflow Services	479
12.1	Überblick	479
12.2	Visual Studio-Projektvorlagen	480
12.3	Sequenzielle Workflows	480
12.3.1	Sequenzielle Workflows erstellen	480
12.3.2	Sequenzielle Workflows ausführen	481
12.3.3	Kontrollfluss	482
12.4	Flussdiagramme	484
12.5	Zustandsautomaten (State Machines)	485
12.6	Transaktionen und Kompensation	487
12.6.1	Transaktionen	487
12.6.2	Kompensation	487
12.7	WCF Workflow Services	489
12.7.1	Workflow Services erstellen	489
12.7.2	Workflow Services testen	490
12.7.3	Workflow Services konfigurieren	491
12.7.4	Korrelation	492
12.7.5	Contract-First (ab .NET 4.5)	493
12.8	Benutzerdefinierte Aktivitäten	495
12.8.1	CodeActivity	495
12.8.2	AsyncCodeActivity	496
12.8.3	NativeActivity	497
12.8.4	Anpassen der Darstellung benutzerdefinierter Aktivitäten	500
12.9	Bookmarks	502
12.10	Persistenz	503
12.10.1	SQLWorkflowInstanceStore	503
12.10.2	Eigenschaften höherstufen	506
12.10.3	Höherstufen von Eigenschaften bei Verwendung von WCF Workflow Services	508

12.11	Versionisierung und Aktualisierung von Workflows (ab .NET 4.5)	510
12.11.1	Side-by-Side-Versionisierung	510
12.11.2	Dynamic Update	516
12.11.3	Dynamic Update mit WorkflowControlEndpoint	521
12.12	Ablaufverfolgung (Tracking)	523
12.12.1	Benutzerdefinierte Tracking Records erzeugen	523
12.12.2	Tracking Records abonnieren	524
12.12.3	Tracking Records einsehen	526
12.13	Workflow-Designer in eigenen Anwendungen hosten	527
12.14	WCF-Dienste und -Services in AppFabric hosten	528
12.14.1	Monitoring	529
12.14.2	Persistenz	531
12.14.3	Weitere Möglichkeiten	532
13	Das WCF-Erweiterungsmodell	533
13.1	Übersicht	533
13.1.1	Was sich erweitern lässt	533
13.1.2	Die Laufzeitumgebung der Applikation erweitern	534
13.2	Erweiterung des Sicherheitsmodells	534
13.2.1	Verantwortungsbereich der WCF Security Component	534
13.2.2	Das WebService-(WS-)Sicherheitsmodell	534
13.2.3	Implementierung der WebService-(WS-)Sicherheit	535
13.3	Erweiterung des Bindungssystems	539
13.3.1	Bindungen und Bindungselemente	540
13.3.2	Nachrichtenkanäle	540
13.3.3	Benutzerdefinierte Bindungen	541
13.4	Weitere Erweiterungsfunktionen	542
13.4.1	Erweiterung des Metadatenmodells	542
13.4.2	Erweiterungen der Serialisierung	543
14	WCF erweitern	545
14.1	ServiceHost und Dienstmodellebene	545
14.1.1	Aufbau der Dienstmodellebene	545
14.1.2	Erweiterung mittels Verhalten	546
14.1.3	Erweiterung von Clients	549
14.1.4	Erweiterung des Dispatchers	554
14.1.5	Erweiterbare Objekte	562
14.1.6	Erweitertes Hosting – die Klasse ServiceHostFactory	565
14.2	Erweiterung des Bindungsmodells	567
14.2.1	Konfigurieren mittels Bindungselementen	567
14.2.2	Bindungen selbst erstellen	568
14.2.3	Erstellen eines eigenen Bindungselements	570
14.3	Channel Layer- und Peer Channel-Erweiterungen	572
14.3.1	Funktionsweise	573
14.3.2	Benutzerdefinierte Übertragungskanäle	573
14.3.3	Benutzerdefinierte Nachrichtenencoder	583

14.4	Das Metadaten­system	585
14.4.1	Funktionsweise	586
14.4.2	Anwendung	586
14.5	Serialisierung und Encoder	591
14.5.1	Ersetzen von Datenverträgen	591
14.5.2	Eigene Formater	597
14.6	Erweiterung des Sicherheitsmodells	603
14.6.1	Aufbau der Sicherheitsarchitektur	603
14.6.2	Benutzerdefinierte Anmeldeinformation und Token	603
14.6.3	Benutzerdefinierte Nachrichtenverschlüsselung	621
Anhang – NuGet		631
Index		635

Geleitwort des Herausgebers

Liebe Leserinnen, liebe Leser,

das .NET Framework ist inzwischen mehr als zehn Jahre alt, und man kam ihm Reife nicht nur altersbedingt, sondern vor allem auch aus fachlichen Gründen bescheinigen. Das Besondere an .NET ist, dass es immer mehr Anwendungsarten gibt, die sich mit der gleichen Sprachsyntax, den gleichen Bibliotheken und den gleichen Werkzeugen erstellen lassen. Zu Desktop- und Standard-Web-Applikationen haben sich inzwischen Multimedia- und Office-Anwendungen sowie Rich Internet Applications und Apps gesellt. Und auch auf der Serverseite gibt es zahlreiche Möglichkeiten für den Einsatz von .NET, vom Microsoft SQL Server über Biztalk bis hin zu SharePoint. Mit der Version 4.5 liefert Microsoft wesentliche Verbesserungen für zahlreiche Teilbibliotheken des .NET Frameworks sowie auch neue Bausteine, insbesondere im Bereich der verteilten Systeme und Webservices.

Anlässlich von .NET 4.5 aktualisieren wir natürlich auch wieder die erfolgreiche Buchreihe .NET-Bibliothek, die ich für den Carl Hanser Verlag seit 2006 als Herausgeber betreue. Die Fachbücher dieser Reihe liefern fundiertes Wissen zu zentralen Bausteinen der Klassenbibliothek im .NET Framework. Die Reihe zeichnet sich durch prägnant gehaltene Bücher aus, die das elementare Wissen zu einem Fachgebiet für den professionellen Entwickler aufbereiten. Ein weiteres Merkmal der Reihe sind die Autoren, die seit vielen Jahren mit .NET-Technologien arbeiten und ihre umfangreiche Praxiserfahrung aus .NET-Projekten in die Bücher einfließen lassen.

Die Welt der verteilten Systeme ist wieder einmal im Umbruch. SOAP wird an immer mehr Stellen durch vermeintlich einfachere REST-Dienste abgelöst. Das macht sich auch im .NET Framework bemerkbar, indem es nun mit dem ASP.NET Web API eine neue Bibliothek für HTTP-basierte Dienste gibt, mit denen man REST flexibler umsetzen kann als mit der Windows Communication Foundation (WCF). Dennoch ist SOAP und damit auch die WCF in vielen Szenarien (insbesondere, wenn .NET mit .NET oder einer anderen Plattform mit starker SOAP-Unterstützung kommuniziert) überlegen, weshalb die WCF auch weiterhin den Löwenanteil in diesem Buch ausmacht.

Als neue Themen behandelt das Buch ASP.NET Web API (und die damit zusammenhängenden Sicherheitsmechanismen insbesondere OAuth), ASP.NET SignalR sowie den Service Bus für Windows Server und Windows Azure. Der Rest des Buchs wurde auf die zum Zeitpunkt der Drucklegung verfügbaren stabilen Versionen (WCF 4.5, ADO.NET Entity Framework 5.0, WCF Data Services 5.1) aktualisiert.

Ich wünsche Ihnen, dass dieses Buch zu Ihrem Projekterfolg beitragen kann.

Essen, im Januar 2013

Dr. Holger Schwichtenberg
www.IT-Visions.de

Vorwort

Die Geschichte verteilter Softwaresysteme ist fast so alt wie die Geschichte programmierbarer Rechner, die mittlerweile in verschiedensten Formen Einzug in unser tägliches Leben gehalten haben. Gerade im Zeitalter von Internet und mobilen Anwendungen sind verteilte Softwaresysteme spannender denn je. Sei es die Kommunikation zwischen Personen, zwischen unternehmensinternen und -externen Applikationen zur Automatisierung von Abläufen, oder zwischen Bestandteilen eines Systems, zum Beispiel Client und Server, – verteilte Systeme sind omnipräsent. Während sich die Prinzipien hinter diesen Applikationen in den letzten 30 Jahren kaum verändert haben, liegt der Teufel – wie so häufig – im Detail. Es gilt, verschiedene Protokolle und Formate in Einklang zu bringen, auf die mannigfaltigen Aspekte der Sicherheit zu achten, Transaktionen zu realisieren, Fehler zu kompensieren und Datenverluste trotz Kommunikationsproblemen zu verhindern. Hinzu kommt, dass immer häufiger Systeme miteinander kommunizieren müssen, die ursprünglich gar nicht dazu konzipiert wurden.

Mit der *Windows Communication Foundation* (WCF) hat Microsoft den Versuch unternommen, die unterschiedlichen Möglichkeiten, die es zur Lösung der aufgezeigten Herausforderungen gibt, unter einen Hut zu bringen. Außerdem steht nun mit der *ASP.NET Web API* eine schlanke Alternative zur WCF zur Auswahl, welche sich zwar auf bestimmte Arten der Kommunikation beschränkt, dafür aber auch einfacher einzusetzen ist.

Die Erfahrung hat jedoch gezeigt, dass man WCF und Web API nicht isoliert betrachten darf. Für die erfolgreiche Implementierung von verteilten Systemen sind weitere Frameworks notwendig, die mit WCF bzw. Web API in Einklang gebracht werden müssen. Aus diesem Grund behandelt das vorliegende Werk auch Technologien wie die *Workflow Foundation* (WF), die zur Automatisierung von Geschäftsprozessen dient, das *Entity Framework* (EF), die WCF Data Services und WCF RIA Services zum Zugriff auf Datenbanken, die *Windows Identity Foundation* (WIF) zur Realisierung von erweiterten Sicherheitsszenarien, die dem Ruf nach Single Sign-On-Lösungen gerecht werden, und den *Azure Service Bus*, welcher mittlerweile auch zur lokalen Installation als Windows Server Service Bus verfügbar ist, zur standortübergreifenden Integration verschiedener Systeme.

Wer ist die Zielgruppe dieses Buchs?

Dieses Buch wendet sich an Personen, die bereits mit .NET gearbeitet haben, und nun verteilte Systeme sowie damit einhergehende Services entwickeln oder auf technischer Ebene planen möchten. Ebenso werden Entwickler mit Erfahrung in den genannten Technologien adressiert, die das eine oder andere Thema vertiefen oder einfach nur bestimmte Aspekte

nachschlagen möchten. Die hier präsentierten Beispiele sind in C# gehalten, da C# die führende Sprache auf der .NET-Plattform darstellt. Allerdings finden sich im Web zahlreiche Tools zur automatischen Übersetzung in andere .NET-basierte Sprachen. Die verwendete C#-Version ist die Version 5; die verwendete Version von Visual Studio ist 2012. Um den Beispielen optimal folgen zu können, sollte der Leser ebenfalls die besagten Versionen oder neuere einsetzen. Sofern jedoch nicht explizit erwähnt ist, dass die beschriebenen Features .NET 4.5 benötigen, ist davon auszugehen, dass sie auch bei Verwendung von .NET 4.0 zur Verfügung stehen.

Wie soll dieses Buch gelesen werden?

Obwohl die einzelnen Kapitel des vorliegenden Werks miteinander korrelieren und deswegen auch immer wieder aufeinander verweisen, müssen sie nicht strikt sequentiell gelesen werden. Einsteigern empfehlen wir jedoch, zumindest das erste Kapitel zu bearbeiten, bevor sie sich den jeweiligen Interessensgebieten in den Folgekapiteln zuwenden. Lesern, die sich in die WCF einarbeiten möchten, sei auch Kapitel 2 und 3 ans Herz gelegt. Jene, die mit der ASP.NET Web API starten möchten, können nach dem Bearbeiten von Kapitel 1 mit den Kapiteln 6 und 7 fortfahren.

Einige Kapitel gehen davon aus, dass der Leser den Paket-Manager NuGet verwendet, da Microsoft bestimmte Bibliotheken vorzugsweise darüber zum Download anbietet. Deswegen befindet sich ein Überblick zu NuGet im Anhang.



Die einzelnen Abschnitte erläutern die behandelten Themen meist anhand von Beispielen. Gerade das Experimentieren mit fertigen Beispielen hat sich beim Erarbeiten neuer Stoffgebiete im Selbststudium bewährt. Die Beispiele zu diesem Buch finden Sie unter <http://downloads.hanser.de>.

Wir freuen uns darüber hinaus über Ihr Feedback und Ihre Fragen. Hierfür wurde unter www.it-visions.de/leser ein eigener Bereich eingerichtet.

Danksagung

Dank für ihre Mitwirkung und Unterstützung an diesem Buch möchten wir aussprechen an:

- unsere Familienangehörigen, die uns neben unserem Hauptberuf das Umfeld geschaffen haben, auch an manchen Abenden und Wochenenden an diesem Buch zu arbeiten.
- Frau Sieglinde Schärl vom Carl Hanser Verlag, die dieses Buchprojekt betreut hat.
- Frau Sandra Gottmann und Frau Julia Stepp, die das Buch sprachlich verbessert haben.
- Frau Irene Weillhart, die sich um die optischen Aspekte des Buchs gesorgt hat.

Über die Autoren

Manfred Steyer

FH-Prof. Manfred Steyer (www.softwarearchitekt.at) ist freiberuflicher Trainer und Berater bei www.IT-Visions.de. Darüber hinaus ist er für den Fachbereich Software Engineering der Studienrichtung IT und Wirtschaftsinformatik an der FH CAMPUS 02 in Graz (www.campus02.at) verantwortlich.

Er schreibt regelmäßig Fachartikel im .NET-Umfeld und ist Autor mehrerer Bücher. Manfred Steyer hat berufsbegleitend IT und IT-Marketing in Graz sowie Computer Science in Hagen studiert und kann auf mehr als zehn Jahre an Erfahrung in der Planung und Umsetzung von großen Applikationen zurückblicken. Er ist ausgebildeter Trainer für den Bereich der Erwachsenenbildung und spricht regelmäßig auf Fachkonferenzen.

In der Vergangenheit war Manfred Steyer mehrere Jahre für ein großes österreichisches Systemhaus tätig. In der Rolle als Entwicklungsleiter hat er gemeinsam mit seinem zehnköpfigen Team Geschäftsanwendungen konzipiert und umgesetzt.

Seine E-Mail-Adresse lautet manfred.steyer@softwarearchitekt.at. Auf Twitter findet man ihn unter <http://www.twitter.com/ManfredSteyer>.



Dr. Holger Schwichtenberg

Dr. Holger Schwichtenberg ist Leiter des .NET-Expertennetzwerks www.IT-Visions.de (<http://www.IT-Visions.de>), das zahlreiche Unternehmen in Europa durch Beratung, Schulung, Coaching und Support unterstützt. Zudem ist er Entwicklungsleiter bei der 5minds IT-Solutions GmbH & Co. KG, die Software im Kundenauftrag entwickelt. Die persönlichen Schwerpunkte von Dr. Holger Schwichtenberg sind Webanwendungen, verteilte Systeme und Datenbankzugriffe.

Dr. Holger Schwichtenberg gehört durch seine Auftritte auf nationalen und internationalen Fachkonferenzen sowie zahlreiche Fachbücher für Addison-Wesley, Microsoft Press und den Hanser Verlag zu den bekanntesten .NET-Experten in Deutschland. Darüber hinaus ist er ständiger Mitarbeiter der Fachzeitschriften *dotnet magazin*, *dotnetpro* und *iX* sowie bei heise.de. Von Microsoft ist er für sein .NET-Fachwissen als Microsoft Most Valuable Professional (MVP) für ASP.NET/IIS ausgezeichnet worden.

Sein Weblog finden Sie unter <http://www.dotnet-doktor.de>. Bei Twitter finden Sie ihn unter <http://www.twitter.com/DOTNETDOKTOR>.



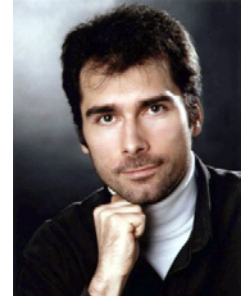
Matthias Fischer

Matthias Fischer ist seit einigen Jahren Softwareentwickler und -architekt. In seiner Tätigkeit als Technical Expert für .NET beschäftigt er sich insbesondere mit dem .NET-Framework, ASP.NET, WCF sowie Netzwerk- und Kommunikationstechnologien in C# und F#.

Des Weiteren ist Matthias Fischer zertifizierter Softwaretester mit Erfahrung im Embedded-Umfeld. Sein umfangreiches Wissen gibt er in diversen Usergroup-Vorträgen, auf Fachkonferenzen sowie als Berater und Fachbuchautor wieder.

Neben seiner beruflichen Tätigkeit beteiligt sich Matthias bei der Organisation und Ausrichtung der .NET Usergroup Berlin-Brandenburg.

Wenn es die Zeit neben den Projekten erlaubt, unternimmt Matthias gern Radtouren, oder er beschäftigt sich mit Fotografie.



Jörg Krause

Jörg Krause ist Partner des .NET-Expertenetzwerks HYPERLINK „<http://www.IT-Visions.de>“ www.IT-Visions.de. Er arbeitet als freier Fachautor, Trainer, Consultant und Softwareentwickler aus Leidenschaft. Schwerpunktthemen sind die Webapplikationen mit ASP.NET, Microsoft SQL Server sowie die SharePoint-Programmierung. Zu allen Themen hat er Bücher und zahlreiche Artikel in Fachzeitschriften verfasst und bietet individuelle Workshops und Trainings auf hohem Niveau an. Seit 1998 tritt er auf zahlreichen Konferenzen als Speaker auf.

Wenn er sich nicht mit Computern beschäftigt, was eher selten vorkommt, liest er bei schlechtem Wetter Bücher (Science-Fiction, Thriller) oder spielt bei besserem Wetter Golf und Badminton.



1

Serviceorientierung

Bevor es in den nachfolgenden Kapiteln um die Realisierung von Services mit Windows Communication Foundation (WCF) gehen wird, bietet dieses Kapitel einen kompakten Überblick zum Thema Serviceorientierung. Neben den unterschiedlichen Sichten auf dieses Thema wird auch auf Protokolle zur Realisierung von Services eingegangen.

■ 1.1 Konzeptionelle Ebene

Sowohl für Techniker als auch für Betriebswirte ist Serviceorientierung von Bedeutung. Allerdings haben beide Gruppen eine unterschiedliche Sicht auf diese Thematik. Dieser Abschnitt geht auf diese beiden Sichten ein und informiert darüber, was allgemein unter Service verstanden werden kann.

1.1.1 Betriebswirtschaftliche Sicht

Auf das Thema Serviceorientierte Architekturen (SOA) gibt es zwei Sichtweisen – eine betriebswirtschaftliche und eine technische. Aus betriebswirtschaftlicher Sicht helfen serviceorientierte Architekturen beim Unterstützen von Geschäftsprozessen. Einzelne Schritte eines Geschäftsprozesses werden dabei von Services, die von verschiedenen Systemen angeboten werden, realisiert. Zur Koordinierung dieser Services wird ein weiterer Service, ein sogenannter Prozessservice, eingesetzt. Dieser ist beispielsweise mit einer Workflow-Engine implementiert, sodass Anpassungen einfach möglich sind und die Umsetzung auch gleichzeitig die Dokumentation widerspiegelt. Bild 1.1 veranschaulicht dies. Dargestellt ist hier ein sehr einfacher, mit den Mitteln der *Business Process Modeling Notation* (BPMN, vgl. www.bpmn.org) modellierter Prozess, der eine Vorgehensweise zur Angebotslegung für Reisen beschreibt und in vier Bereiche geteilt ist. Diese Bereiche werden in der BPMN als *Pools* bezeichnet. Die Aktivitäten in den Pools *System*, *Fluggesellschaft* und *Hotelreservierung* werden idealerweise von Services implementiert. Diese können entweder innerhalb der eigenen Organisation zur Verfügung stehen oder von Geschäftspartnern angeboten werden. Ersteres ist bei den Aktivitäten im Pool *System* der Fall; Letzteres bei den Aktivitäten in den Pools

Fluggesellschaft und Hotelreservierung. Die Aktivitäten im Pool *Reisebüro* würden die Applikation darstellen, die durch Kombination der restlichen Services geschaffen werden soll, und der Prozess an sich würde mit den Mitteln einer Workflow-Engine, welche die einzelnen Services koordiniert, umgesetzt werden. Hierbei ist auch von *Orchestrierung* die Rede.

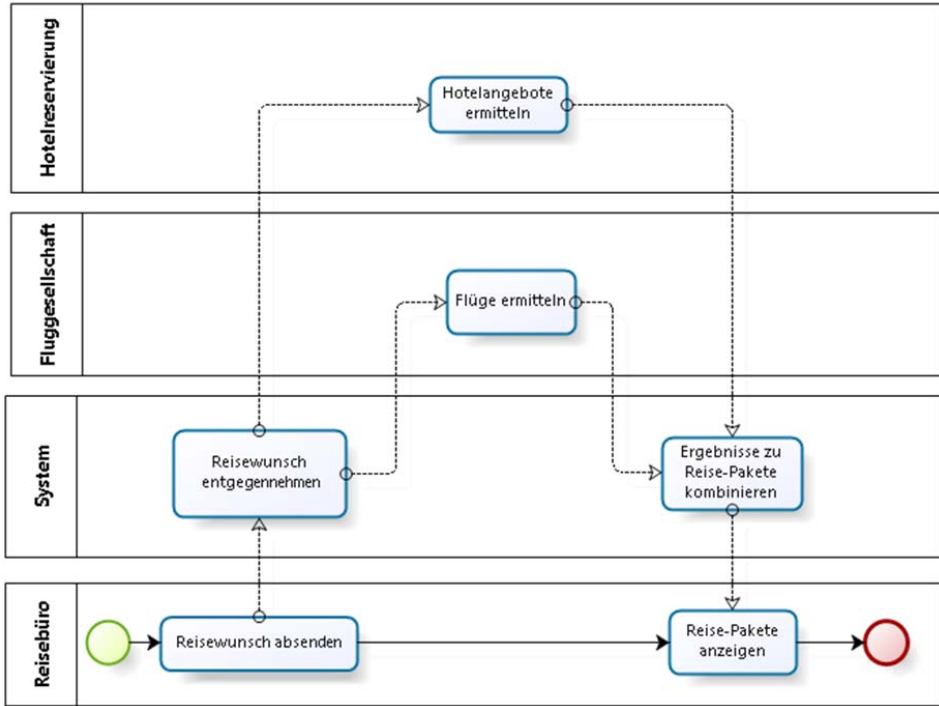


Bild 1.1 Beispielhafter Geschäftsprozess

1.1.2 Technische Sicht

Aus technischer Sicht geht es beim Thema SOA um verteilte Systeme und Systemintegration. Einzelne Dienste werden, meist über das Netzwerk, in Anspruch genommen. Dazu sendet ein Client eine Anfrage an einen Service. Dieser führt daraufhin die gewünschte Aufgabe durch und sendet das Ergebnis retour an den Client. Je nach Kommunikationspartner können dabei die zu verwendenden Protokolle und Datenformate variieren. Letztere gilt es bei Bedarf zu „übersetzen“, sodass sie in einer für das Gegenüber bearbeitbaren Form vorliegen. All diese Konzepte sind wahrscheinlich so alt wie das Konzept von Computernetzwerken und firmierten vor den Zeiten von SOA zuletzt unter dem Begriff *Enterprise Application Integration* (EAI) und *Komponentenorientierung*. Der Übergang zwischen diesen Paradigmen und SOA ist, nüchtern betrachtet, fließend, zumal auch die verwendeten Technologien großteils dieselben sind. SOA unterscheidet sich von seinen Vorgängern vor allem darin, dass der Aspekt der Interoperabilität, also das Zusammenspiel verschiedener Systeme, stärker im Vordergrund steht. Dies manifestiert sich in den bevorzugt verwendeten offenen Protokollen und Datenformaten, wie zum Beispiel HTTP, XML oder auch SOAP. Daneben

schlägt sich dieser Aspekt auch in den vorzufindenden Mustern und Best Practices wieder. So wird beispielsweise häufig auf serviceübergreifende Transaktionen verzichtet, da sich deren interoperable Implementierung in der Regel als sehr herausfordernd darstellt. Stattdessen werden im Fehlerfall Kompensationslogiken, welche die ursprünglichen Aktionen rückgängig machen, angestoßen. Bezogen auf das zuvor betrachtete Beispiel könnte es sich dabei um das Stornieren eines Fluges handeln.

1.1.3 Was ist ein Service?

Neben der Tatsache, dass es sich bei einem Service um eine Sammlung von Operationen, die von einzelnen Clients über ein Netzwerk konsumiert werden können, handelt, geht damit auch ein bestimmtes Mindset einher. Im Gegensatz zum klassischen Ansatz der *Remote Procedure Calls* (RPC) ist ein Service in sich geschlossen. Das bedeutet, dass das Innenleben eines Services für den Aufrufer eine Blackbox darstellt, sodass dieser nicht über technische Details Bescheid wissen muss. Aus diesem Grund sind Service-Operationen auch stark Use-Case-orientiert und selten technisch bzw. generisch. Dies führt auch dazu, dass Service-Operationen mitunter grobgranularer als herkömmliche Methoden sind. Ein gutes Beispiel hierfür bietet die nachfolgend dargestellte Operation `BucheFlug`.

```
public Ticket BucheFlug(Buchung buchung) { ... }
```

Einige dazu passende Negativbeispiele gehen aus Listing 1.1 hervor, zumal sich der Aufrufer hier mit technischen Konstrukten des Systems, wie zum Beispiel *FlugHandles* oder *SeatHandles*, beschäftigen muss. Gute Services würden hingegen fachliche Konstrukte verwenden, z. B. Flüge anstatt *FlugHandles*.

Diese Operationen könnte es im vorhin betrachteten Positivbeispiel zwar auch geben, allerdings wären diese hier zum einen nicht öffentlich zugänglich und würden zum anderen von der Operation `BucheFlug` koordiniert werden. Services mit solchen Koordinierungsmethoden werden auch als *Fassaden* bzw. *Servicefassaden* bezeichnet, da sie dahinter liegende Details verbergen. Dabei soll nicht verschwiegen werden, dass die Verwendung von Fassaden auch schon vor dem Aufkommen des Begriffs SOA gängige Praxis im Bereich verteilter Geschäftsanwendungen war. Der Vorteil dieser Vorgehensweise liegt neben der Tatsache, dass sich der Aufrufer nicht mit Interna belasten muss, in einer geringeren Netzwerkbelastung, da anstatt vieler kleiner Nachrichten wenige oder nur eine größere zu übersenden sind. Daneben führt dieses Muster dazu, dass die von der Fassade gekapselten Details ohne Anpassungen des Clients ausgetauscht werden können. Dies erlaubt es, die verwendeten Software-Systeme einfacher an sich ändernde Geschäftsregeln und Abläufe anzupassen, was in weiterer Folge auch die Agilität des Unternehmens erhöht. Hierbei ist auch von loser Kopplung die Rede, was im betrachteten Fall bedeutet, dass der Client möglichst wenig über den konsumierten Service wissen muss.

Listing 1.1 Negativbeispiele für Service-Operationen

```
public FlightHandle GetFlightHandle(string flugNummer, DateTime datum) { ... }  
public SeatHandle CheckFreeSeats(FlightHandle h) { ... }  
public bool CheckFlightCancelled(FlightHandle h) { ... }  
public bool CheckCustomerExists(String customerNumber) { ... }
```