Kapitel 8

# Konfiguration

**In diesem Kapitel:**





Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.0 mit Visual C# 2010 - Das Entwicklerbuch,
Microsoft Press 2011 (ISBN 978-3-86645-530-6)

Die Konfiguration einer Webanwendung wird in ASP.NET primär durch Einstellungen in XML-basierten Konfigurationsdaten vorgenommen. Weiterhin wirken auch einzelne Einstellungen in der IIS-Konsole auf ASP.NET. Eine starke Verbesserung in ASP.NET seit Version 2.0 ist, dass verschiedene Administrationswerkzeuge existieren, die eine direkte Manipulation der Konfigurationsdateien oft nicht mehr notwendig machen. In Verbindung mit IIS 7.x können sehr viele Einstellungen für ASP.NET über die IIS-Konsole erfolgen. Außerdem nutzt IIS 7.x die ASP.NET-Konfigurationsdateien selbst zur Speicherung von Einstellungen.

**HINWEIS** Die ASP.NET-Konfigurationsdateien bestehen aus zahlreichen Konfigurationselementen, die im Umfang dieses Buchs nicht vollständig beschrieben werden können. Sie werden aber Hinweise auf einzelne Konfigurationseinstellungen in den entsprechenden Kapiteln finden.

# XML-Konfigurationsdateien

Wie in .NET Framework üblich, werden auch in ASP.NET alle Konfigurationsinformationen in XML-Datenstrukturen als Dateien im Dateisystem gespeichert. In den Konfigurationsdateien liegen primär Informationen, die ASP.NET Page Framework steuern. Ein Entwickler hat aber auch die Möglichkeit, eigene Daten hier abzulegen.

Für ASP.NET gibt es drei wichtige XML-Konfigurationsdateien:

- *machine.config* (Systemebene)
- *web.config* (Systemebene)
- *web.config* (Webanwendungsebene oder Anwendungsteil)

**HINWEIS** ASP.NET profitiert seit Version 2.0 von den neuen Möglichkeiten in .NET zur Speicherung von Verbindungszeichenfolgen und zur Verschlüsselung von Konfigurationselementen.

## web.config-Dateien

Konfigurationsdateien für einzelne ASP.NET-Anwendungen tragen immer den Namen *web.config*. Mittels dieser lassen sich viele Konfigurationseinstellungen der systemweiten Konfigurationsdateien überschreiben.

Es kann pro Webanwendung eine oder mehrere *web.config*-Dateien geben. In jedem Verzeichnis der Webanwendung ist eine solche Datei erlaubt. Die Konfigurationsdateien gelten für das jeweilige Dateisystemverzeichnis und alle untergeordneten Verzeichnisse. Einzelne Einstellungen werden nach unten vererbt, das heißt es gelten Einstellungen aus übergeordneten *web.config*-Dateien, falls in untergeordneten *web.config*-Dateien eine entsprechende Einstellung nicht existiert. An der Spitze der Hierarchie steht die *machine.config*-Datei.

Eine *web.config*-Datei kann innerhalb der Website, einer einzelnen Anwendung oder aber auch in beliebigen Unterverzeichnissen liegen. Die Definition einer *web.config*-Datei ist nicht Pflicht. Wird keine entsprechende Datei verwendet, so werden alle Eigenschaften der globalen Dateien unverändert übernommen. Zwar ist es nur möglich, eine *web.config*-Datei pro Verzeichnis zu verwenden; innerhalb von Unterverzeichnissen dürfen allerdings ebenfalls entsprechende Dateien eingesetzt werden. Da es sich dabei um einen hierarchischen Konfigurationsmechanismus handelt, werden alle Einstellungen der Konfigurationsdateien übergeordneter Verzeichnisse übernommen und gegebenenfalls überschrieben.

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.0 mit Visual C# 2010 - Das Entwicklerbuch,
Microsoft Press 2011 (ISBN 978-3-86645-530-6)

Führt man zur Laufzeit der ASP.NET-Anwendungen Änderungen an dieser Datei durch, so werden diese in der Regel sofort erkannt und übernommen. Ein Neustart ist dafür nicht notwendig. Dieses Verhalten trifft für (fast) alle Einstellungen zu, allerdings gibt es auch einige Ausnahmen, auf die an anderer Stelle noch eingegangen wird.

## Größe der Konfigurationsdateien im Standard

Nach ASP.NET 2.0 ist die zentrale *web.config*-Datei im Stammverzeichnis einer ASP.NET-Webanwendung immer größer und unübersichtlicher geworden. Sie umfasste in ASP.NET 3.5 SP1 unter Visual Studio 2008 im Standard ganze 168 Zeilen. Der Grund dafür ist, dass Microsoft ASP.NET AJAX, ASP.NET 3.5 und ASP.NET 3.5 SP1 derart auf Basis von ASP.NET 2.0 implementiert hat, dass die neuen und geänderten Funktionen durch Konfigurationsänderungen aktiviert wurden.

Beim Anlegen eines ASP.NET 4.0-Webprojekts sieht man nun schon auf den ersten Blick die aufgeräumte *web.config*-Datei. Der Weg dahin war letztlich ganz einfach: Microsoft hatte alle Änderungen seit ASP.NET 2.0 zum Standard erklärt. Man kann das vergleichen mit einem Microsoft Word-Dokument mit Änderungsverfolgung: Bisher waren die Änderungen sichtbar. Jetzt hat Microsoft auf *Alle Änderungen übernehmen* geklickt.

```
<?xml version="1.0"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
    <system.web>
        <compilation debug="false" targetFramework="4.0" />
    </system.web>
</configuration>
```

**Listing 8.1** Konfigurationsdatei im Projekttyp *Leere ASP.NET-Website* (*ASP.NET Empty Website*) unter .NET Framework 4.0

**HINWEIS** Wenn Sie in Visual Studio 2010 nicht *Leere ASP.NET-Website* (*ASP.NET Empty Website)* wählen, sondern *ASP.NET-Website* (*ASP.NET Web Site*), finden Sie eine größere *web.config*-Datei mit 52 Zeilen vor. Dies sind Einstellungen für die Benutzerverwaltung, die diese Vorlage nun schon vorbereitet enthält.

## Globale Konfigurationsdateien

In ASP.NET gibt es pro Computer zwei globale Konfigurationsdateien in *%Systemroot%\Microsoft.NET\Framework\v4.0.…\CONFIG*:

- *machine.config* enthält globale Einstellungen für alle Anwendungsarten
- *web.config* enthält globale Einstellungen speziell für Webanwendungen

In diesen Dateien erhält man neben einer guten Übersicht über alle vorhanden Konfigurationselemente, die Klassen, die für die entsprechende Handhabung verantwortlich sind. Ein Blick in die globalen Konfigurationsdateien bietet die Möglichkeit, die Grundkonfiguration von ASP.NET-Anwendungen zu untersuchen.

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.0 mit Visual C# 2010 - Das Entwicklerbuch,
Microsoft Press 2011 (ISBN 978-3-86645-530-6)

^

**HINWEIS** In .NET Framework 3.0 und 3.5 werden diese Dateien noch im Installationsverzeichnis von .NET Framework 2.0 gespeichert. Zur Erinnerung: .NET 3.0 und 3.5 sind Obermengen von .NET 2.0. Viele Mechanismen von .NET 2.0 arbeiten im Untergrund von .NET 3.0 und 3.5 weiter wie in .NET 2.0.

**TIPP** Hinweise auf mögliche Optionswerte findet man in den Dateien *machine.config.comments* und *web.config.comments* im gleichen Verzeichnis.

**ACHTUNG** Im Test kam es auf mehreren Systemen zu Problemen bei der Vererbung von globalen Einstellungen, wenn ASP.NET Development Webserver verwendet wurde. Beispielsweise waren globale Verbindungszeichen in den Webanwendungen nicht sichtbar. Ob dies ein definiertes Verhalten oder ein Bug ist, konnte bis zum Redaktionsschluss des Buchs nicht geklärt werden. Im Zweifel wenden Sie sich bitte an den Microsoft-Support.

# Werkzeugunterstützung

VWD bietet IntelliSense-Eingabeunterstützung für die Konfigurationsdateien.

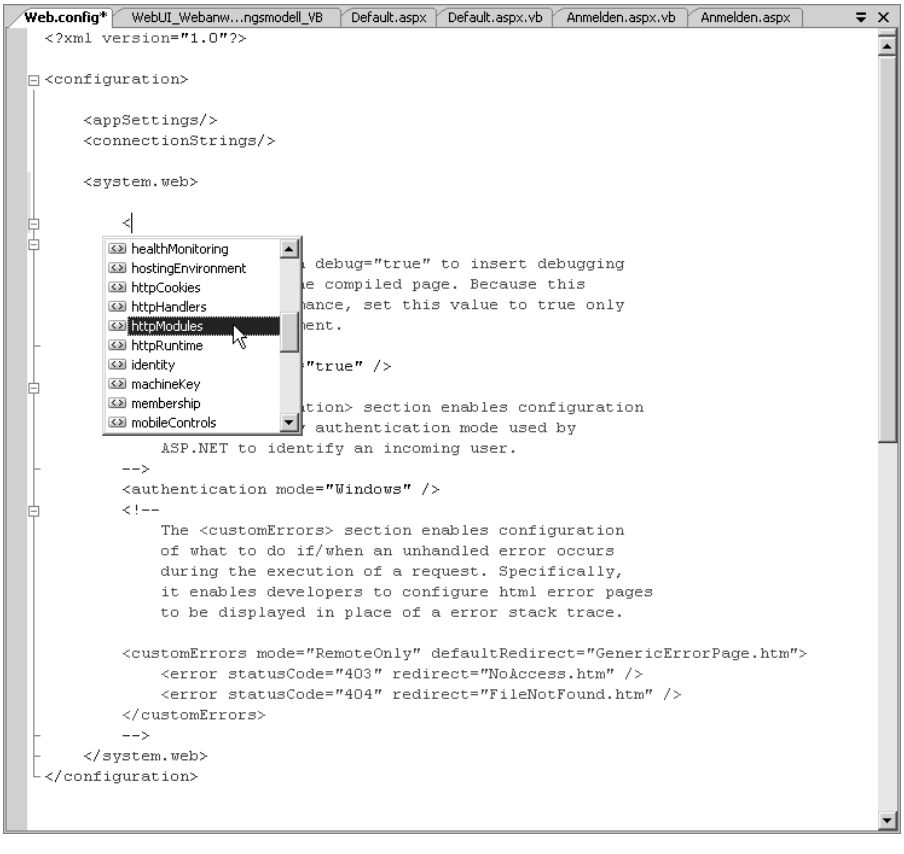



**Abbildung 8.1** IntelliSense für *web.config* in VWD

# Konfigurationsebenen

Für ASP.NET können Konfigurationen hierarchisch auf den folgenden Ebenen definiert werden: Maschine → Website → Anwendung → Verzeichnis. Jede Ebene übernimmt grundsätzlich die Konfiguration ihrer übergeordneten Ebene, allerdings können die meisten Einstellungen auf jeder Ebene überschrieben werden.

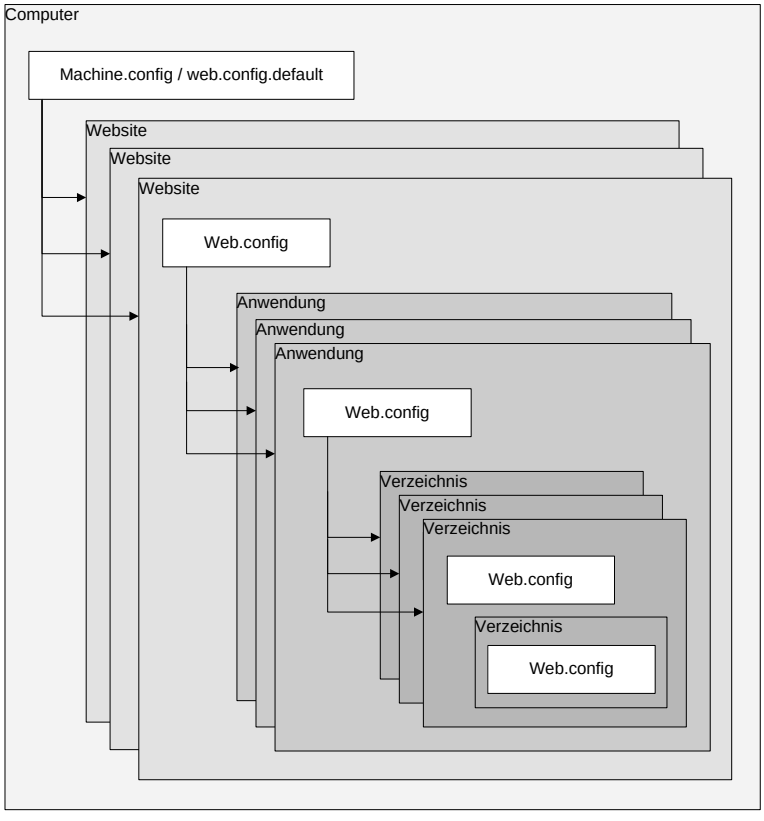
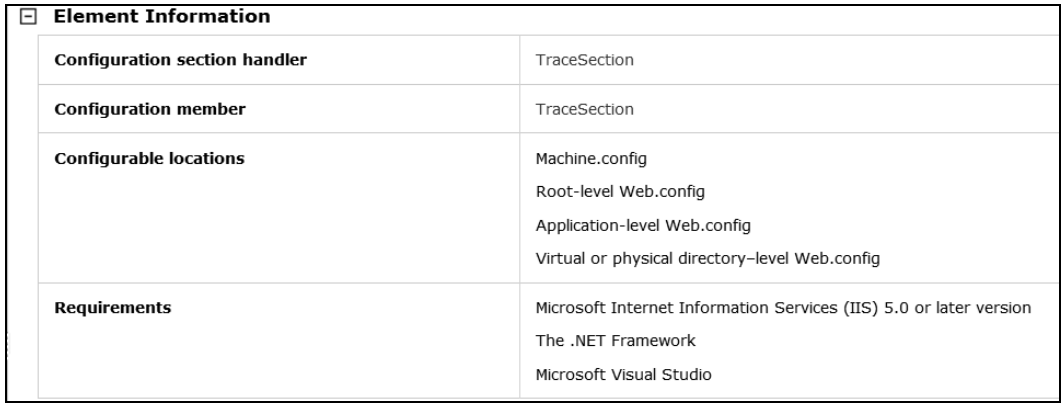



**Abbildung 8.2** Hierarchie der Konfigurationsdateien

**HINWEIS** Bitte beachten Sie folgende Punkte:

Die hierarchische Vererbung von Konfigurationsdateien beruht auf virtuellen Verzeichnissen, also URL-Pfaden und nicht den physischen Dateipfaden. Die Anwendung mit dem URL-Pfad *http://localhost/abc/Anwendung1/* (Dateipfad *C:\InetPub\wwwroot\Anwendung1*) erbt die Konfiguration vom URL-Pfad *http://localhost/abc/* (Dateipfad *C:\InetPub\wwwroot\misc\abc*), obwohl die Verzeichnishierarchie im Dateisystem anders ist.

Es sind nicht alle Konfigurationseinträge auf untergeordneten Ebenen erlaubt. Beispielsweise führt der Versuch, durch ein Element `<authentication mode="Windows" />` in einem untergeordneten Verzeichnis ein anderes Authentifizierungsverfahren festzulegen als in dem übergeordneten Verzeichnis, zu dem Fehler »It is an error to use a section registered as allowDefinition='MachineToAnwendung' beyond Anwendung level. This error can be caused by a virtual directory not being configured as an application in IIS.«.



**Element Information**

| | |
|---|---|
| **Configuration section handler** | TraceSection |
| **Configuration member** | TraceSection |
| **Configurable locations** | Machine.config<br>Root-level Web.config<br>Application-level Web.config<br>Virtual or physical directory–level Web.config |
| **Requirements** | Microsoft Internet Information Services (IIS) 5.0 or later version<br>The .NET Framework<br>Microsoft Visual Studio |

**Abbildung 8.3** Die MSDN-Dokumentation gibt für jedes Konfigurationselement Aufschluss darüber, auf welcher Ebene es erlaubt ist (hier am Beispiel des <*trace*>-Elements)

# Konfigurationselemente

Da Konfigurationseinstellungen sich mittels XML-Dateien definieren lassen, können die Informationen sehr einfach gelesen und verändert werden. Innerhalb dieser Dateien befinden sich vordefinierte XML-Elemente, die für diverse Aspekte von .NET-Anwendungen und deren Umgebung zuständig sind.

Tabelle 8.1 zeigt eine kurze und unvollständige Liste von Hauptelementen, die innerhalb der Konfigurationsdateien verwendet werden können. Für detaillierte Informationen muss hier allerdings auf die .NET Framework-Dokumentation verwiesen werden, da eine genaue Beschreibung jedes Elements den Umfang dieses Buch sprengen würde.

Besonders interessant für ASP.NET-Anwendungen sind die Elemente `<system.web>` und `<appSettings>`. Auf diese wird in den folgenden Abschnitten noch genauer eingegangen.

**HINWEIS** Alle Angaben innerhalb von .NET-Konfigurationsdateien sind *case-sensitiv*, unterscheiden also Groß- und Kleinschreibung.

| Element | Beschreibung |
|---|---|
| `<appSettings>` | Benutzerdefinierte Konfigurationseinträge. Hier können beliebige Werte durch den Entwickler hinterlegt werden, die aus einer laufenden Anwendung heraus ausgelesen werden können. |
| `<configSections>` | Definition von eigenen Elementen und den für die Behandlung dieser Elemente zuständigen Klassen. Dadurch lassen sich Konfigurationsdateien durch benutzerdefinierte Elemente erweitern. |
| `<system.diagnostics>` | Ermöglicht die Festlegung verschiedener Ablaufverfolgungen und die Form, wie Meldungen gesammelt und präsentiert werden können. |
| `<system.net>` | Definiert Einstellungen für Klassen im Namensraum `System.Net`. Dazu gehören unter anderem die Authentifizierungsmodule, die Verbindungsverwaltung, der Proxyserver und die Anforderungsverarbeitung für Internethosts. ▶ |

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.0 mit Visual C# 2010 - Das Entwicklerbuch,
Microsoft Press 2011 (ISBN 978-3-86645-530-6)

| Element | Beschreibung |
|---|---|
| <system.web> | Kern der ASP.NET-Konfiguration. Hier sind Konfigurationen enthalten, die definieren, wie sich Anwendungen verhalten sollen. |
| <system.runtime.remoting> | Beinhaltet Informationen über entfernte Objekte und Kanäle, über welche die Kommunikation zwischen diesen Objekten geführt wird. |

**Tabelle 8.1** Allgemeine Konfigurationselemente

# Konfiguration für einzelne Verzeichnisse und Dateien

Alternativ zur Verwendung mehrerer *web.config*-Dateien in verschiedenen Verzeichnissen ist es möglich, eine *web.config*-Datei in verschiedene Bereiche zu unterteilen. Innerhalb von *web.config*-Dateien können für einzelne Unterverzeichnisse oder einzelne Dateien spezielle Konfigurationen angewendet werden, die sich von der normalen Konfiguration unterscheiden. Dadurch lassen sich Teile der Anwendung individuell konfigurieren.

## Beispiel

Die nachstehende Konfigurationsdatei definiert für verschiedene Unterverzeichnisse und Web Forms den zu verwendenden Kulturkreis. Dadurch greifen diese Anwendungsbereiche (Web Form und Verzeichnis) auf die kulturüblichen Gegebenheiten zurück. Dies wirkt sich beispielsweise auf das Datums- und Währungsformat aus. Dieses Thema wird in Kapitel 29 »Mehrsprachige Webanwendungen (Internationalisierung/Lokalisierung)« detaillierter behandelt.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <!-- Konfiguration für deutsche Benutzer -->
  <location path="VerzeichnisDE">
    <system.web>
      <globalization requestEncoding="utf-8"
          responseEncoding="utf-8" culture="de-DE" uiCulture="de-DE" />
    </system.web>
  </location>
  <location path="VerzeichnisDE/deutsch.aspx">
    <system.web>
      <globalization requestEncoding="utf-8"
          responseEncoding="utf-8" culture="de-DE" uiCulture="de-DE" />
    </system.web>
  </location>
  <!-- Konfiguration für amerikanische Benutzer -->
  <location path="Verzeichnis/Andere/US">
    <system.web>
  …
    </system.web>
  </location>
  <!-- Konfiguration für französische Benutzer -->
  <location path="Verzeichnis/Andere/FR">
    <system.web>
      …
    </system.web>
  </location>
</configuration>
```

**Listing 8.2** Datei- und Verzeichniskonfiguration

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.0 mit Visual C# 2010 - Das Entwicklerbuch,
Microsoft Press 2011 (ISBN 978-3-86645-530-6)

**HINWEIS** Nicht alle Standardelemente lassen sich auf allen Konfigurationsebenen festlegen bzw. überschreiben. Einige Standardelemente erlauben eine Konfiguration lediglich bis zur Anwendungsebene. Auf Verzeichnisebene stehen hingegen nicht alle Konfigurationselemente zur Verfügung. Dadurch ist eine Verwendung des `<location>`-Elements nicht immer möglich. Dies gilt insbesondere für das `<authentication>`-Element. Dieses kann für jede ASP.NET-Anwendung nur einmal, und zwar in der Datei *web.config* des Stammverzeichnisses verwendet werden.

## Überschreiben von Konfigurationseinstellungen verhindern

Die Möglichkeit zum Überschreiben von vererbten Konfigurationseinstellungen ist nicht immer gewünscht und darf aus Sicherheitsgründen nicht immer möglich sein (vgl. Hostingszenarien). Aus diesem Grund hat der Administrator eines Computers bzw. der Webmaster der Webfarm die Möglichkeit, bestimmte Konfigurationen verbindlich und unveränderlich vorzugeben. Das `<location>`-Element ermöglicht es, bestimmte Konfigurationsabschnitte mit dem Attribut `allowOverride="false"` als nicht überschreibbar zu deklarieren:

```
<configuration>
   <location path="Anwendung1" allowOverride="false">
      <system.web>
         <identity impersonate="true" userName=" Anwendung1" password="password1"/>
      </system.web>
   </location>
   <location path="Anwendung2" allowOverride="false">
      <system.web>
         <identity impersonate="true" userName="Anwedung2" password="password2"/>
      </system.web>
   </location>
</configuration>
```

**Listing 8.3** Unterbinden von Konfigurationsänderungen

## <system.web>-Konfigurationselemente

Von besonderer Bedeutung für ASP.NET-Anwendungen sind Elemente, die sich innerhalb der `<system.web>`-Sektion befinden. Die folgende Übersicht beschreibt einige Möglichkeiten und Einsatzbereiche. Aus Platzgründen können auch hier nicht alle Optionen genannt werden. Die .NET Framework-Dokumentation enthält eine sehr umfangreiche Dokumentation aller Einstellungen.

| Elemente | Beschreibung |
|---|---|
| `<authentication>` | Legt die zu verwendende Authentifizierungsmethode fest. Mögliche Ausprägungen sind: `Windows`, `Forms`, `Passport` oder `None` (keine Authentifizierung).<br>`<mode="Windows\|Forms\|Passport\|None">`<br>Unterelemente: `<forms>`, `<passport>` |
| `<authorization>` | Definition von Zugriffsrechten. Allen Bestandteilen der Anwendung können detaillierte Zugriffsrechte zugewiesen werden.<br>Unterelemente: `<allow>`, `<deny>` ▶ |

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.0 mit Visual C# 2010 - Das Entwicklerbuch,
Microsoft Press 2011 (ISBN 978-3-86645-530-6)

| Elemente | Beschreibung |
|---|---|
| <browserCaps> | Definition von unterstützten Browserfähigkeiten. Festlegung, ob und wie die Fähigkeiten des Browsers erkannt werden sollen.<br>Unterelemente: <user>, <filter>, <result><br>Hinweis: ASP.NET kennt ab Version 2.0 zusätzlich eine neue, mächtigere Funktion zur Definition von Browserfähigkeiten über *.browser*-Dateien, siehe PDF-Zusatzkapitel »Browserfähigkeiten« (Datei »G45 - Browserfähigkeiten.pdf«) auf der Buch-DVD und im Leser-Webportal zum Buch. |
| <clientTarget> | Browsertypen, auf welche die Anwendung ausgerichtet sind. Mögliche Ausprägungen sind so genannte Uplevel- und Downlevel-Browser oder eine individuelle Unterstützung für jede Anfrage.<br>Um ausschließlich Uplevel-Browser zu unterstützen, kann dieses Attribut auf UpLevel gesetzt werden. Downlevel-Browser werden anschließend nicht mehr berücksichtigt. |
| <compilation> | Element zur Steuerung der automatischen Kompilierung von ASP.NET-Anwendungen<br>debug="true\|false"<br>batch="true\|false"<br>batchTimeout="number of seconds"<br>defaultLanguage="language"<br>explicit="true\|false" |
| <customErrors> | Festlegung der Rückgabeseiten für HTTP-Fehler. Dadurch lässt sich für bestimmte Fehlersituationen ein angepasstes Verhalten implementieren.<br>defaultRedirect="url"<br>mode="On\|Off\|RemoteOnly"<br>Unterelemente: <error> |
| <globalization> | Einstellung der Zeichenkodierung für Anfragen/Antworten sowie Ländereinstellungen |
| <httpHandlers> | Definition der Verarbeitungsmodule für eingehende Anforderungen für unterschiedliche URLs und HTTP-Verben |
| <httpModules> | Konfiguration der verwendeten HTTP-Module innerhalb einer Anwendung |
| <httpRuntime> | Konfiguration des Verhaltens des ASP.NET-Arbeitsprozesses<br>useFullyQualifiedRedirectUrl="true\|false"<br>maxRequestLength="size in kbytes"<br>executionTimeout="seconds" |
| <identity> | Benutzerkonto, unter dem die ASP.NET-Seiten auf dem Server agieren (Identität)<br>impersonate="true\|false"<br>userName="username"<br>password="password" |
| <machineKey> | Konfiguriert die zu verwendenden Schlüssel für die Ver- und Entschlüsselung von Cookieinformationen bei Formularauthentifizierung. |
| <pages> | Vorgabe der seitenspezifischen Konfigurationseinstellungen. Entspricht den Attributen der @Page-Direktive, die auch in jedem einzelnen Web Form verwendet werden kann.<br>buffer="true\|false"<br>enableSessionState="true\|false\|ReadOnly"<br>enableViewState="true\|false"<br>enableViewStateMac="true\|false"<br>autoEventWireup="true\|false" |
| <processModel> | Konfiguriert die Einstellungen des ASP.NET-Prozessmodells auf einem IIS 5.x-Server. Diese Einstellung gilt nicht mehr ab IIS 6.0 (außer im so genannten »Isolationsmodus«, den es aus Kompatibilitätsgründen gibt). ▶ |

| Elemente | Beschreibung |
|---|---|
| <securityPolicy> | Verweist für verschiedene Sicherheitsebenen auf Dateien mit gewünschten Sicherheitsrichtlinien. |
| <sessionState> | Konfiguration der zu verwendenden Sitzungsstatuseinstellungen für die aktuelle Anwendung<br>mode="Off\|Inproc\|StateServer\|SQLServer"<br>cookieless="true\|false"<br>timeout="Anzahl Minuten" |
| <trace> | Konfiguration der Ablaufverfolgung<br>enabled="true\|false"<br>localOnly="true\|false"<br>pageOutput="true\|false" |
| <webServices> | Legt die Einstellungen für erstellte XML-Webservices fest. |

**Tabelle 8.2** <*system.web*>-Konfigurationselemente

# Verschlüsselte Sektionen (Protected Configuration)

.NET erlaubt es seit Version 2.0, einzelne Sektionen der *.config*-Dateien zu verschlüsseln (*Protected Configuration*). Für die Verschlüsselung sind zwei verschiedene Provider verfügbar: RSA und *Windows Data Protection API* (DAPI). Da DAPI-Schlüssel computerspezifisch sind, ist die Verwendung von RSA vorteilhaft, wenn die Anwendung auf einen anderen Computer migriert werden soll. Ein Werkzeug liefert Microsoft derzeit nur für *web.config*-Dateien. Die Verschlüsselung für die übrigen Anwendungskonfigurationsdateien muss in eigenem Programmcode erfolgen.

**ACHTUNG** Nicht alle Sektionen können verschlüsselt werden. Ausnahmen sind beispielsweise <mscorlib>, <system.runtime.remoting> und <protectedData>.

## Verschlüsselung von web.config-Dateien mit aspnet_regiis.exe

Das in .NET Framework Redistributable enthaltene Werkzeug *aspnet_regiis.exe* erlaubt es, die notwendigen Schritte zur Verschlüsselung von Konfigurationselementen auszuführen.

Erforderlich ist zuerst, dass das Benutzerkonto, unter dem die Webanwendung läuft, Zugriff auf den zu verwendenden Schlüssel hat. Wenn kein expliziter Schlüsselcontainer angegeben wird, wird der eingebaute NetFrameworkConfigurationKey verwendet. Das Standardbenutzerkonto für Webanwendungen erhält mit folgendem Befehl Zugriff auf den Schlüssel:

```
aspnet_regiis -pa "NetFrameworkConfigurationKey" "EO1\ASPNET"
```

**TIPP** Falls Ihnen nicht bekannt ist, unter welcher Identität Ihre Webanwendung läuft, können Sie diese mit

```
System.Security.Principal.WindowsIdentity.GetCurrent().Name
```

ermitteln. Weitere Informationen über die Anwendungsidentität in Webanwendungen erhalten Sie in Kapitel 30 »Sicherheit«.

Mit dem folgenden Befehl wird dann die Sektion <connectionStrings> in der *web.config*-Datei im Stammverzeichnis der Webanwendung *WWWings_Web*, die als erste Website in IIS eingetragen ist, mit dem RSA-Provider asymmetrisch verschlüsselt:

```
aspnet_regiis -pe "ConnectionStrings" -prov "RSAProtectedConfigurationProvider" -app
"W3SVC/1/WWWings_Web"
```

Das Ergebnis ist in den nachstehenden Listings dokumentiert:

```
<protectedData>
  <protectedDataSections>
    <add name="ConnectionStrings" provider="RSAProtectedConfigurationProvider"
      inheritedByChildren="false" />
  </protectedDataSections>
</protectedData>
```

**Listing 8.4** Festlegung, welche Elemente verschlüsselt werden sollen

```
<ConnectionStrings>
<EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
   xmlns="http://www.w3.org/2001/04/xmlenc#">
   <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
   <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
    <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
     <EncryptionMthod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
      <KeyName>Rsa Key</KeyName>
     </KeyInfo>
     <CipherData>
<CipherValue>UVwRWnqcICxUQRJOcxFUOdc4+mf/v/yOLaXLqOhEa/ecpvEemloBa4J6oO4CgOBKC5JOerxmDqX55BO8aVRLvb6iudHyt/
hMLXSDwx7Q1SisttS9AD/L55jfbY6cUo83VL9yGOivJgv9MJoCANI/hqIMd9XUe16NjRxnXOVbiJ8=</CipherValue>
     </CipherData>
    </EncryptedKey>
   </KeyInfo>
   <CipherData>
<CipherValue>YEkARsOqzYdCOU+JrVuIBORczV1z9nt+JcIA9/pJh4n9s8XCuGFBQC+AQa+T6T/hc75dNEWuHPbfZ99SlguXXoZHrzn
smRFKMUR1OtWxdfHfuiE9dcESuVrigqovtspsTUhnQvoB1JSJu174zmbeeDzh11mkIM7EBI26WOwl6fqXPH/…
/JOJaUM7hLUc65lHGJtMU2vfo15muzpaP+wCbGYeMr/mK52wZT2An3NjvojXGadqmahm6vfIsVWTBaxhMC2uMVZO4Sjbafa4QqItVHPW
rSQZ9YtOcDGXMPJPd77cBUs/vw2ickGwL9bQ6pHqHYaqLs8eDl9bN</CipherValue>
   </CipherData>
  </EncryptedData>
</ConnectionStrings>
```

**Listing 8.5** Verschlüsselte Verbindungszeichenfolgen

Sie können die Verschlüsselung wieder aufheben mit dem Befehl:

```
aspnet_regiis -pd "ConnectionStrings" -app "/"
```

### Verschlüsselung per Programmcode

Für Anwendungskonfigurationsdateien von Windows- und Konsolenanwendungen bleibt derzeit nur die Möglichkeit, die Verschlüsselung per Programmcode selbst zu schreiben. Dies ist zum Glück kein großer Aufwand: Die Klasse `ConfigurationSection` bietet dafür zwei einfache Methoden, `ProtectSection()` und `UnprotectSection()`. Kurioserweise werden diese Methoden in der MSDN-Dokumentation als »not intended to be used directly from your code« bezeichnet.

Das folgende Beispiel kehrt den Verschlüsselungsstatus der Sektion <connectionStrings> in der Anwendungskonfigurationsdatei um. Verwendet wird hier der DAPI-Provider:

```
public void Config_Verschluesseln()
{
   Demo.Print("Zugriff auf Sektion...");
   Configuration c =
     ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
   ConfigurationSection s = c.GetSection("ConnectionStrings");
   Demo.Print("   Name: " + s.SectionInformation.Name);
   Demo.Print("   Verschlüsselt?: " + s.SectionInformation.IsProtected);
   if (!s.SectionInformation.IsProtected)
   {
    Demo.Print("Verschlüsseln...");
    s.SectionInformation.ProtectSection
      (ProtectedConfiguration.DataProtectionProviderName);
   }
   else
   {
    Demo.Print("Entschlüsseln...");
    s.SectionInformation.UnprotectSection();

   }
   c.Save();
   Demo.Print("Kontrolle...");
   Configuration c2 =
     ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
   ConfigurationSection s2 = c2.GetSection("ConnectionStrings");
   Demo.Print("   Name: " + s2.SectionInformation.Name);
   Demo.Print("   Verschlüsselt?: " + s2.SectionInformation.IsProtected);
   Demo.Print("Nutzen...");
   string WWWAccessDatenbank =
     ConfigurationManager.ConnectionStrings["WWWDatenbank_MSSQL"].ConnectionString;
   Demo.Print("ConnectionString: " + WWWAccessDatenbank);
   Demo.Print("Fertig! :-)");
}
```

**Listing 8.6** Verschlüsseln und Entschlüsseln der Verbindungszeichenfolgen [VerschiedeneDemos_CS/FCL/Configuration.cs]

# Anwendungsspezifische Einstellungen

Nicht für alle Szenarien bieten Konfigurationsdateien ausreichend Standardelemente an, um alle möglichen Konfigurationsaspekte abzudecken. Die *web.config*-Datei ermöglicht auch die Ablage beliebiger Zeichenketten, in denen anwendungsspezifische Daten abgelegt werden können. Ein Beispiel dafür sind Namen von Mailservern und Verbindungszeichenfolgen zu Datenquellen.

**TIPP** Damit eine Anwendung portabel und maschinenneutral installierbar ist, muss während der Entwicklung darauf geachtet werden, dass möglichst keine absoluten Pfade zu URL-Ressourcen und sonstigen Dateien (Datenbankdateien, Textdateien, XML-Dateien) sowie keine statischen Servernamen verwendet werden. Es sollte jederzeit möglich sein, solche Einstellungen von »außen« zentral konfigurieren und verändern zu können, ohne dass innerhalb des Codes »hart« kodiert werden muss.

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.0 mit Visual C# 2010 - Das Entwicklerbuch,
Microsoft Press 2011 (ISBN 978-3-86645-530-6)

# <appSettings>-Element

Es ist es auf einfache Weise möglich, beliebige Zeichenketten in der *web.config*-Datei abzulegen und aus der Webanwendung heraus abzufragen. Damit bietet sich die *web.config*-Datei auch als Speicher für anwendungsspezifische Konfigurationsdaten an, z. B. Datenbankverbindungszeichenfolgen und Pfade zu Dateien.

Diese benutzerdefinierten Daten können in Form von Schlüssel-Wert-Paaren in dem `<appSettings>`-Element abgelegt werden:

```
<appSettings>
  <add key="Schluessel" value="Wert />
</appSettings>
```

**TIPP** `<appSettings>`-Sektionen dürfen auch in untergeordneten *web.config*-Dateien (*web.config* in Unterordnern) vorkommen und dort Werte aus übergeordneten Dateien überschreiben.

## Beispiel

Das folgende Beispiel verwendet das `<appSettings>`-Element, um eine Datenbankverbindungszeichenfolge und einige Benutzerinformationen zu hinterlegen:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <!--
        Diese Konfigurationseinträge überschreiben entsprechende Einträge
        in übergeordnete Verzeichnisse, falls vorhanden.
        Beispielkonfigurationseinträge
    -->
    <add key="Erstellungsjahr" value="2004-2006" />
    <add key="Autor" value="Dr. Holger Schwichtenberg" />
    <add key="Hintergrundfarbe" value="#669966" />
    <add key="Version" value="1.0.2" />
    <add key="UploadVerzeichnis" value="Dateien" />
  </appSettings>
</configuration>
```

**Listing 8.7** Beispiele für benutzerdefinierte Anwendungskonfigurationseinstellungen

**TIPP** Anwendungseinstellungen, die von übergeordneten Konfigurationsdateien geerbt wurden, können mit `<remove>` deaktiviert werden. Mit `<clear>` können alle übergeordneten Einstellungen deaktiviert werden.

Leider kann man Anwendungseinstellungen nicht in der Konfigurationsdatei, sondern nur im Code aus Teilen zusammensetzen.

## Programmgesteuerter Zugriff

Um innerhalb einer Anwendung auf diese Werte des `<appSettings>`-Elements zugreifen zu können, existiert die Objektmenge `AppSettings` innerhalb der Klasse `System.Configuration.ConfigurationManager`. Mit dieser Klasse lassen sich alle Werte direkt als `String` auslesen, z. B.

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.0 mit Visual C# 2010 - Das Entwicklerbuch,
Microsoft Press 2011 (ISBN 978-3-86645-530-6)

```
Me.TextBox1.Text = System.Configuration.ConfigurationManager.AppSettings["Autor"]
```

Falls eine Einstellung *Autor* nicht vorhanden ist, gibt es keine Fehlermeldung, sondern als Ergebnis eine leere Zeichenkette.

**HINWEIS** In ASP.NET 1.x gab es diese Funktion auch bereits in der Klasse `System.Configuration.ConfigurationSettings`. Diese Klasse ist weiterhin vorhanden. `System.Configuration.ConfigurationManager` bietet aber mehr Funktionen, z. B. auch das Beschreiben der Konfigurationsdateien.

Bitte beachten Sie, dass die Klasse `System.Configuration.ConfigurationManager` in der Assembly *System.Configuration.dll* liegt, die in einigen Projektarten standardmäßig nicht referenziert wird. Viele Entwickler suchen lange nach der Klasse, weil die Referenz fehlt.

## Deklarativer Zugriff

Mit ASP.NET 2.0 wurde die Möglichkeit eingeführt, aus dem ASPX-Code heraus deklarativ auf die benutzerspezifischen Einstellungen zuzugreifen. Die Funktion heißt *ASP.NET Expressions*. Die Expressions werden mit <%$ eingeleitet. Sie sind in jedem beliebigen Attribut eines Serversteuerelements erlaubt, dürfen aber nicht wie <%= %> als eigenständige Platzhalter außerhalb eines Servertags verwendet werden. VWD bietet im Eigenschaftenfenster unter dem Eintrag *Expressions* einen Assistenten für ASP.NET Expressions an. Der Designer zeigt zur Entwicklungszeit bereits die Werte an.

```
<asp:Label ID="C_Copyright" runat="server" Text="<%$ AppSettings:Autor %> ">
</asp:Label>
<asp:Label ID="Label1" runat="server" Text="<%$ AppSettings:Erstellungsjahr %>">
</asp:Label>
```

**Listing 8.8** Verwendung von ASP.NET Expressions [/master/WWWings.master]

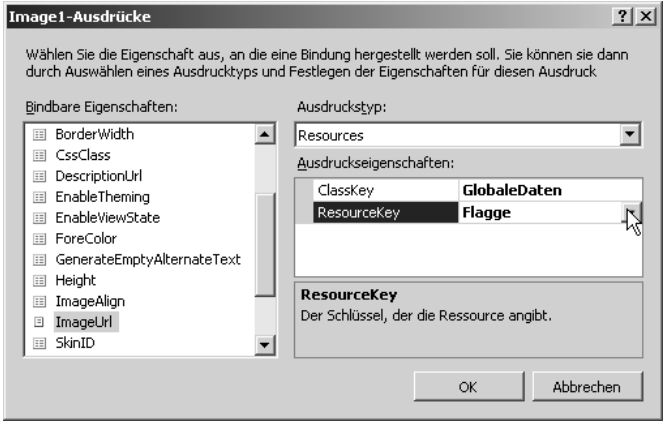



**Abbildung 8.4** Definieren von ASP.NET Expressions mit VWD

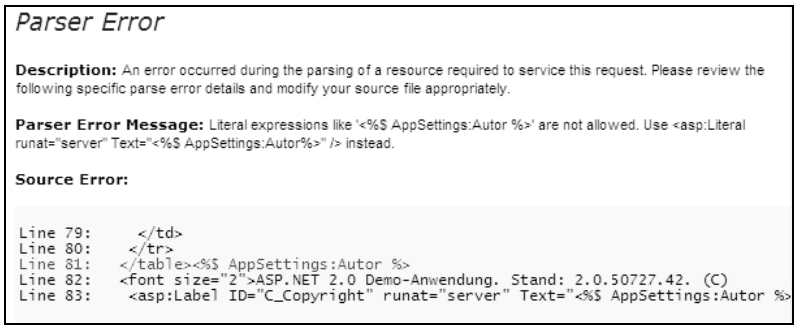



**Abbildung 8.5** Das ist nicht erlaubt: ASP.NET Expressions dürfen nicht außerhalb von Servertags stehen (Zeile 81). Richtig gemacht ist es in dem *Label*-Steuerelement in Zeile 83.

# Eigene Konfigurationssektionen

Das `<appSettings>`-Element ist hinsichtlich der Struktur der Daten (Attribut-Wert-Paare) sehr eingeschränkt. Die *web.config*-Datei ist besser erweiterbar durch eigene Konfigurationssektionen. Für jede Sektion muss eine Klasse realisiert werden, welche die Schnittstelle `System.Configuration.IConfigurationSectionHandler` implementiert.

Eine weitere Alternative besteht darin, eine komplette eigene (XML-)Konfigurationsdatei zu verwenden.

**HINWEIS** Weitere Erläuterungen und Beispiele zu diesen beiden Themen sind aus Platzgründen leider nicht möglich.

# Datenbankverbindungszeichenfolgen

Während in .NET 1.x Verbindungszeichenfolgen nur in den benutzerdefinierten Anwendungseinstellungen hinterlegt werden konnten, bietet .NET seit Version 2.0 dafür eine eigene Sektion `<connectionStrings>`. Die Sektion funktioniert genau wie die `<appSettings>`-Sektion mit `<add>`, `<remove>` und `<clear>`. Für jede Datenverbindung kann man zwei Eigenschaften festlegen:

- Verbindungszeichenfolge
- ADO.NET-Datenprovider

**TIPP** Die Ablage des ADO.NET-Datenproviders in der Konfigurationsdatei macht Sinn im Zusammenhang mit der Funktion *Providerfabriken*, die mit ADO.NET 2.0 eingeführt wurde, siehe PDF-Zusatzkapitel »Datenzugriff mit ADO.NET« (Datei »5531K11-ADO.NET.pdf«) auf der Buch-DVD und im Leser-Webportal zum Buch).

## Beispiel

In dem folgenden Listing werden mit `<clear>` zunächst alle übergeordneten Verbindungszeichenfolgen für diese und die untergeordneten Konfigurationsebenen ausgeblendet. Danach werden zwei neue Verbindungszeichenfolgen hinzugefügt:

```
<ConnectionStrings>
<clear/>
<add name="LocalSqlServer"
  ConnectionString="data source=.\SQLEXPRESS;
  Integrated Security=SSPI;AttachDBFilename=|DataDirectory|aspnetdb.mdf;
  User Instance=true"
  providerName="System.Data.SqlClient" />
<add name="Protokolldatenbank"
  ConnectionString="data source=.\SQLEXPRESS;
  Integrated Security=SSPI;AttachDBFilename=|DataDirectory|Protokolldatenbank.mdf;
  User Instance=true"
  providerName="System.Data.SqlClient" />
</ConnectionStrings>
```

**Listing 8.9** Verbindungszeichenfolgen und Providerfestlegung in der Konfigurationsdatei

## Programmgesteuerter Zugriff

Der programmgesteuerte Zugriff auf die in den Konfigurationsdateien gespeicherten Datenverbindungen erfolgt über die Klasse `System.Configuration.ConfigurationManager`.

```
public static DbConnection GetConfiguredConnection()
  {
   // --- Fabrik erzeugen
   DbProviderFactory provider =
    DbProviderFactories.GetFactory(System.Configuration.
    ConfigurationManager.ConnectionStrings["WWWings"].ProviderName);
   // --- Verbindung aufbauen
   DbConnection conn = provider.CreateConnection();
   conn.ConnectionString = System.Configuration.ConfigurationManager.
    ConnectionStrings["WWWings"].ConnectionString;
   conn.Open();
   return conn;
  }
```

**Listing 8.10** Auslesen der gespeicherten Einstellungen für Datenprovider und Verbindungszeichenfolgen beim Öffnen einer Datenbankverbindung mithilfe von ADO.NET-Providerfabriken

## Deklarativer Zugriff

Datenbankverbindungen können auch über ASP.NET Expressions (vgl. vorheriger Abschnitt zu `<appSettings>`) verwendet werden. Dies kommt insbesondere in dem `SqlDataSource`-Steuerelement zum Einsatz:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:CS_WorldWideWingsSQLServer %>"
ProviderName="<%$ ConnectionStrings:CS_WorldWideWingsSQLServer.ProviderName %>" …>
```

**Listing 8.11** Verwendung von ASP.NET Expressions [/Fluege/RAD/Fluege_SDS.master]

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.0 mit Visual C# 2010 - Das Entwicklerbuch,
Microsoft Press 2011 (ISBN 978-3-86645-530-6)

# Einheitliche Konfiguration in IIS 7.x

In IIS 7.x (Internet Information Services) legt Microsoft Wert auf eine einfachere Konfiguration. In den früheren Versionen ergaben sich die Einstellungen für eine Webanwendung aus dem Zusammenspiel der Einstellungen in der IIS-Metabase, die über den IIS-Manager festgelegt wurden, und den Einstellungen in den XML-basierten ASP.NET-Konfigurationsdateien, den *web.config*-Dateien.

IIS 7.x übernimmt das .NET-basierte Konfigurationssystem, das heißt, alle Einstellungen einer Webanwendung, sowohl die von ASP.NET als auch die von IIS, werden in .config-XML-Dateien gespeichert. Microsoft spricht vom *Configuration Store*, der die bisherige Metabase ersetzt.

## Nutzung der web.config-Dateien

Einstellungen von IIS zu einer bestimmten Webanwendung werden in der obersten *web.config*-Datei einer Webanwendung in der Sektion `<system.webServer>` gespeichert. Abbildung 8.6 zeigt ein Beispiel, in dem die IIS-Einstellungen *directoryBrowse* und *defaultDocument* zusammen mit den ASP.NET-Einstellungen *profile* und *authentication* in einer Konfigurationsdatei stehen.

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>
    <system.web>
        <profile>
            <properties>
                <group name="Adresse">
                    <add allowAnonymous="false" defaultValue="" name="Strasse" readOnly="false" serializeAs="String" type="System.String" />
                    <add allowAnonymous="false" defaultValue="" name="Ort" readOnly="false" serializeAs="String" type="System.String" />
                    <add allowAnonymous="false" defaultValue="" name="PLZ" readOnly="false" serializeAs="String" type="System.Boolean" />
                </group>
            </properties>
        </profile>
        <authentication mode="Forms">
            <forms loginUrl="anmeldung.aspx" />
        </authentication>
    </system.web>
    <system.webServer>
        <defaultDocument>
            <files>
                <clear />
                <add value="default.aspx" />
                <add value="start.aspx" />
            </files>
        </defaultDocument>
        <directoryBrowse enabled="true" showFlags="LongDate, Extension, Size, Time, Date" />
    </system.webServer>
</configuration>
```

**Abbildung 8.6** Eine gemeinsame Konfigurationsdatei für IIS 7.x und ASP.NET

Die Nutzung der *web.config*-Dateien bietet gegenüber dem bisherigen metabasebasierten Konfigurationsmodell vier wesentliche Vorteile:

- Die Konfigurationsdateien lassen sich mit einfachen Text- oder XML-Editoren bearbeiten
- Die Konfigurationsdateien sind einfacher, nämlich per Dateikopie (XCopy-Deployment) und auch per FTP übertragbar. Geänderte Konfigurationsdateien führen außerdem sofort zur Verhaltensänderung des Servers.
- Die Konfigurationsdateien liegen im Ordner des jeweiligen Webprojekts. Das macht die Delegation von administrativen Aufgaben einfacher, da der für diese Datei verantwortliche Mitarbeiter weder Frontpage Server Extensions noch einen RPC-Zugang zu dem Webserverdienst benötigt.
- Die Konfigurationsdateien bilden eine Hierarchie. In jedem Unterverzeichnis können Konfigurationsdateien existieren, wobei untergeordnete Konfigurationsdateien übergeordnete Einstellungen überschreiben.

Microsoft spricht bei IIS 7.x von einem *Unified Configuration Model*.

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.0 mit Visual C# 2010 - Das Entwicklerbuch,
Microsoft Press 2011 (ISBN 978-3-86645-530-6)

# ApplicationHost.config-Dateien

Zentrale Einstellungen, die für den ganzen Webserver gelten, befinden sich in der Datei *ApplicationHost.config* im Verzeichnis *%Systemroot%\System32\inetsrv\config*. Die *ApplicationHost.config*-Datei enthält die Pfade der virtuellen Webserver (Abbildung 8.7) sowie die Anwendungspooleinstellungen. Angaben für die Pfade der Websites lassen sich nicht auf untergeordneter Ebene überschreiben, da dies keinen Sinn ergeben würde.

```
<location path="www.IT-Visions.de">
    <system.webServer>
        <security>
            <authentication>
                <basicAuthentication enabled="true" />
            </authentication>
        </security>
        <modules>
            <clear />
            <add name="HttpCacheModule" type="" preCondition="" />
            <add name="StaticCompressionModule" type="" preCondition="" />
            <add name="DefaultDocumentModule" type="" preCondition="" />
            <add name="DirectoryListingModule" type="" preCondition="" />
            <add name="ProtocolSupportModule" type="" preCondition="" />
            <add name="HttpRedirectionModule" type="" preCondition="" />
            <add name="StaticFileModule" type="" preCondition="" />
            <add name="AnonymousAuthenticationModule" type="" preCondition="" />
            <add name="RequestFilteringModule" type="" preCondition="" />
            <add name="Profile" type="System.Web.Profile.ProfileModule" preCondition="" />
            <add name="CustomErrorModule" type="" preCondition="" />
            <add name="HttpLoggingModule" type="" preCondition="" />
            <add name="FailedRequestsTracingModule" type="" preCondition="" />
            <add name="RequestMonitorModule" type="" preCondition="" />
        </modules>
    </system.webServer>
</location>
<location path="www.IT-Visions.de/Lexikon">
    <system.webServer>
        <httpLogging selectiveLogging="LogError" />
        <urlCompression doDynamicCompression="true" />
    </system.webServer>
</location>
```

**Abbildung 8.7** Die Datei *ApplicationHost.config*

Die Datei *ApplicationHost.config* erbt wieder von der .NET Framework-Datei *Machine.config* . In der Vererbungshierarchie unter der Datei *ApplicationHost.config* steht die globale *web.config*-Datei, die sich in *%Systemroot%\Microsoft.NET\Framework\<Versionsnummer>\Config* befindet. Die *web.config*-Dateien der einzelnen virtuellen Webserver – alias Websites – erben wiederum von dieser globalen *web.config*-Datei.

# Konfigurationselemente

Das folgende Listing zeigt die Hauptelemente der IIS 7.x-Konfigurationsdateien. Die Überschreibbarkeit von *web.config*-Einstellungen kann durch den Zusatz `allowOverride="false"` in einem Element bei Bedarf auch verhindert werden; ein Webprovider kann auf diese Weise festlegen, welche Einstellungen er einem Kunden zur Verfügung stellen will und welche nicht:

```
<system.ApplicationHost>   <!-- section-group -->
   <ApplicationPools/>
   <listenerAdapters/>
   <log/>
   <sites/>
   <webLimits>
</system.ApplicationHost>
<system.webServer>   <!-- section-group -->
```

```
   <asp/>
   <cgi/>
   <defaultDocument/>
   <directoryBrowse/>
   <globalModules/>
   <serverModules/>
   <serverHandlers/>
   <httpCompression/>
   <httpProtocol/>
   <httpRedirect/>
   <httpErrors/>
   <tracing/>
   <isapiFilters/>
   <serverSideInclude/>
   <staticContent/>
   <httpLogging/>
   <security>   <!-- section-group -->
      <authentication>   <!-- nested section-group -->
         <anonymousAuthentication/>
         <basicAutentication/>
         <clientCertificateMappingAuthentication/>
         <iisClientCertificateMappingAuthentication/>
         <digestAuthentication/>
         <windowsAuthentication/>
      </authentication>
      <extensionRestrictions/>
      <access/>
      <ipSecurity/>
      <urlAuthorization/>
      <impersonation/>
      <hiddenNamespaces/>
      <urlScan/>
   </security>
</system.webServer>
```

**Listing 8.12** Hauptelemente in den IIS 7.x-Konfigurationsdateien

# Administration

Für ASP.NET 1.x existierte kein Administrationswerkzeug von Microsoft; alle Einstellungen mussten ohne jegliche Eingabeunterstützung in den XML-Dateien (*web.config* und *machine.config*) direkt eingegeben werden. Ein Editor wurde nur von einem Drittanbieter angeboten (HunterStone *web.config* Editor).

ASP.NET bietet seit Version 2.0 folgende Administrationsoptionen im Standardlieferumfang:

- Eine direkte Bearbeitung der XML-Dateien
- Eine Erweiterung der MMC-Konsole *Internetinformationsdienste-Manager* (IIS-Manager)
- Webbasiertes Verwaltungswerkzeug *(/asp.netwebadminfiles*)
- Konfigurations-API (Klassen im Namensraum `System.Web.Configuration`)

**TIPP** In IIS 7.0/7.5 kann man die meisten Einstellungen von ASP.NET komfortabel über den neuen IIS-Manager verwalten.

**HINWEIS** Das Konfigurations-API von ASP.NET wird in diesem Buch nicht besprochen.

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.0 mit Visual C# 2010 - Das Entwicklerbuch,
Microsoft Press 2011 (ISBN 978-3-86645-530-6)

# MMC-basierte Verwaltung in IIS 5.x / 6.0

ASP.NET erweitert bei der Installation das auf einem System vorhandene MMC-Snap-In für IIS. Für Webserver und jedes einzelne Unterverzeichnis erscheint in den Eigenschaften eine neue Registerkarte *ASP.NET*. Für alle Verzeichnisse, die eigenständige IIS-Anwendungen darstellen, bietet die Registerkarte die Auswahl der zu verwendenden ASP.NET-Laufzeitumgebung. Aufgelistet werden in dem Auswahlfeld alle installierten ASP.NET-Versionen. Die Schaltfläche *Konfiguration bearbeiten* (*Edit Configuration*) führt zu einem Fenster mit mehreren Registerkarten, auf denen die Einstellungen der *web.config*-Datei für das jeweilige Verzeichnis bearbeitet werden können. *Globale Konfiguration bearbeiten* (*Edit Global Configuration)* bearbeitet die *machine.config*-Datei.

**WICHTIG** Sie können die Konfiguration nur für Webanwendungen mit ASP.NET-Versionen 2.0 oder höher bearbeiten. Wenn die ASP.NET-Version nicht auf 2.0.50727 steht, sind die Schaltflächen inaktiv. Es gibt in diesem Dialog keine Auswahl für ASP.NET 3.0 oder 3.5. »2.0« umfasst hier auch 3.0/3.5!


**Abbildung 8.8** Zusätzliche Registerkarte in der IIS-Konsole

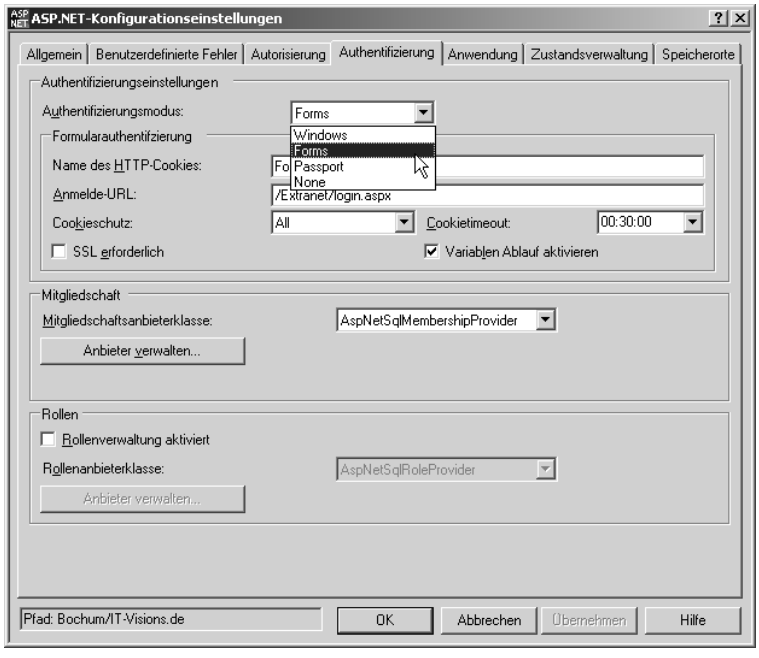



**Abbildung 8.9** Verwaltung der Authentifizierungseinstellungen

Mit MMC können insbesondere Verbindungszeichenfolgen und Anwendungseinstellungen komfortabel bearbeitet werden. Bitte beachten Sie, dass hier alle Einstellungen, auch die geerbten, angezeigt werden. Einstellungen, die durchgestrichen sind, wurden vererbt, aber deaktiviert (mit `<remove>` oder `<clear>`).

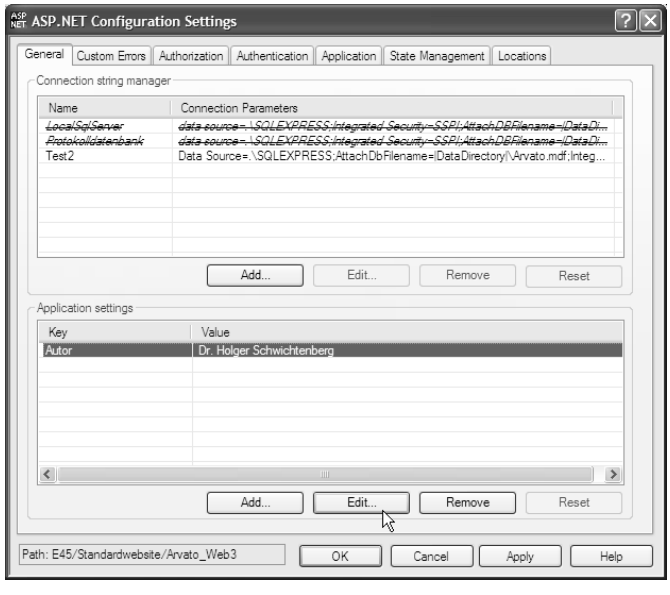



**Abbildung 8.10** Verwaltung der Verbindungszeichenfolgen und der benutzerdefinierten Einstellungen

# MMC-basierte Verwaltung in IIS 7.x

In IIS 7.x ist die Verwaltung von ASP.NET komplett in den IIS-Manager integriert.


**Abbildung 8.11** ASP.NET in der IIS 7.x-Verwaltungskonsole

# Webbasierte Verwaltung

Die MMC-basierte Verwaltung funktioniert nur für Webanwendungen, die in IIS laufen und zu denen ein RPC-Zugang zu IIS besteht. Für die Konfiguration ohne RPC-Zugang zu IIS (übliches Webhostingszenario) oder zur Konfiguration von Webanwendungen, die mit ASP.NET Development Server entwickelt werden, stellt Microsoft alternativ ein webbasiertes Administrationswerkzeug (Websiteverwaltungswerkzeug – im Englischen: *Website Administration Tool* – kurz *WAST*) zur Verfügung, das in ASP.NET entwickelt wurde und im .NET Framework-Verzeichnis

```
\WINDOWS\Microsoft.NET\Framework\v4.0.30319\ASP.NETWebAdminFiles
```

liegt. Die Anwendung kann durch den Aufruf

*http://localhost:12345/asp.netwebadminfiles/default.aspx?applicationPhysicalPath=c:\website*

gestartet werden. Innerhalb von VWD kann die Administrationsanwendung einfach über den Menüeintrag *Website/ASP.NET-Konfiguration* aufgerufen werden.

**HINWEIS** Das MMC-basierte und das webbasierte Verwaltungswerkzeug sind nicht äquivalent. Das webbasierte Verwaltungswerkzeug bietet eine eher assistentengesteuerte Oberfläche mit »freundlichen« Begriffen, während sich das MMC-Werkzeug hinsichtlich der Einstellung sehr stark an der *web.config*-Datei orientiert. Die Weboberfläche bietet als zusätzliche Funktion die Benutzerverwaltung für das Membership-System.

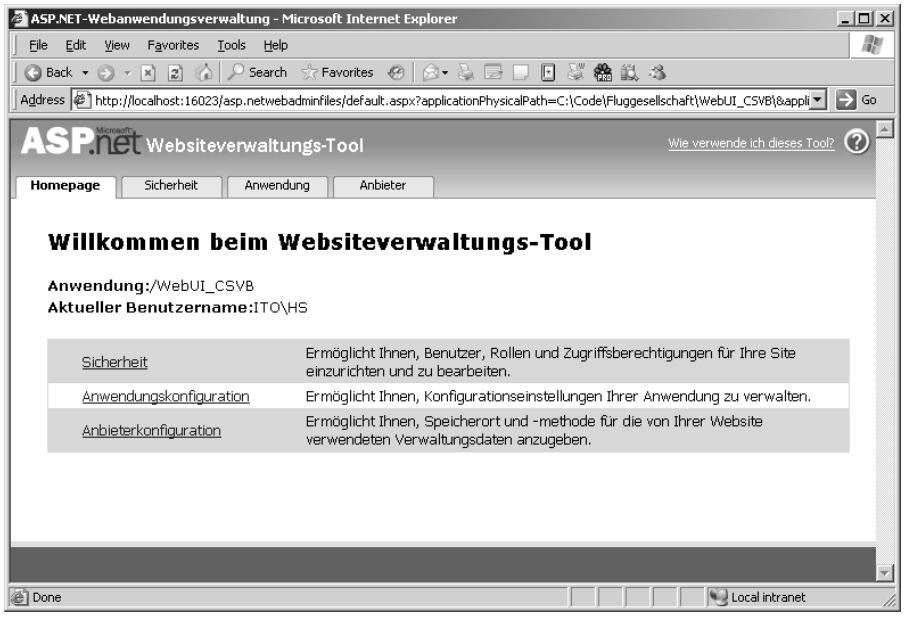



**Abbildung 8.12** Startseite des Administrationswerkzeugs

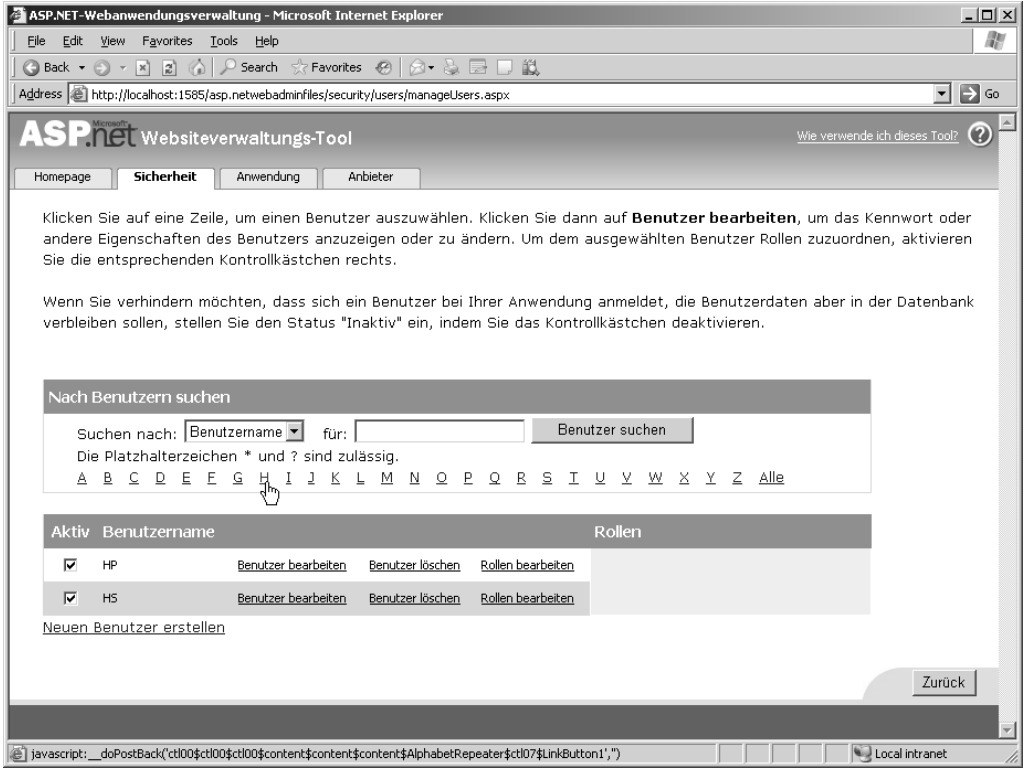



**Abbildung 8.13** Benutzerverwaltung mit dem webbasierten ASP.NET-Administrationswerkzeug

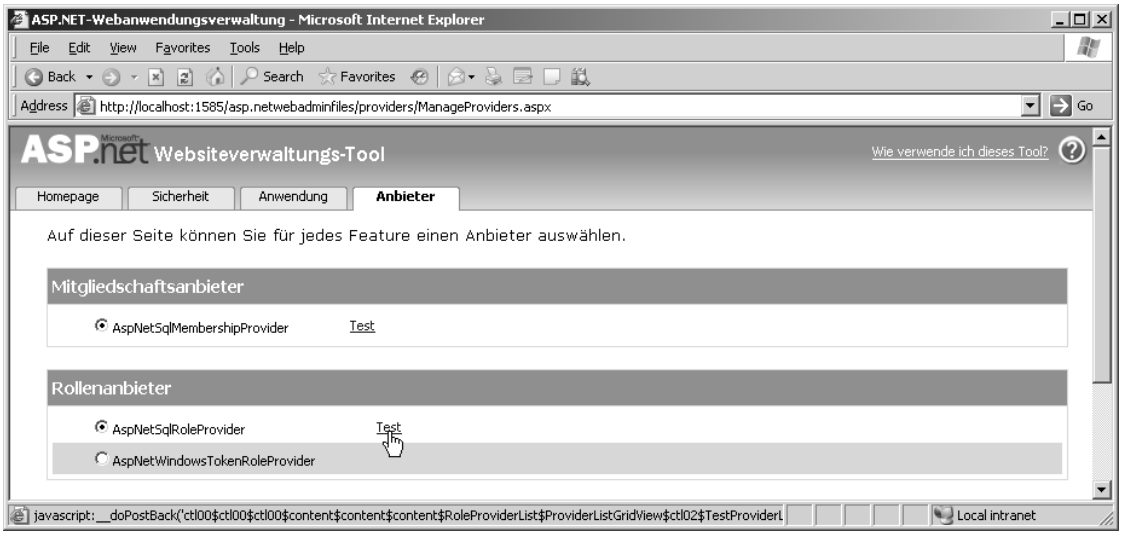



**Abbildung 8.14** Verwaltung der Provider mit dem webbasierten ASP.NET-Administrationswerkzeug

**TIPP** Sie können die webbasierte Administrationsschnittstelle an Ihre Bedürfnisse anpassen. Den kompletten Quellcode findet man unter:

```
c:\WINDOWS\Microsoft.NET\Framework\v4.0…\ASP.NETWebAdminFiles
```

**ACHTUNG** Das Websiteverwaltungswerkzeug stürzt mit Fehlermeldungen ab, wenn man während der Arbeit mit der Weboberfläche parallel auch Veränderungen an dem Webprojekt in VWD vornimmt. In diesem Fall muss man das Websiteverwaltungswerkzeug neu starten.