

Grundlagen der Authentifizierung in der WCF

Erkennungsdienst

Die Authentifizierung ist der erste Schritt zu einem sicheren WCF-Service. Dabei findet sich der Entwickler aber schnell in einem Gewirr von möglichen Einstellungen und Konfigurationen wieder. Das hier vorgestellte Service-Framework beantwortet viele der Fragen und übernimmt praktischerweise gleich den Großteil der Konfiguration.

Auf einen Blick



Nikolai Solovev studiert Software-Engineering an der HTWG Konstanz. Im Rahmen seiner Diplomarbeit bei combit entwickelt er ein Service-Framework für die Reporting-Komponente List & Label.



Jochen Bartlau ist seit 1998 beim Konstanzer Softwareunternehmen combit beschäftigt. Als Projektleiter ist er zusammen mit seinem Team für die technische Weiterentwicklung von List & Label verantwortlich.

Inhalt

- › Überblick über die sechs Authentifizierungsverfahren und zwanzig Binding-Arten von WCF.
- › Digitale Zertifikate einsetzen.
- › Konfiguration der Authentifizierung über ein eigenes Framework erleichtern.



dnpCode

A0909AuthWCF

Wenn ein Entwickler die Funktionalität einer Applikation über das Netzwerk anbieten will, dann sind Webdienste die erste Wahl. Die Vorteile dieser Technologie sind insbesondere die Unabhängigkeit von Programmiersprachen und Plattformen sowie die Standardisierung durch zahlreiche Organisationen. Die von Microsoft bereitgestellte Kommunikationsplattform WCF (Windows Communication Foundation) bietet für die Entwicklung von Webdiensten ein umfangreiches API, das in Kombination mit Visual Studio (VS) die ersten Schritte zu einer Webdienstapplikation sehr erleichtert.

In der Regel möchte man die Funktionen des Dienstes aber nur bestimmten Clients zur Verfügung stellen. Dafür ist Authentifizierung erforderlich. Befasst man sich mit WCF-Authentifizierungsverfahren, kann man sich in dem Gewirr aus Bindings, Sicherheitsmodi und Authentifizierungsarten schnell etwas verloren fühlen. Die große Auswahl der Konfigurationsmöglichkeiten ist aber durchaus beabsichtigt und auch nötig, um sich den Besonderheiten verschiedener Anwendungsszenarien anzupassen. Es macht einen großen Unterschied, ob der Dienst und der Client über das Internet oder über das Intranet miteinander kommunizieren und ob sie auf demselben oder auf verschiedenen Rechnern ausgeführt werden et cetera.

Dieser Artikel klärt zuerst einige Grundbegriffe, um dann die Bindings und die Einstellungen für verschiedene Authentifizierungsmethoden unter die Lupe zu nehmen. Je nach Authentifizierungsart muss der Client verschiedene Identitätsinformationen bereitstellen. Es wird gezeigt, um welche Informationen es sich handelt und wie diese übergeben werden. Schließlich werden drei grundlegende Anwendungsszenarien beschrieben, ihre optimalen Einstellungen definiert und ein Framework vorgestellt, das die Konfiguration eines Dienstes erleichtert.

Was ist Authentifizierung?

Authentifizierung bezeichnet den Versuch, eine präsentierte Identität zu bestätigen. Bei einem Webdienst werden die Identitäten von Clients und Diensten verifiziert. Für den Dienst ist es daher notwendig zu wissen, wer eine Anfrage stellt.

Erst mit der korrekten Bestimmung der Identität kann dieser entscheiden, welche Funktionen der Client nutzen darf. Eine Identität kann im Allgemeinen auf zwei verschiedenen Wegen bestätigt werden [1]:

- Zum einen durch die Kenntnis einer bestimmten Information. Dazu gehören Verfahren, bei denen ein Passwort oder eine PIN benötigt wird.
- Zum anderen durch den Besitz eines Objektes, wie zum Beispiel eines digitalen Zertifikats.

WCF bietet dem Entwickler insgesamt sechs Authentifizierungsverfahren zur Auswahl:

- **None:** Es wird keine Authentifizierung durchgeführt.
- **Windows:** Diese Methode verwendet die Windows-Anmeldeinformationen für die Authentifizierung. Befindet sich der Client in einer Arbeitsgruppe, wird NTLM (NT LAN Manager) als Authentifizierungsverfahren eingesetzt. Steht ein Windows Domain Server zur Verfügung, wird die Kerberos-Authentifizierung durchgeführt.
- **Benutzername:** Benutzername und Passwort werden an den Server übertragen. Daraufhin vergleicht dieser die Informationen mit einer Datenbank oder den Windows-Konten.
- **Zertifikat:** Um sich zu authentifizieren, verwendet der Client ein Zertifikat, welchem der Server vertrauen muss.
- **Benutzerdefiniert:** Der Entwickler implementiert ein eigenes Authentifizierungsverfahren.
- **Issued Token:** Hier wird eine dritte Partei für die Authentifizierung herangezogen. STS (Security Token Service) authentifiziert den Client und sendet ihm einen SAML-Token (Security Assertion Markup Language). Der Client authentifiziert sich beim Dienst mithilfe des erhaltenen Tokens. Die Voraussetzung dafür ist, dass der Dienst dem Token vertraut, was durch eine digitale Signatur sichergestellt werden kann.

Die benutzerdefinierte und die Issued-Token-Authentifizierung gehören zu den fortgeschrittenen Techniken im WCF. Ihre Erläuterung würde den Rahmen des Artikels sprengen, sodass sie hier nicht weiter behandelt werden.

Tabelle 1

Mögliche Einstellungen für die Arten der Benutzeranmeldung.

Binding	Transport	Message	TransportWithMessageCredential	TransportCredentialOnly
basicHttpBinding	N, B, D, NT, W, C	C	U, C	N, B, D, NT, W
basicHttpContextBinding	N, B, D, NT, W, C	C	U, C	N, B, D, NT, W
netTcpBinding	N, W, C	N, W, U, C, I	W, U, C, I	–
netTcpContextBinding	N, W, C	N, W, U, C, I	W, U, C, I	–
webHttpBinding	N, B, D, NT, W, C	–	–	N, B, D, NT, W
ws2007FederationHttpBinding	–	I	I	–
ws2007HttpBinding	N, B, D, NT, W, C	N, W, U, C, I	W, U, C, I	–
wsDualHttpBinding	–	N, W, U, C, I	–	–
wsFederationHttpBinding	–	I	I	–
wsHttpBinding	N, B, D, NT, W, C	N, W, U, C, I	W, U, C, I	–
wsHttpContextBinding	N, B, D, NT, W, C	N, W, U, C, I	W, U, C, I	–

N: None, B: Basic, D: Digest, NT: NTLM, W: Windows, C: Certificate, U: Username, I: IssuedToken

Die Art der Authentifizierung kann der Entwickler administrativ oder im Code festlegen. Bei administrativer Konfiguration werden die Einstellungen in einer Konfigurationsdatei *app.config* eingetragen. Listing 1 zeigt, wie Sie die Art der Authentifizierung in WCF für einen Dienst festlegen. Die gleichen Einstellungen werden einmal im Code und das andere Mal in der Konfigurationsdatei vorgenommen. Auf der Heft-DVD finden Sie zwei Projekte, die beide Möglichkeiten demonstrieren und Ihnen dabei helfen, ein Konfigurationsattribut dem richtigen Codeschnipsel zuzuordnen.

Binding-Dschungel

Die Art der Authentifizierung wird in der Binding-Konfiguration durch die Eigenschaft *clientCredentialsType* festgelegt, siehe Tabelle 1. Ein Binding ist eine Sammlung von Einstellungen für die Transportprotokolle und Nachrichtencodierungen eines Endpunkts, die das Senden und Empfangen von Nachrichten übernehmen. Insgesamt bietet WCF zwanzig verschiedene Binding-Arten zur Auswahl an, die sich in mehrere Gruppen unterteilen lassen:

- Alle Bindings mit *mex*-Präfix sind für den Austausch der Metainformationen konzipiert.
- Die *basic*- und *ws*-Bindings zeichnen sich durch eine hohe Interoperabilität aus.
- Ist sichergestellt, dass der Dienst und der Client mindestens die Version 3.0 des .NET Frameworks verwenden, wird ein *net*-Binding eingesetzt.

Die Bindings aus den drei Gruppen lassen sich weiter nach der Transportmethode unterscheiden: WCF bietet TCP, HTTP/HTTPS, Named Pipes und MSMQ

(Microsoft Message Queuing) für die Übertragung der Nachrichten an. Die TCP-Bindings werden für die Kommunikation im Intranet verwendet, Dienst und Client befinden sich hierbei innerhalb der Firewall. Die Kommunikation über das Internet wird mit HTTP oder HTTPS realisiert. Dieser Transportweg eignet sich besonders dafür, da er in der Regel von Firewalls nicht geblockt wird. Werden die Dienst- und die Clientanwendung auf demselben Rechner ausgeführt, kann über Named Pipes kommuniziert werden. Will man eine asynchrone Nachrichtenübertragung realisieren, stehen MSMQ-Bindings zur Verfügung.

Die Authentifizierung kann nur korrekt funktionieren, wenn sichergestellt ist, dass die erhaltenen Nachrichten nicht verändert wurden. Außerdem sollte ein unbefugtes Lesen der enthaltenen Informationen nicht möglich sein. WCF bietet für das Signieren und Verschlüsseln der Nachrichten folgende Sicherheitsmodi: *None*, *Transport*, *Message*, *TransportWithMessageCredential*, *TransportCredentialOnly* und *Both*, siehe Tabelle 2. Die Auswahl der Transfermethode hängt jedoch von der Art der Bindung ab, die man für seinen Dienst sucht. Denn nicht jedes Binding unterstützt alle Optionen, siehe Tabelle 3.

Listing 1

Authentifizierung aktivieren.

```
// Administrativ
<message clientCredentialType="UserName" />

// Im Programmcode
binding.Security.Message.ClientCredentialType = MessageCredentialType.UserName;
```

Tabelle 2

Die Sicherheitsmodi von WCF.

Transfersicherheitsmethode	Beschreibung
None	Es wird keine Verschlüsselung durchgeführt.
Transport	Es wird ein sicherer Transportkanal verwendet, z. B. HTTPS. Die Aushandlung sicherheitsrelevanter Aspekte wird automatisch durch das Protokoll abgewickelt.
Message	Die zu übertragende Nachricht wird verschlüsselt und signiert. Dadurch kann eine sichere Übertragung über einen unsicheren Transportkanal stattfinden.
TransportWithMessageCredential	Eine gemischte Methode der Verschlüsselung. Transportsicherheit wird für die Authentifizierung und Sicherheit der Nachricht verwendet. Die Benutzeranmeldeinformationen werden auf der Nachrichtenebene für die Übertragung verschlüsselt.
TransportCredentialOnly	Die Nachrichten werden nicht verschlüsselt oder signiert. Die Authentifizierung erfolgt über die HTTP-Mechanismen.
Both	Es werden beide Methoden der Verschlüsselung angewendet.

Tabelle 3

Mögliche Einstellungen für die Transfersicherheit.

Binding	None	Transport	Message	TransportWithMessage-Credential	TransportCredentialOnly	Both
basicHttpBinding	X	X	X	X	X	-
basicHttpContextBinding	X	X	X	X	X	-
msmqIntegrationBinding	X	X	-	-	-	-
netMsmqBinding	X	X	X	-	-	X
netNamedPipeBinding	X	X	-	-	-	-
netPeerTcpBinding	X	X	X	X	-	-
netTcpBinding	X	X	X	X	-	-
netTcpContextBinding	X	X	X	X	-	-
webHttpBinding	X	X	-	-	X	-
ws2007FederationHttpBinding	X	-	X	X	-	-
ws2007HttpBinding	X	X	X	X	-	-
wsDualHttpBinding	X	-	X	-	-	-
wsFederationHttpBinding	X	-	X	X	-	-
wsHttpBinding	X	X	X	X	-	-
wsHttpContextBinding	X	X	X	X	-	-

Die Kunst der Konfiguration

Hat man sich für eine Authentifizierungsmethode entschieden, gibt es je nach Wahl weitere Einstellungsmöglichkeiten. Zudem unterscheidet sich die Art der Informationen, die der Client bereitstellen muss.

Wird keine Authentifizierung durchgeführt, werden auch keine Identitätsinfor-

mationen übertragen. Jeder Client ist in der Lage, sich mit dem Dienst zu verbinden und sämtliche angebotenen Funktionen zu nutzen.

Bei der Windows-Authentifizierung müssen der Windows-Kontenname und das Passwort an den Server übertragen werden. Befinden sich Server und Client in der

gleichen Domain oder Arbeitsgruppe, können die Informationen automatisch dem aktiven Prozess entnommen werden. Ist das nicht der Fall, muss man sie explizit angeben, siehe Listing 2. Zusätzlich kann eine anonyme Anmeldung für den Dienst eingestellt werden.

Bei der Authentifizierung mit Benutzernamen kann die Datenbasis, mit der die übergebenen Benutzerinformationen verglichen werden, eingestellt werden. Dabei kann man zwischen *Windows*, *Custom* und *MembershipProvider* wählen. Listing 3 zeigt, wie man eine Authentifizierung mit Benutzernamen so konfiguriert, dass man die Daten mit den vorhandenen Windows-Konten abgleicht.

Auf der Clientseite müssen Benutzername und Passwort bei dieser Form der Authentifizierung explizit mit übergeben werden, siehe Listing 4. Weitere Einstellungen des Endpunktes müssen mit denen auf der Seite des Dienstes übereinstimmen.

Aus Sicherheitsgründen verhindert WCF eine unverschlüsselte Übertragung von Benutzeranmeldeinformationen, weshalb bei der Authentifizierung per Benutzername ein digitales Zertifikat benötigt wird.

Ein digitales Zertifikat verwenden

Im Rahmen der Authentifizierung wird ein digitales Zertifikat für die Verschlüsselung und Signierung der Kommunikation, aber auch für die Authentifizierung selbst verwendet.

Der erste Schritt beim Einsatz eines digitalen Zertifikats ist dessen Beschaffung. Dabei gibt es zwei Wege, die zum Ziel führen – die Benutzung des VS-Tools *make-*

Listing 2

Domainübergreifende Anmeldung.

```
ServiceClient client;
client.ClientCredentials.Windows.ClientCredential.Domain = "test.domain.com";
client.ClientCredentials.Windows.ClientCredential.UserName = "Username";
client.ClientCredentials.Windows.ClientCredential.Password = "password";
```

Listing 3

Konfiguration der Authentifizierung.

```
//Administrativ
<userNameAuthentication userNamePasswordValidationMode="Windows"/>

//Programmgesteuert
tempHost.Credentials.UserNameAuthentication.UserNamePasswordValidationMode =
UserNamePasswordValidationMode.Windows;
```

Listing 4

Mit Benutzernamen anmelden.

```
ServiceClient client;
client.ClientCredentials.UserName.UserName = "Username";
client.ClientCredentials.UserName.Password = "password";
client.Open();
```

Listing 5

Angabe des Zertifikats.

```
// Administrativ
<serviceCertificate findValue="TestCert" storeLocation="LocalMachine"
storeName="My" x509FindType="FindBySubjectName"/>

// Programmgesteuert
tempHost.Credentials.ServiceCertificate.SetCertificate(StoreLocation.LocalMachine,
StoreName.My, X509FindType.FindBySubjectName, "TestCert");
```

Listing 6

Benutzerdefinierte Validierung des Zertifikats.

```
// Administrativ
<authentication certificateValidationMode="Custom"
customCertificateValidatorType="Client.MyCertValidator, Client"/>

// Programmgesteuert
tempProxy.ClientCredentials.ServiceCertificate.Authentication.Certificate-
ValidationMode = X509CertificateValidationMode.Custom;

tempProxy.ClientCredentials.ServiceCertificate.Authentication.CustomCertificate-
Validator = new MyCertValidator();
```

`cert.exe` ist die einfachste und schnellste Möglichkeit, ein Zertifikat zu generieren. Hierzu geben Sie den folgenden Befehl in den VS Command Prompt ein:

```
makecert.exe -sr LocalMachine -ss My -pe
-sky exchange -n "CN=TestCert" TestCert.cer
-a sha1
```

Dadurch wird das Zertifikat generiert und automatisch im Zertifikatsspeicher abgelegt. Aber Vorsicht: Führen Sie den Aufruf mehrere Male mit demselben Zertifikatsnamen aus, lässt sich der Dienst nicht mehr starten, und eine *InvalidOperationException* weist darauf hin, dass mehrere Zertifikate unter dem angegebenen Namen gefunden wurden. Am einfachsten löscht man die unnötigen Kopien über die Microsoft Management Console (MMC).

Im nächsten Schritt wird der Dienst so konfiguriert, dass er das erstellte Zertifikat für die Verschlüsselung der Nachrichten verwendet. Hierfür werden der Ort und der Ordnername im Zertifikatsspeicher benötigt sowie der Name der Eigenschaft, nach der gesucht werden soll, und zuletzt der Suchbegriff, siehe Listing 5.

Da es sich nur um ein Testzertifikat handelt, muss man auf dem Client dafür sorgen, dass das generierte Zertifikat als vertrauenswürdig eingestuft wird. Dazu erstellt man eine eigene Klasse *MyCertValidator*, die

von der *X509CertificateValidator*-Klasse erbt und ihre *Validate*-Methode überschreibt. Zuletzt muss man die eigene Klasse als Validierer eintragen, siehe Listing 6.

Alternativ kann auf die eigene Validator-Klasse verzichtet werden. Dazu wird das verwendete Zertifikat in den Zertifikatsspeicher des Clients kopiert, und zwar in den Ordner *Vertrauenswürdige Personen*, und *certificateValidationMode* auf *PeerTrust* gesetzt.

Soll das Zertifikat nicht nur für die Verschlüsselung, sondern für die Authentifizierung selbst zum Einsatz kommen, bedarf es weiterer Einstellungen. Ähnlich wie oben beschrieben, wird auf der Clientseite das Zertifikat eingestellt, welches verwendet werden soll. Dieses Zertifikat sollte eigens für den Client generiert worden sein und sich in dessen Zertifikatsspeicher befinden. Auf der Serverseite werden abschließend die Vertrauenswürdigkeitsprüfung des Clientzertifikats und die Authentifizierungsart eingestellt, siehe Listing 7.

Mit *makecert* generierte Zertifikate sind nur für Testzwecke geeignet und können nicht für den kommerziellen Betrieb genutzt werden. Ein gültiges Zertifikat erhält man von einer Zertifizierungsstelle, die Dienstleistung ist jedoch im Regelfall kostenpflichtig, weil die Identität des Antragstellers überprüft werden muss. Möchte

man das Zertifikat gewerblich nutzen, empfiehlt es sich, einen von der Bundesnetzagentur akkreditierten Anbieter auszusuchen. Eine Liste mit detaillierten Informationen kann auf der Homepage der Behörde [2] eingesehen werden.

Einstieg leicht gemacht

Wie bisher deutlich wurde, gibt es bei der Einrichtung der Sicherheitsmechanismen einiges zu beachten. Aus der Anzahl der Konfigurationsparameter ergeben sich sehr viele Kombinationsmöglichkeiten [3]. Manche Varianten machen jedoch keinen Sinn, und andere sind schlichtweg nicht kompatibel. Der Einstieg kann sich für Entwickler ohne Vorkenntnisse mühsam gestalten, und es ist schwierig, einen einmal gewählten Ansatz später zu ändern. Das Design des Dienstes nach den Anforderungen typischer Anwendungsszenarien erleichtert die Suche nach den richtigen Einstellungen.

Konfiguration per Framework

Auf der Heft-DVD steht ein einfaches Framework zur Verfügung, das im Folgenden beschrieben wird. Damit ist es nicht mehr nötig, die einzelnen Einstellungen der Bindings und der Authentifizierung manuell festzulegen. Das Framework wird durch die Angabe einer Adresse, unter welcher

Listing 7

Authentifizierung mit einem Zertifikat.

```
//Client
...
<endpointBehaviors>
...
<clientCredentials>
<clientCertificate findValue="ClientCert"
storeLocation="LocalMachine"
storeName="My"
x509FindType="FindBySubjectName"/>
</clientCredentials>
...
</endpointBehaviors>
...

//Server
<serviceCredentials>
<clientCertificate>
<authentication certificateValidation-
ationMode="PeerTrust"/>
</clientCertificate>
</serviceCredentials>
...
<message
clientCredentialType="Certificate"/>
...
```

Listing 8

Konfiguration mithilfe des Frameworks.

```
// Dienst
string address = "http://localhost:8000/Service";
ServiceContract, Contract> service = new ServiceContract, Contract> (address);
SecurityConfiguration _config = SecurityConfiguration.Intranet;

Endpoint endpoint = new Endpoint(config);
service.AddEndpoint(endpoint);

service.Start();

// Client
SecurityConfiguration _config = SecurityConfiguration.Intranet;

MyProxy _proxy = new MyProxy(_config, address);
proxy.SetCredentials("test.domain.net", "username", "password");
proxy.Open();
proxy.MyFunction();
```

der Dienst erreichbar sein sollte, und durch einen Parameter, welcher eines der drei unterstützten Szenarien – keine Sicherheit, Internet, Intranet – angibt, konfiguriert, siehe Listing 8. Das gleiche Prinzip wird auf der Clientseite umgesetzt. Die Implementierung eines sicheren Dienstes wird dadurch zum Kinderspiel. Abbildung 1 zeigt die auf dem Framework aufbauende Beispielapplikation.

Keine Sicherheit

Es kann durchaus sinnvoll sein, auf Sicherheit komplett zu verzichten. Soll der Dienst möglichst viele Arten von Clients akzeptieren und ihnen den vollen Zugriff auf die Funktionen erlauben, sind eine Authenti-

fizierung und Autorisierung nicht notwendig. Jedes WCF-Binding unterstützt diesen Modus, das *wsHttpBinding* bietet jedoch die besten Voraussetzungen für dieses Szenario. Zum einen ist es für die höchstmögliche Interoperabilität ausgelegt, zum anderen ermöglicht dieses Binding den Zugriff auf den Dienst über Internet, Intranet und deckt auch den Fall ab, dass Client und Dienst auf demselben PC ausgeführt werden. Der Dienst wird für dieses Szenario konfiguriert, indem man den Sicherheitsmodus des Bindings auf *None* einstellt. Weitere Einstellungen sind nicht nötig.

Intranet

Im Intranetszenario befinden sich Client und Server und in einem nicht-öffentlichen Netz. Meistens verwenden die Teilnehmer dasselbe Betriebssystem, sodass Kerberos oder NTLM für die Authentifizierung der Clients verwendet werden können. Entwickelt man mit WCF einen Dienst für das Intranet, liegt es nahe, auch die Clientanwendungen mit .NET zu implementieren. Für die Kommunikation zwischen den Teilnehmern bietet sich das TCP/IP-Protokoll an. Das Framework wählt in dieser Konfiguration ein *netTcpBinding* aus. Dieses unterstützt die Windows-Authentifizierung und verwendet das gewünschte Protokoll für den Transfer der Nachrichten.

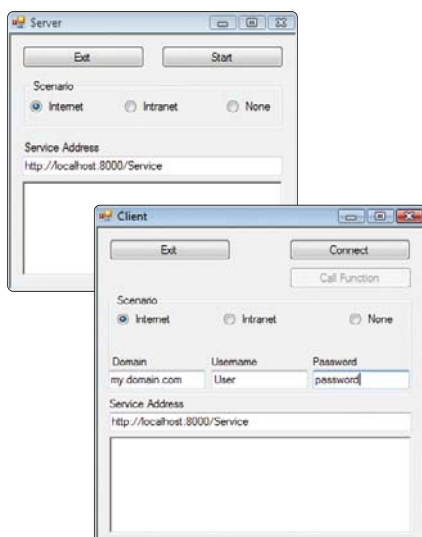
Internet

Im Internet basiert die Implementierung von Webdiensten und Clients oft auf unterschiedlichen Technologien. Aus diesem Grund sollte beim Design ein möglichst

hoher Grad an Interoperabilität angestrebt werden. Eine Authentifizierung mit Benutzernamen und Passwort ist für Webdienste im Internet am geeignetsten. Die Nachrichten passieren Firewalls und zahlreiche unbekannte Zwischenstationen, somit sollte auf Integrität und Authentizität der ankommenden Informationen geachtet werden. Das *wsHttpBinding* erfüllt die genannten Kriterien am besten. Die standardisierten WS*-Mechanismen garantieren die Kompatibilität mit anderen Webdiensttechnologien. Es ist möglich, die Nachrichten mit einem digitalen Zertifikat zu verschlüsseln und zu signieren, was den Sicherheitsaspekt bei der Übertragung abdeckt. Der Transport der Nachrichten wird über HTTP abgewickelt und somit in der Regel von den Firewalls nicht geblockt. Die Authentifizierung per Benutzername ermöglicht die Verbindung von Clients, die nicht auf Windows-Systemen ausgeführt werden.

Fazit

Analysiert man die Umgebung und die Anforderungen an einen WCF-Dienst, lassen sich leicht weitere Szenarien und die dazu passenden Einstellungen definieren, mit denen das vorgestellte Framework erweitert werden kann. Dieses szenarienbasierte Vorgehen ist – auch ohne Framework – bei der Suche nach der richtigen Konfiguration für einen Dienst sehr hilfreich. Durch Ausschlussverfahren reduziert sich die Anzahl an möglichen Kombinationen rasch und deutlich, sodass sich der Entwickler schnell mit den eigentlich wichtigen Dingen beschäftigen kann: dem Implementieren der Dienstfunktionalitäten. **[ml]**



[Abb. 1] Die Beispielanwendung auf der Heft-DVD nutzt das im Artikel beschriebene Framework.

- [1] John Brainard et al., Fourth-Factor Authentication: Somebody You Know, Proceedings of the 13th ACM Conference on Computer and Communications Security, 2006
- [2] www.bundesnetzagentur.de
- [3] Juval Löwy, Declarative WCF Security, MSDN Magazine 08/2007, **dnplink** SL0909AuthWCF1
- [4] Dominick Baier: Claims abstecken, Claimbasierte Anwendungen mit dem Geneva-Framework implementieren, dotnetpro 07/2009, Seite 76ff. **dnpcode** A0907Geneva
- [5] Milan Spalek: Wer ist das? Was darf der? Das Identitäts-Framework Geneva für Authentifizierung und Autorisierung nutzen, dotnetpro 05/2009, Seite 84ff. **dnpcode** A0905STS
- [6] Klaus Aschenbrenner: Rundum-Türsteher-Service, Anspruchsbasierte Zugangssicherung mit dem Geneva-Framework, dotnetpro 03/2009, Seite 124ff. **dnpcode** A0903Geneva