Kapitel 6

Webserver für ASP.NET (insbesondere IIS)

In diesem Kapitel:

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

Ein ASP.NET-fähiger Webserver ist sowohl für den Test als auch den Betrieb einer ASP.NET-Webanwendung notwendig.

HINWEIS Soweit nicht anders erwähnt, wird in diesem Buch davon ausgegangen, dass während der Entwicklungszeit IIS Express und zur Laufzeit Internet Information Services (IIS) zum Einsatz kommen.

ASP.NET-fähige Webserver

ASP.NET-fähige Webserver sind:

- IIS (Internet Information Services), das in Windows Server-Betriebssystemen, aber auch in Windows Client-Betriebssystemen mitgeliefert wird (als Installationsoption)
- Der IIS Express-Webserver, der alle Funktionen von IIS bietet, aber nicht als Dienst, sondern als einfache Anwendung läuft. IIS Express ist nun der Standard-Webserver in Visual Studio.
- ASP.NET Development Webserver, der in früheren Versionen den Standard-Webserver in Visual Studio darstellte, wird weiterhin mit Visual Studio 2012 automatisch auf dem Entwicklungscomputer installiert und steht als Option zur Verfügung.).
- Apache
- Einen ASP.NET-fähigen Webserver als Softwarekomponente bietet die Firma Intorel mit *Active Server Component* [INT01]

HINWEIS Wie Sie in Visual Studio den verwendeten Webserver festlegen (und damit auch zwischen IIS Express und ASP.NET Development Webserver umschalten) und konfigurieren, erfahren Sie im Abschnitt »Konfiguration des Webservers in der Entwicklungsumgebung« (ab Seite 226).

ASP.NET Development Server

ASP.NET Development Server (alias *Visual Studio Web Development Server, WebDev.WebServer.exe*) basiert auf dem bereits für ASP.NET 1.x verfügbaren in .NET geschriebenen Webserver *Cassini* und ist nur lokal erreichbar. Es ist auch nicht möglich, ASP.NET Development Server entsprechend umzukonfigurieren, um diese Einschränkung zu umgehen, da diese Restriktion fest im Programmcode definiert ist. Entsprechend kann ASP.NET Development Server keine HTTP-Aufrufe von anderen Computern bedienen. Der Port wird standardmäßig beim Start aus Visual Studio automatisch vergeben.

ASP.NET Development Server wird automatisch verwendet, wenn der Entwickler beim Anlegen eines Webprojekts einen Dateisystempfad statt eines HTTP-Pfads angibt. Den Webserver kann man nur geringfügig konfigurieren über das Eigenschaftenfenster (siehe unten).

Starten und Stoppen des Webservers

Der Webserver beendet sich nach einiger Zeit automatisch oder kann manuell über ein Symbol im Windows-Infobereich gestoppt werden. Er wird automatisch heruntergefahren, wenn Visual Studio geschlossen wird. Der Webserver kann über *WebDev.WebServer.exe* auch manuell für ein bestimmtes Verzeichnis auf einem bestimmten Port gestartet werden. ASP.NET Development Server ist nur in der Lage, ASP.NET-Webseiten zu verarbeiten, keine klassischen ASP-Seiten oder andere dynamische Webtechniken.

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

Konfiguration des Webservers

Die Konfiguration erfolgt über die Einstellungen im Eigenschaftenfenster, wenn man das Stammverzeichnis der Website im Projektmappen-Explorer auswählt. Bei Anwendungen nach dem Webanwendungsmodell kann der Webserver auch über die Eigenschaftenseiten des Projekts auf der Registerkarte *Web* konfiguriert werden.

TIPP Um zu verhindern, dass ASP.NET Development Server beim Start von Visual Studio immer eine andere Portnummer erhält, muss man in den Eigenschaften des Stammverzeichnisses des Webs die Option *Dynamische Ports verwenden* (*Use dynamic ports*) auf *False* setzen. Auch der virtuelle Pfad lässt sich hier anpassen. Es bietet sich an, den virtuellen Pfad auf »/« zu setzen, wenn man später die Webanwendung in IIS direkt in einem virtuellen Webserver ohne zusätzliche Pfadangabe betreiben will.

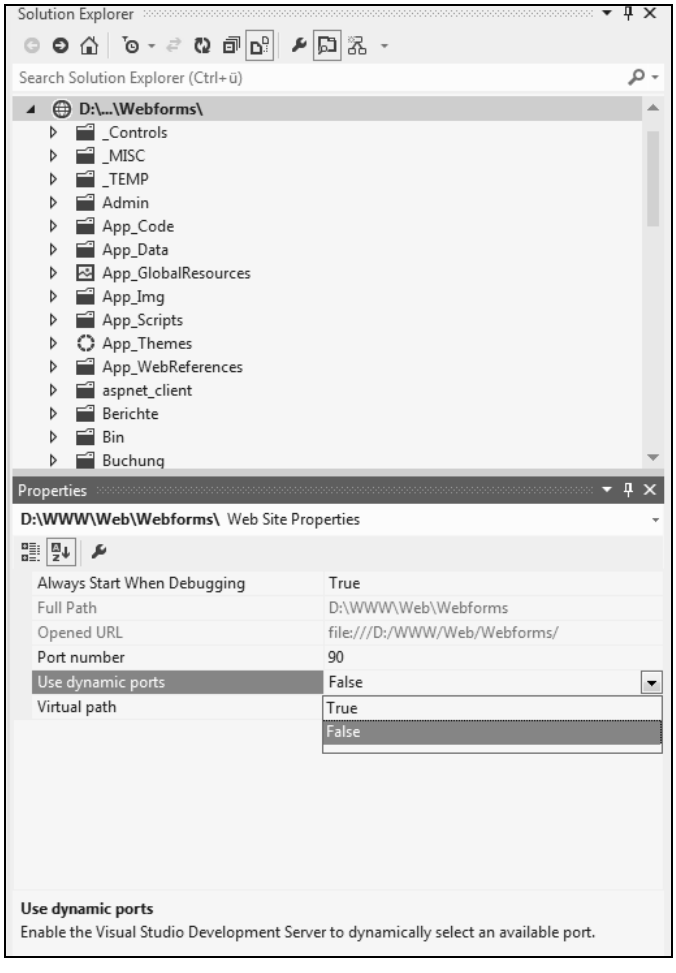

Abbildung 6.1 Festlegung eines statischen Ports für den eingebauten Webserver

Probleme

Leider stürzt ASP.NET Development Server gelegentlich mit unklaren Fehlermeldungen ab.

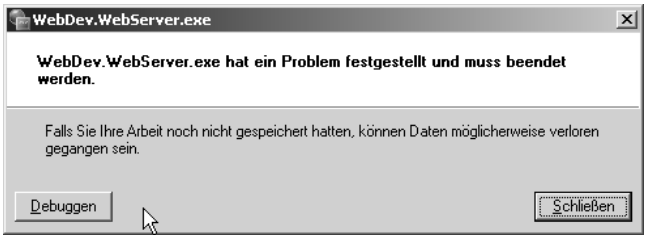

Abbildung 6.2 Totalabsturz von ASP.NET Development Server

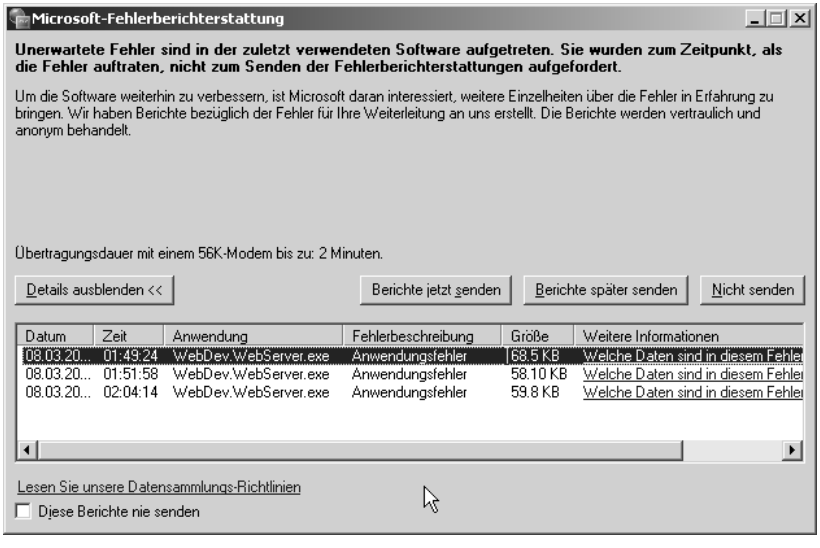

Abbildung 6.3 Mehrere Totalabstürze von ASP.NET Development Server

Internet Information Services (IIS)

IIS (*Internet Information Services*) ist eine Sammlung von Diensten, um selbst Internetangebote bereitzustellen. Der *World Wide Web-Dienst* (WWW-Dienst) erlaubt es, einen eigenen Webserver zur Bereitstellung von Webseiten über das Hypertext Transfer Protocol (HTTP) zu betreiben. Der Name *World Wide Web-Dienst* ist etwas irreführend, denn auch lokale Webangebote (Intranets) können mit dem Dienst bereitgestellt werden. Besser wäre daher Webservice. Der *FTP-Dienst* erlaubt die Ablage und das Herunterladen von Dateien und der *SMTP-Dienst* den Versand von E-Mails.

HINWEIS Offiziell steht die Abkürzung IIS inzwischen nicht mehr für *Internet Information Server*, sondern für *Internet Information Services*, um zu verdeutlichen, dass IIS kein eigenständiges Produkt, sondern ein Teil von Windows ist. Die Bezeichnung *Server* ist aber immer noch weit verbreitet und auch Microsoft selbst verwendet immer noch auf vielen Webseiten die alte Bezeichnung.

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

IIS wird auch in ASP.NET 4.5 weiterhin der hauptsächliche Webserver für den Betrieb von ASP.NET-Webanwendungen bleiben. Die Nutzung von IIS als Webserver und von Microsoft SQL Server als Datenbank ist die klassische Variante zur Webprogrammierung auf der Windows-Plattform. Einige Autoren sprechen von *WIMA* (Windows, Internet Information Services, Microsoft SQL Server/MSDE, ASP.NET) als Alternative zu *LAMP* (Linux, Apache, MySQL, PHP).

Verbreitung von IIS

Die amerikanische Firma Netcraft erhebt monatlich eine Statistik über die Verbreitung von Webserversoftware. Auch wenn nicht das komplette Internet (und keine Intranets) erfasst werden (sondern »nur« rund 633 Millionen Websites) kann, lässt sich dies dennoch als Indikator für die Verbreitung von IIS (in der Statistik aufgeführt als *Microsoft*) heranziehen.

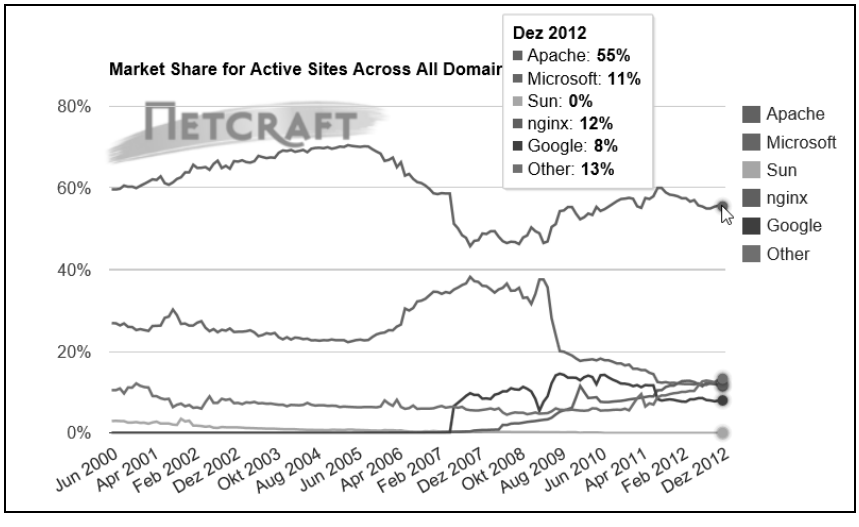

Abbildung 6.4 Verbreitung der Webserversoftware (Quelle: [NETCRAFT01])

Versionen von IIS

IIS gibt es seit Windows NT 4 (damals im so genannten NT 4 Option Pack). Seit Windows 2000 gehört es zum Betriebssystem. Tabelle 6.1 zeigt die im Umlauf befindlichen Versionen.

IIS-Version	Betriebssysteme	Nutzbare ASP.NET-Versionen
5.0	Windows 2000 Professional und Server	1.0, 1.1, 2.0
5.1	Windows XP	1.0, 1.1, 2.0, 3.5, 4.0
6.0	Windows Server 2003	1.0, 1.1, 2.0, 3.5, 4.0
7.0	Windows Vista und Windows Server 2008	1.1, 2.0, 3.5, 4.0
7.5	Windows 7 und Windows Server 2008 R2	1.1, 2.0, 3.5, 4.0, 4.5
8.0	Windows 8 und Windows Server 2012	3.5, 4.5 (ASP.NET 4.0 kann nicht auf diesen Systemen installiert werden, da dort .NET 4.5 schon vorinstalliert ist)

Tabelle 6.1 IIS-Versionen

ACHTUNG Die IIS-Versionen auf Clientbetriebssystemen sind auf zehn gleichzeitige Verbindungen beschränkt, stellen also reduzierte Programmversionen dar. Dabei bedeuten zehn gleichzeitige Verbindungen nicht einmal, dass tatsächlich zehn Benutzer gleichzeitig bedient werden, da moderne Webbrowser durch mehrere Verbindungen Webseiten schneller laden. Daher eignen sich diese IIS-Varianten nicht zum Hosting von produktiven Webanwendungen, sondern nur als Entwicklungssystem.

Kernfunktionen von IIS

Kernfunktionen von IIS sind:

- HTTP-Server für statische Inhalte (z. B. *.htm*, *.jpg*, *.gif*, *.pdf*) und dynamische Inhalte (z. B. ASP.NET, PHP)
- Verwalten von Prozessen und Erneuern von Prozessen nach definierten Regeln
- Optionale Authentifizierung von Aufrufen
- Absicherung von Verbindungen durch Secure Socket Layer (SSL)
- Beschränkung der Zugriffe von Clients
- Protokollierung aller Aufrufe
- Festlegung der Ausgaben im Fehlerfall

Als dynamische Webtechniken unterstützt IIS von Hause aus:

- Internet Information Service Application Programming Interface (ISAPI)
- Common Gateway Interface (CGI)
- Internet Database Connector (IDC)
- Server Side Includes (SSI)
- Active Server Pages (ASP)
- Active Server Pages .NET (ASP.NET)

Neuerungen seit IIS 7.x

Mit Windows Vista und Windows Server 2008 hat Microsoft den IIS-Webserver komplett überarbeitet. Wesentliche neue Funktionen von IIS 7.0 gegenüber dem Vorgänger IIS 6.0 sind:

- Der Server ist komponentenorientiert, das heißt, es müssen nur wirklich die benötigten Teile von IIS installiert werden
- IIS ist nun auch Server für andere Protokolle außer HTTP/HTTPS
- Der IIS-Manager wurde komplett überarbeitet
- ASP.NET und IIS benutzen nun das gleiche Konfigurationskonzept (siehe Kapitel 8 »Konfiguration«)
- ASP.NET und IIS verwenden die gleiche Verarbeitungspipeline
- Es gibt nun Programmierschnittstellen zur automatisierten Administration (.NET und WMI) und Systemüberwachung

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

IIS 7.5 (in Windows Server 2008 R2 und Windows 7) bietet zwei zentrale neue Funktionen:

- FTP-Server
- Autostart für ASP.NET-Anwendungen (siehe Abschnitt »Autostart« auf Seite 221)

Alle ISAPI-Erweiterungen von früheren IIS-Versionen sollten auch in IIS 7.x funktionieren, ebenso die meisten ISAPI-Filter. Ausnahmen sind Filter, die *READ_RAW* verwenden. Folgende Funktionen aus IIS 6.0 hat Microsoft allerdings gestrichen:

- IIS-5.0-kompatibler Isolationsmodus
- Internet Database Connector (IDC)
- Server-Side Image Maps
- ASP-Content-Rotator-Komponente
- Convlog-Werkzeuge zum Konvertieren von Protokolldateien
- Unterstützung für PICS Content Rating
- Authentifizierung mit Microsoft Passport
- IIS Cluster Admin Tool Extension für Windows Server

HINWEIS In IIS (Internet Information Services) ab Version 7.x ist das Deployment von Webanwendungen dadurch verbessert, indem IIS auf das verteilte XML-basierte Konfigurationssystem von ASP.NET zurückgreift.

Neuerungen seit IIS 8.0

IIS 8.0 (in Windows Server 2012 und Windows 8) bietet folgende zentrale neue Funktionen, insbesondere für das Hosting zahlreicher IIS-Websites auf einem Windows Server:

- Unterstützung für das Websocket-Protokoll (*http://dev.w3.org/html5/websockets/*)
- Erweiterung der in IIS 7.5 eingeführten Autostartfunktion: Während der Initialisierungsphase kann IIS statische Seiten (»Splash Screen«) anzeigen. Man kann über URL-Rewriting-Muster verschiedene Splash Screens definieren.
- Erweiterung der Einstellungen der CPU-Nutzungsbeschränkung für IIS-Anwendungspools: Nun wird eine Website, die zu viel CPU nutzt, nicht mehr abgeschaltet, sondern gedrosselt. Die Drosselung ist an Benutzer gebunden und wirkt für alle Anwendungspools, die unter demselben Benutzer laufen.
- Dynamische IP-Adressbeschränkungen: Blockierung von IP-Adressen anhand der gleichzeitigen Anzahl von Anfragen oder der Anzahl der Anfragen in einem bestimmten Zeitraum
- Festlegung der Antwort an den Client bei Zutreffen einer IP-Adressbeschränkung (bisher gab es immer als Antwort »403.6 Forbidden«, nun sind auch »401 Unauthorized«, »404 Not Found« oder ein Abbruch der Verbindung möglich)
- Die Einstellung *Proxymodus aktivieren* (*Enable Proxy Mode*) für IP-Adressbeschränkungen erlaubt die Berücksichtigung des HTTP-Headers »X-Forwarded-For (XFF)«, der die IP-Adresse des Aufrufers enthält, wenn der Zugriff über einen Proxy erfolgt

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

- Unterstützung für Server Name Indication (SNI) zur Einbeziehung eines Hostnamens in die Verhandlungsphase eines SSL-Zertifikats (wenn sich mehrere Websites eine IP-Adresse teilen). Das Feature funktioniert aber nur mit neueren Browsern und Betriebssystemen (z. B. unterstützen Windows XP oder Internet Explorer 6.0 kein SNI).
- Verbesserung des Ladeverhaltens für SSL-Zertifikate, indem nun nicht mehr alle Zertifikate, sondern nur noch die benötigten SSL-Zertifikate geladen werden
- Zentraler Zertifikatsspeicher als *.pfx*-Dateien in einem Netzwerkpfad (Central Certificate Store – CSS). Die Zertifikatszuordnung erfolgt nach Namenskonventionen.
- Erweiterung der grafischen Administrationsmodule im IIS-Manager für ASP.NET-Einstellungen der *web.config*-Datei. Nun können für ASP.NET 4.x-Webanwendungen auch Provider, ASP.NET-Membership-Benutzer und ASP.NET-Rollen sowie einige in ASP.NET 4.x hinzugekommene Kompilierungs- und Seiteneinstellungen über den IIS-Manager verwaltet werden.
- Erweiterte Einstellungen für Skalierung bei Einsatz von Non-Uniform Memory Access (NUMA)-Hardware
- Beschränkung der Anzahl der FTP-Anmeldeversuche in einem definierbaren Zeitraum (FTP IP Restrictions Module)
- Das Ladeverhalten der IIS-Konfigurationsdateien (*applicationHost.config*, *web.config*) wurde verbessert und erlaubt nun größere Dateien

ACHTUNG In IIS 8.0 kann man nur ASP.NET 3.5 und ASP.NET 4.5 nutzen. ASP.NET 3.5 wird aber in den Einstellungen eines Applications Pools weiterhin als »ASP.NET 2.0« angezeigt.

Komponentenorientierung

Microsoft nennt IIS seit Version 7.0 einen *komponentenbasierten Server*, weil er aus einzelnen unabhängigen Softwarekomponenten zusammengebaut ist. Das bietet den Vorteil, dass nur die wirklich benötigten Funktionen installiert werden müssen. Im Unterschied zum monolithischen Vorgängern besteht IIS ab Version 7.x aus einem kleinen Webserverkern (Web Core Server) und rund 40 IIS-Modulen für Netzwerkprotokolle, Protokollierung, Konfiguration, Authentifizierungsverfahren und Diagnose (Abbildung 6.5).

HINWEIS Im Hinblick auf Sicherheit reduziert dies die Angriffsfläche und erhöht die Sicherheit des Webservers gegenüber früheren Versionen von IIS, die nur für die Anwendungsentwicklungsframeworks eine Möglichkeit zum Deaktivieren von Merkmalen boten. Neben höherer Sicherheit ist von einem auf die notwendigen Module reduzierten Webserver auch eine bessere Leistung zu erwarten.

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

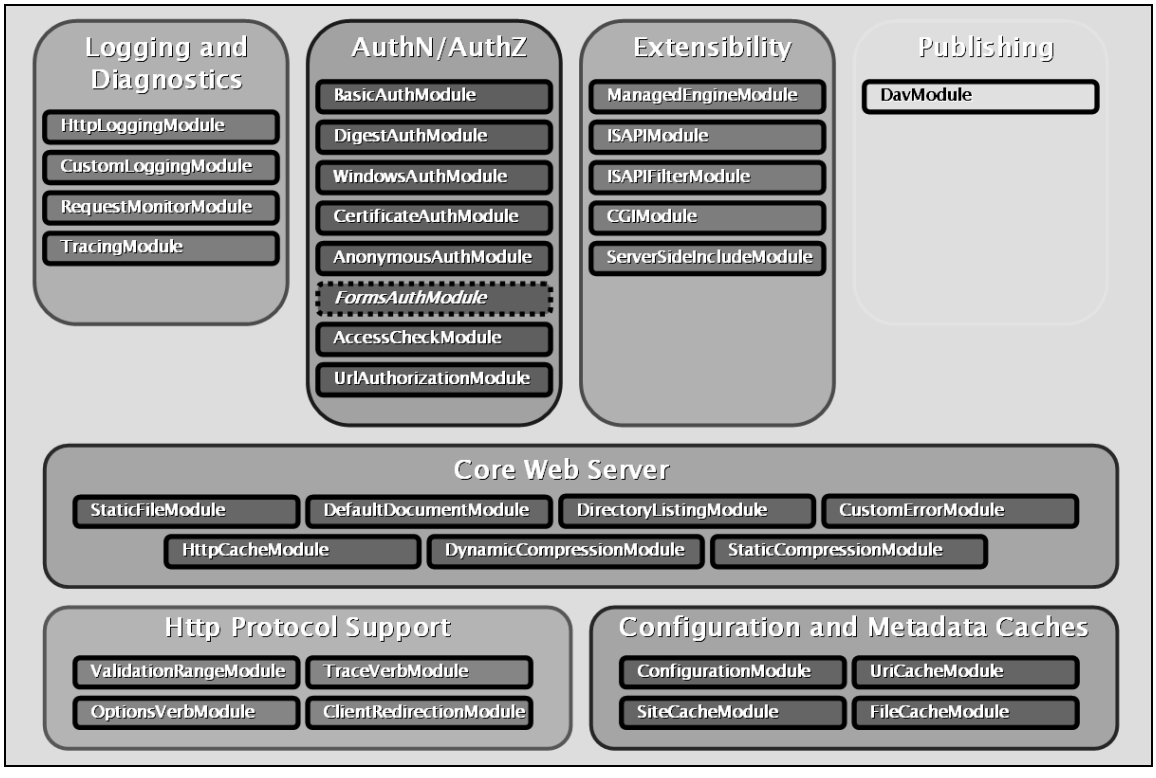

Abbildung 6.5 Die Architektur von IIS ab Version 7.x basiert auf Komponenten (Quelle: PowerPoint-Präsentation von Microsoft)

Installation von IIS

Auf allen Betriebssystemen nach Windows 2000 muss IIS explizit installiert werden. Diesen Strategiewechsel hat Microsoft nach der Verbreitung des Wurms *Code Red* vollzogen. Für die Infektion war eine Sicherheitslücke in IIS verantwortlich. Dabei kam erschwerend dazu, dass viele Windows-Systeme nur deshalb infiziert wurden, weil auf diesen ein Webserver lief, der zwar standardmäßig installiert, aber gar nicht gebraucht wurde.

Unter Windows XP und Windows Server 2003 erfolgt die Installation der IIS 6.0 in der *Systemsteuerung* über das Symbol *Software*. Über die *Windows-Komponenten* aktivieren Sie den Eintrag *Internet Informationsdienste (IIS)*. Unter den Details wählt man dann die zu installierenden Dienste sowie die Zusatzwerkzeuge mit aus.

In Windows Vista (IIS 7.0), Windows 7 (IIS 7.5) und Windows 8 (IIS 8.0) geschieht die Installation weiterhin in der *Systemsteuerung*. Dort wählen Sie aber jetzt *Programme und Funktionen* und dann *Windows-Funktionen aktivieren oder deaktivieren* an (Abbildung 6.6). In Windows Server 2008 (IIS 7.0) und Windows Server 2008 R2 (IIS 7.5) sowie Windows Server 2012 (IIS 8.0) erfolgt die Installation aus dem Server-Manager heraus durch Installation der Serverrolle *Webserver (IIS)* (*Web Server (IIS)*) (Abbildung 6.7) und des Rollendiensts *ASP.NET* (Abbildung 6.8). Bei der Installation fordert Windows Server als Grundlage die Installation von WAS (*Windows Activation Service*). WAS ist in der neuen Windows-Generation der Systembaustein, der für IIS die Anwendungspools und Prozesse verwaltet.

In dem Installationsfenster ab IIS 7.x kann der Administrator sehr viel genauer als in der Vergangenheit die einzurichtenden Funktionen auswählen. Neben Programmierframeworks wie ASP, ASP.NET, CGI und ISAPI lassen sich in den Bereichen *Gemeinsam genutzte HTTP-Features* (*Common HTTP Features*), *Systemzustand und Diagnose* (*Health and Diagnostics*), *Sicherheit* (*Security*) und *Webverwaltungstools* (*Management Tools*) die gewünschten Module selektieren. Im Bereich Sicherheit sind verschiedene Authentifizierungsverfahren wie z.B. Basic, Windows, Digest oder Zertifikate wählbar. Bei den Management-Diensten steht unter anderem zur Wahl, ob sich IIS auch mit den Verfahren von IIS, also mit Konsole, Skript oder per WMI, verwalten lassen soll und ob eine Fernverwaltung von IIS über einen Management-Dienst erlaubt sein soll.

Die komponentenorientierte Architektur erlaubt auf jeder Ebene (Webserver, Website, Anwendung oder Verzeichnis), Modulsätze zu erstellen. So ist es beispielsweise möglich, einen Webserver zu betreiben, der ausschließlich Windows-NTLM-Authentifizierung, statische Webseiten, Kompression und Protokollierung beherrscht.

ACHTUNG Bei der Installation müssen Sie *ASP.NET* in der Kategorie *Anwendungsentwicklungsfeatures* (*Application Development*) auswählen, wenn Sie mit ASP.NET, unabhängig von der Version, arbeiten wollen!

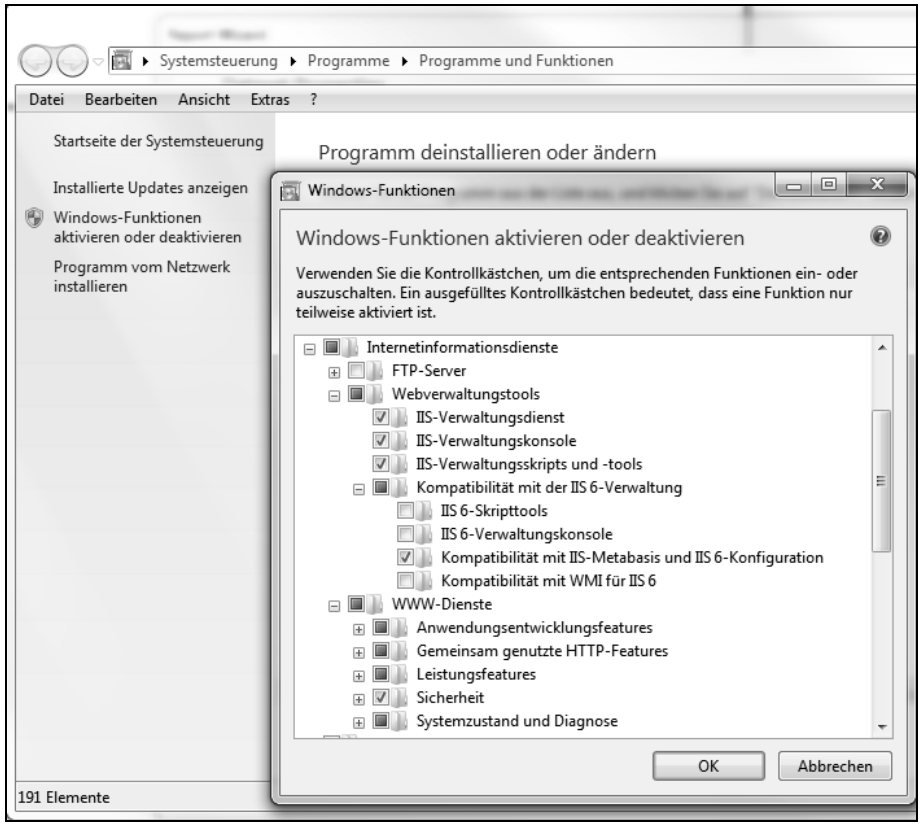

Abbildung 6.6 Installation von IIS 7.5 in Windows 7

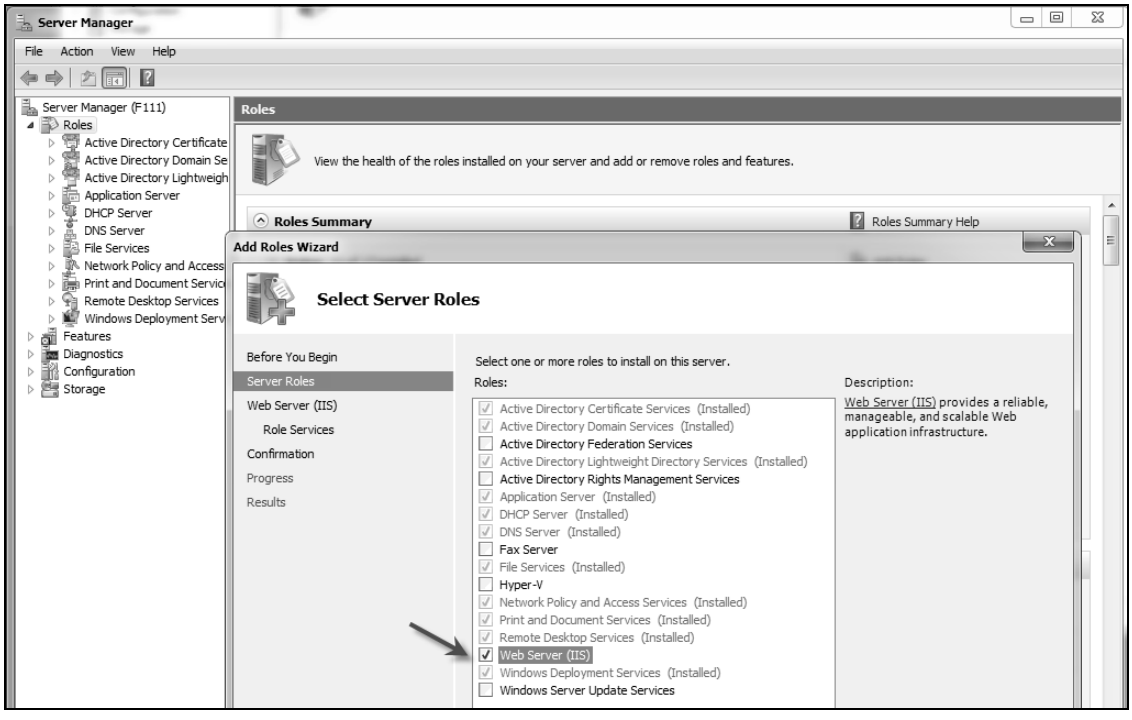

Abbildung 6.7 Installation von IIS 7.5 in Windows Server 2008 R2

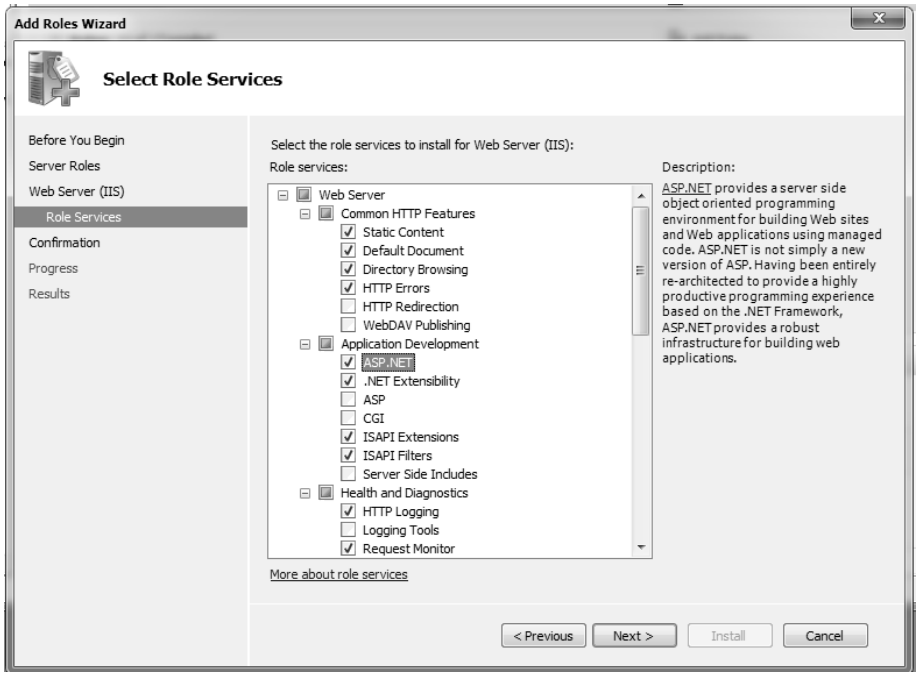

Abbildung 6.8 Aktivierung der Features von IIS 7.5 in Windows Server 2008 R2

Integration zwischen ASP.NET und IIS

Die Integration zwischen ASP.NET und IIS erfolgt automatisch, wenn IIS bei der Installation von .NET Framework Redistributable bereits vorhanden ist. Sofern IIS erst später (nach der Installation von .NET Framework Redistributable) installiert wird, muss die Integration durch Ausführung der Kommandozeilenanwendung *aspnet_regiis.exe* mit dem Parameter *–i* manuell angestoßen werden.

Test der Installation

Um zu testen, ob IIS erfolgreich installiert wurde, geben Sie im Browser *http://localhost* ein. Danach sollten Sie eine statische Seite mit dem Text »IIS 8 microsoft.com/web« sehen.

HINWEIS Die Willkommensseite befindet sich physisch im Verzeichnis *c:\inetpub\wwwroot*, das bei der Installation angelegt wurde. Grundsätzlich ist dies auch der Speicherort für Ihre eigenen Webanwendungen. Es ist aber auch möglich, andere Standorte auf der Festplatte zu nutzen.

IIS-Administration

Der Konfigurationsspeicher von IIS sind XML-Dateien (siehe *C:\Windows\System32\inetsrv\config*). Deren direkte Bearbeitung ist zwar möglich, aber nicht komfortabel und es können leicht Fehler passieren.

IIS-Manager

Das primäre Administrationsinstrument ist der *Internetinformationsdienste (IIS)-Manager* (*Internet Information Services (IIS) Manager*). Der neue IIS-Manager umfasst die Funktionen des bisherigen IIS-Managers, der bisherigen IIS-Manager-Erweiterungen für ASP.NET, die mit .NET Framework installiert wurden, sowie die Funktionen der webbasierten ASP.NET-Verwaltungsoberfläche.

Die Einstellungen sind in Bereiche (*Areas*) und die einzelnen Bereiche wiederum in Kategorien gegliedert (Abbildung 6.9). Dazu erscheinen die Aufgaben, die der aktuellen Auswahl von Bereich und Kategorie entsprechen. Nach Wahl der Aufgabe präsentiert sich entweder eine eigene Oberfläche zur Verwaltung der zugehörigen Einstellungen oder das aus Visual Studio bekannte Eigenschaftsgitter (Abbildung 6.10).

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

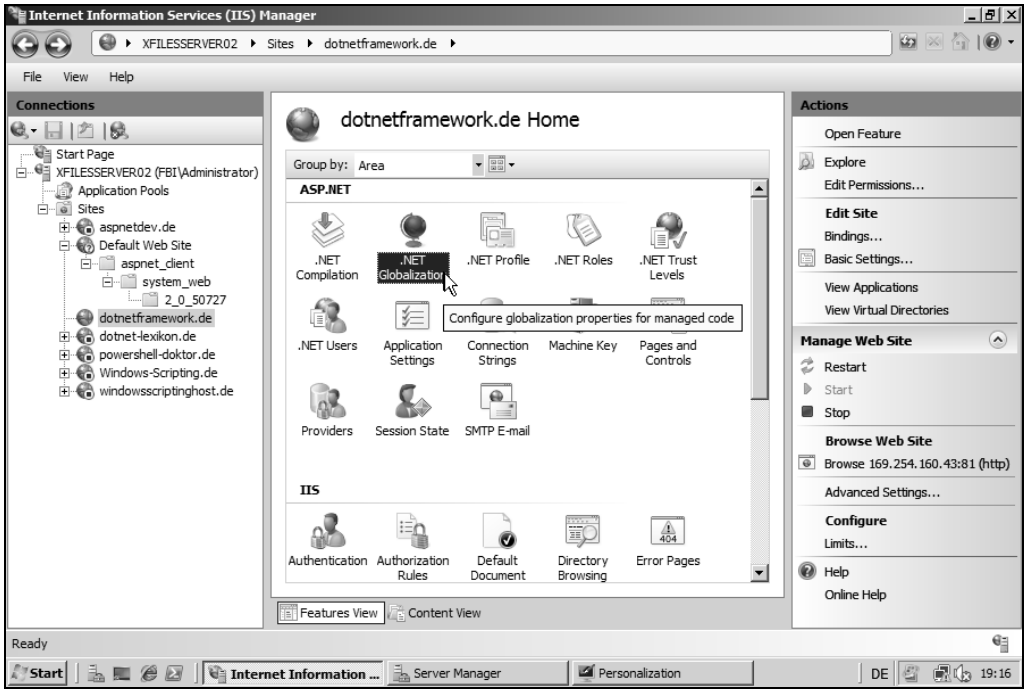

Abbildung 6.9 Die neue IIS-Verwaltungskonsole ist aufgabenbasiert und fasst die Administration von IIS und von ASP.NET zusammen

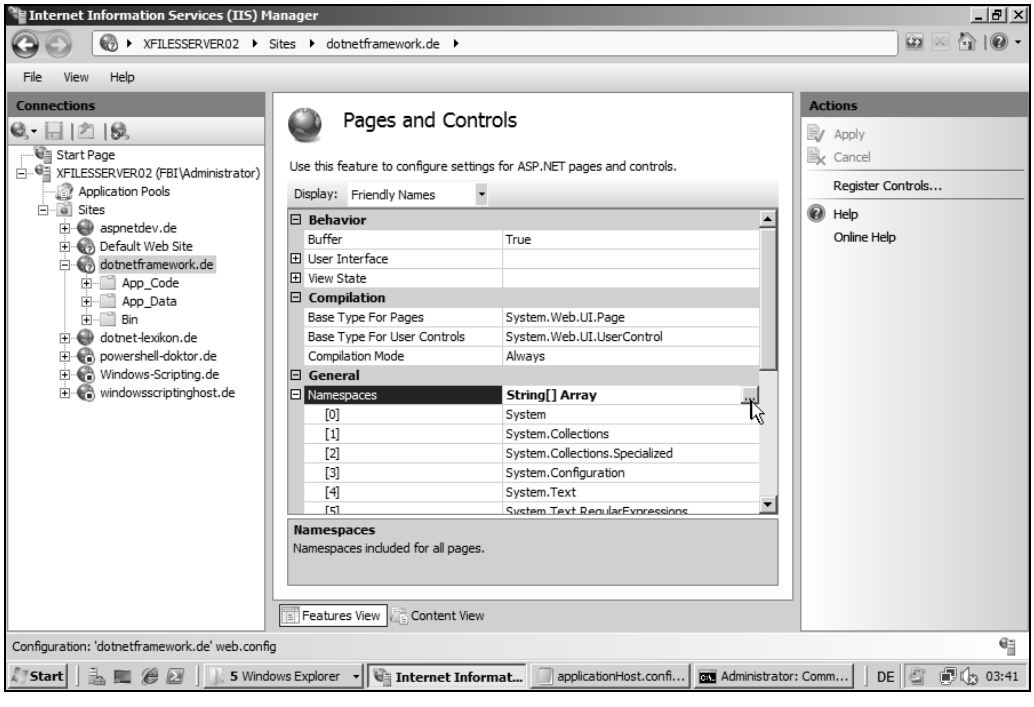

Abbildung 6.10 Einige Einstellungen präsentieren sich im weniger komfortablen Eigenschaftsgitter

Automatisierte Administration und APIs

Zur automatisierten Administration bietet IIS ab Version 7.x das Kommandozeilenwerkzeug *appcmd.exe* und einen neuen Provider für Windows Management Instrumentation (WMI). Microsoft verwendet für den neuen WMI-Provider den Namespace `root/AppServer` und unterstreicht mit dem Namen `AppServer` die neue Bedeutung von IIS.

HINWEIS Der alte WMI-Provider im Namespace `root/MicrosoftIISv2` kann weiterhin optional installiert und verwendet werden.

Für die Laufzeitüberwachung steht auch eine COM-basierte Schnittstelle mit dem Namen *Runtime Status and Control API* (RSCA) zur Verfügung. Mit dieser Komponente lassen sich Anwendungspools, Arbeitsprozesse, Websites und Anwendungsdomänen in IIS überwachen. RSCA ist auch über WMI auslesbar und liefert Daten an den IIS-Manager (*Systemzustand und Diagnose/Arbeitsprozesse* (*Health and Diagnostics/Worker Processes*) sowie *Systemzustand und Diagnose/Ablaufverfolgungsregeln für Anforderungsfehler* (*Health and Diagnostics/Failed Request Tracing*)). Der IIS-Manager zeigt daher nun eine Liste der aktuellen Anfragen. Über eine .NET-basierte Schnittstelle, das Web Management Framework API mit den Namespaces *Microsoft.Web.Management.Server, Microsoft.Web.Management.Host* und *Microsoft.Web.Management.Client* in *Microsoft.Web.Management.dll*, lässt er sich außerdem erweitern.

IIS-Websites (Virtuelle Webserver)

IIS (alias physischer Webserver) besteht aus einem oder mehreren so genannten *virtuellen Webservern*. Ein virtueller Webserver ist die Zuordnung einer Kombination aus IP-Adresse und Portnummer zu einem physischen Verzeichnis auf einem Speichermedium.

Webserver erstellen

Zum Anlegen einer Website wählt man *Website hinzufügen* (*Add Web Site*) im Zweig *Sites*. Hier erfolgt die Zuordnung *IP-Adresse* und *TCP-Portnummer* zum *Dateisystemverzeichnis*. Anzugeben sind:

- Name der Website
- Pfad im Dateisystem (Standort der Webanwendungsdateien)
- Protokollart (HTTP, HTTPS, FTP etc.)
- IP-Adresse
- Portnummer

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

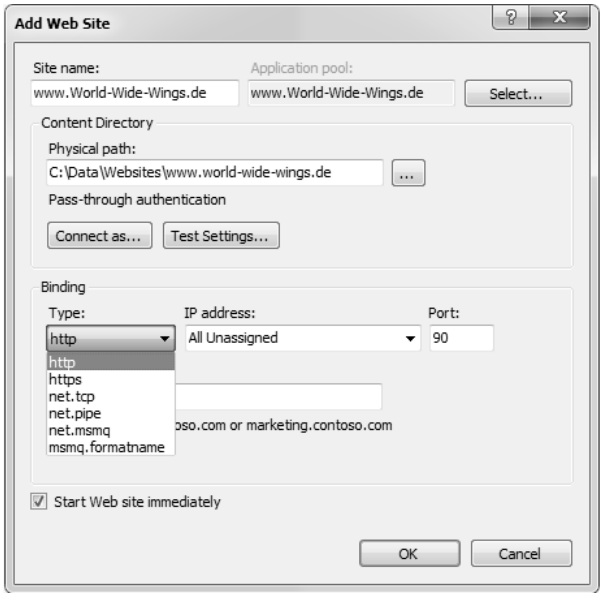

Abbildung 6.11 Anlegen einer Website

WICHTIG Die Einstellung *Keine zugewiesen* (*All Unassigned*) für die IP-Adresse bedeutet, dass der WWW-Dienst auf alle dem Computer in der Netzwerkkonfiguration zugeordneten IP-Adressen reagiert.

Danach erscheint die neue Website in der Liste der Websites. Für die Website kann man die Inhalte ansehen (*Ansicht "Inhalt"* bzw. *Content View*) und die Einstellungen ändern (*Ansicht "Features"* bzw. *Features View*).

Die Liste der virtuellen Webserver lässt sich durch Gruppierung und Filter übersichtlicher gestalten; so wird es leichter, eine große Anzahl von virtuellen Webservern zu verwalten. Neu ist auch, dass jeder virtuelle Webserver eine eigene Liste von Administratoren besitzt, was die Delegation von administrativen Aufgaben vereinfacht. Neben Windows-Benutzerkonten kennt IIS auch eigene Benutzerkonten, damit in Hostingszenarien eine Delegation ohne Vergabe eines Windows-Benutzerkontos möglich ist. Eine Fernadministration von IIS ab Version 7.x ist via HTTP möglich; der DCOM-Zugriff zum Webserver ist also nicht mehr notwendig.

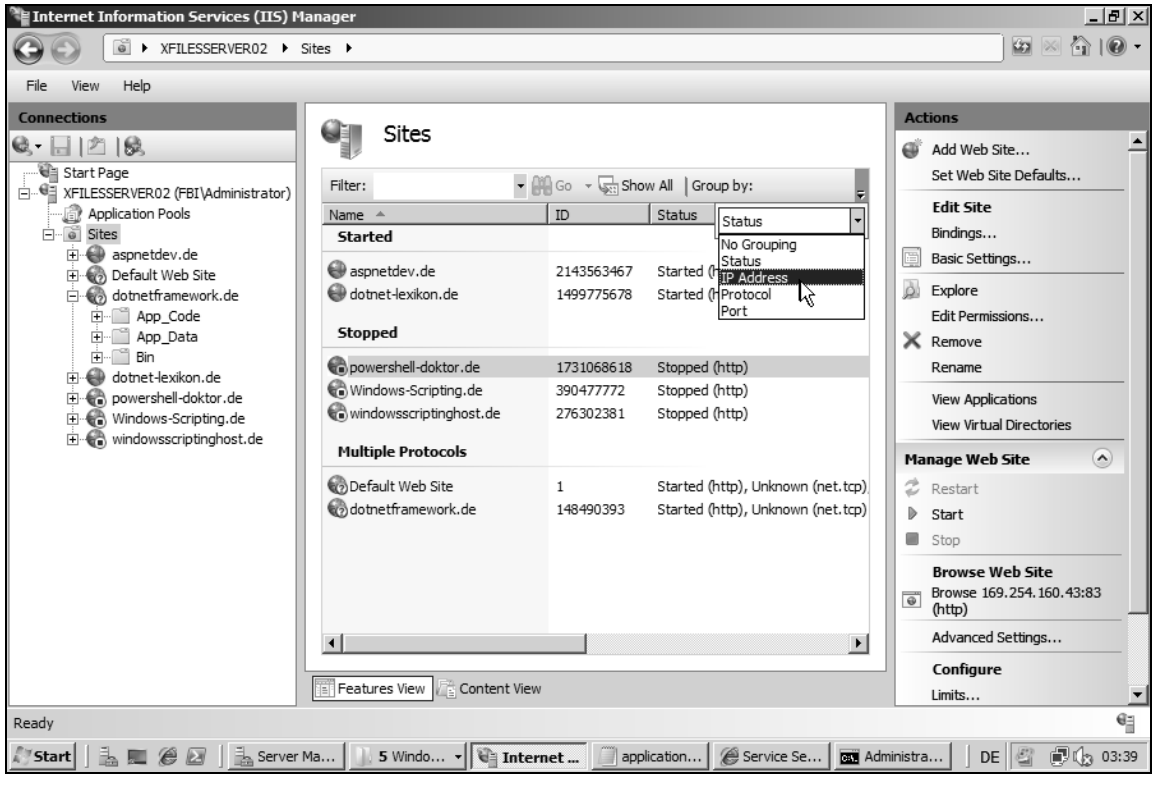

Abbildung 6.12 Gruppieren und Filtern in der Liste der virtuellen Webserver

Websites erstellen per Skript

Das folgende WSH-Skript legt einen Webserver in IIS an:

```
' --- Notwendige Werte festlegen
Const ServerName = "MVPWeb"
Const Port = "80"
Const IP = "192.168.123.14"
Const RootDir = "c:\temp\MVPWeb"
' --- Info
Wscript.echo "Virtuellen Webserver anlegen"
wscript.echo "Dr. Holger Schwichtenberg 2006"
wscript.echo "www.windows-scripting.de"
wscript.echo "-----------------------------------"

' --- Variablendeklarationen
Dim W3SVC
Dim name

' --- Bindung an Webservice
Set W3SVC = GetObject("IIS://E01/w3svc")
' --- Ermittlung einer freien ServerID
For Each objvirtweb In W3SVC
```

```
    If objvirtweb.Class = "IIsWebServer" Then
        Max = objvirtweb.name
    End If
Next
name = Max + 10
WScript.echo "Webserver erhält Nummer: " & name

' --- Erzeugen des virtuellen Webservers
Set objNewWeb = W3SVC.Create("IIsWebServer", name)
objNewWeb.servercomment = ServerName
objNewWeb.KeyType = "IIsWebServer"
objNewWeb.ServerBindings = IP & ":" & Port & ":"
objNewWeb.SetInfo
WScript.echo "Webserver angelegt!"

' -- Erzeugung des Wurzelverzeichnisses für den virtuellen Webserver
Set objRootDir = objNewWeb.Create("IIsWebVirtualDir", "ROOT")
objRootDir.Path = RootDir
objRootDir.AccessScript = True
objRootDir.SetInfo
WScript.echo  "Wurzelverzeichnis zugewiesen!"

' --- Starten des neuen Webservers
objNewWeb.start
WScript.echo  "Webserver gestartet!"
```

Listing 6.1 Skript für den Windows Script Host zum Anlegen eines neuen virtuellen Webservers (Programmiersprache: VBScript)

Wichtige Einstellungen

Eine zu prüfende Einstellung befindet sich in der Rubrik *IIS/Standarddokument* (*IIS/Default Document*): Hier ist festzulegen, welches Dokument an den Webbrowser gesendet werden soll, wenn der Aufrufer nur den Verzeichnisnamen (z.B. *http://localhost/MeineAnwendung*) angibt, aber kein konkretes Dokument (z.B. *http://localhost/MeineAnwendung/Produktliste.aspx*). Falls die Startseite nicht bereits unter den hier aufgelisteten Seitennamen zu finden ist, müssen Sie einen entsprechenden Eintrag hinzufügen.

Soll dem Benutzer gestattet werden, eine Liste der in dem Verzeichnis verfügbaren Dateien einzusehen, um selbst eine Datei auszuwählen, ist dies unter *Verzeichnis durchsuchen* (*Directory Browsing*) einzustellen. Voraussetzung dafür ist, dass das Feature *Verzeichnis durchsuchen* (*Directory Browsing*) in der Installation von IIS aktiviert wurde.

Beschränken der möglichen Clients

Der Zugriff auf einen IIS-Webserver kann so eingeschränkt werden, dass nur bestimmte Netze oder Hostsysteme als Client zugelassen sind. Dies ist möglich in der Funktion *Einschränkungen für IP-Adressen und Domänenname* (*IP Address and Domain Restrictions*). Abbildung 6.13 zeigt ein Szenario, bei dem alle IP-Adressen verboten sind außer IP-Adressen aus dem Bereich 192.168.1.*.

In IIS 8.0 ist es über das neue Dialogfeld *Einstellungen für die dynamische Einschränkung bearbeiten* (*Edit Dynamic IP Restriction Settings*) möglich, eine Blockierung von IP-Adressen anhand der gleichzeitigen Anzahl von Anfragen von einer IP-Adresse oder der Anzahl der Anfragen in einem bestimmten Zeitraum von einer IP-Adresse festzulegen.

Ferner kann in IIS 8. unter *Featureeinstellungen bearbeiten* (*Edit Feature Settings*) die Antwort an den Client bei Zutreffen einer IP-Adressbeschränkung ausgewählt werden. Bisher gab es immer als Antwort »403.6 Forbidden«, nun sind auch »401 Unauthorized«, »404 Not Found« oder ein Abbruch der Verbindung möglich. Die Einstellung *Proxymodus aktivieren* (*Enable Proxy Mode*) erlaubt die Berücksichtigung des HTTP-Header »X-Forwarded-For (XFF)«, der die IP-Adresse des Aufrufers enthält, wenn der Zugriff über einen Proxy erfolgt.

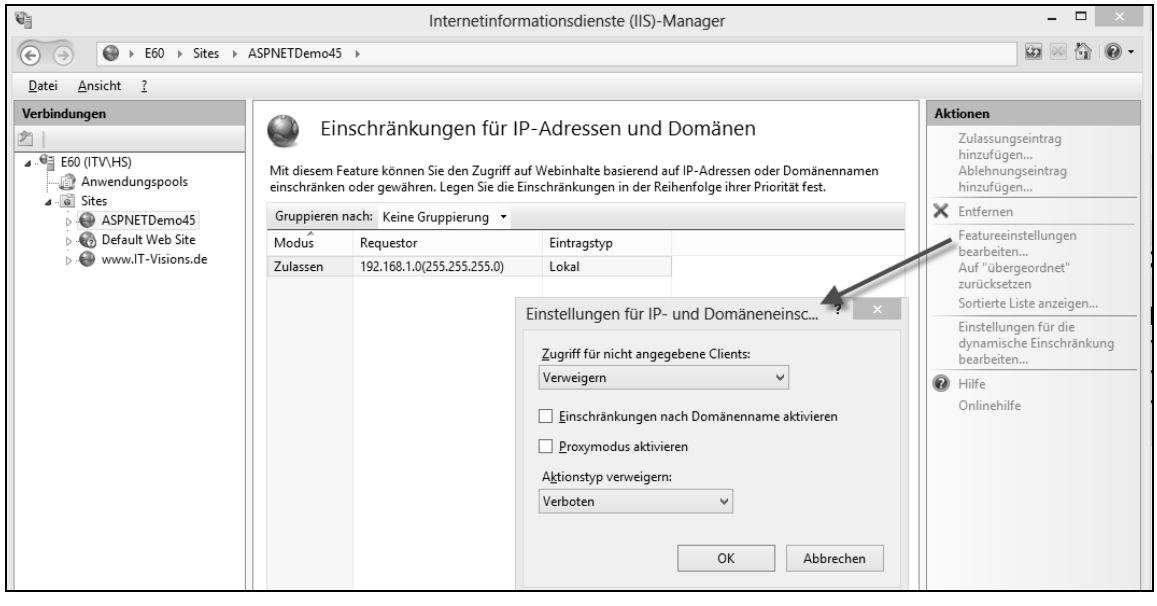

Abbildung 6.13 Zugriffsbeschränkungen auf IP-Ebene

HINWEIS Clients, die durch eine IP-Adressbeschränkung ausgeschlossen sind, erhalten beim Versuch, die Webanwendung anzusprechen, die Fehlermeldung »HTTP Error 403.6 - Forbidden: IP address of the client has been rejected«. Von diesen Clients geht dann nur noch geringere Bedrohung aus.

Authentifizierung

Standardmäßig ist in IIS der anonyme Zugriff aktiviert, das heißt, ein Benutzer muss sich gegenüber IIS nicht authentifizieren. Sofern Sie IIS explizit nicht als öffentlichen Webserver nutzen wollen, sollten Sie den anonymen Zugriff deaktivieren. In einer reinen Windows-Umgebung ist die *Integrierte Windows-Authentifizierung* sinnvoll, die eine Einmalanmeldung (Single Sign-on) innerhalb einer Domäne ermöglicht. Ein Client, der sich nicht an der Windows-Domäne angemeldet hat, hat dann keinen Zugriff auf die Webanwendungen. Sie erreichen die Authentifizierungseinstellungen im IIS-Manager unter *Authentifizierung* (*Authentication*).

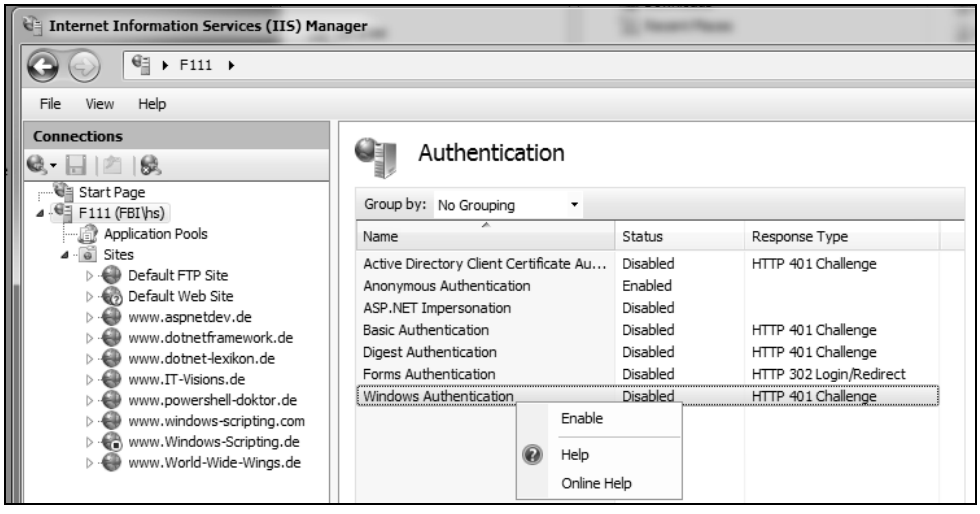

Abbildung 6.14 Änderung der Authentifizierungsverfahren

Secure Socket Layer (SSL)

Um den Zugang zu einer Website mit Secure Socket Layer (SSL) zu ermöglichen, sind folgende Schritte notwendig:

1. Einfügen eines Serverzertifikats (unter *Serverzertifikate* (*Server Certificates)*). Dies muss im Stammverzeichnis des physischen Webservers erfolgen (siehe Symbol mit Servergehäuse im Baum). Man kann hier ein selbstsigniertes Zertifikat anlegen. Für den Betrieb im Internet braucht man aber in der Regel ein Zertifikat einer der in Windows bereits bekannten Zertifizierungsstellen (z.B. VeriSign, Thawte, Deutsche Telekom, GlobalSign und andere).

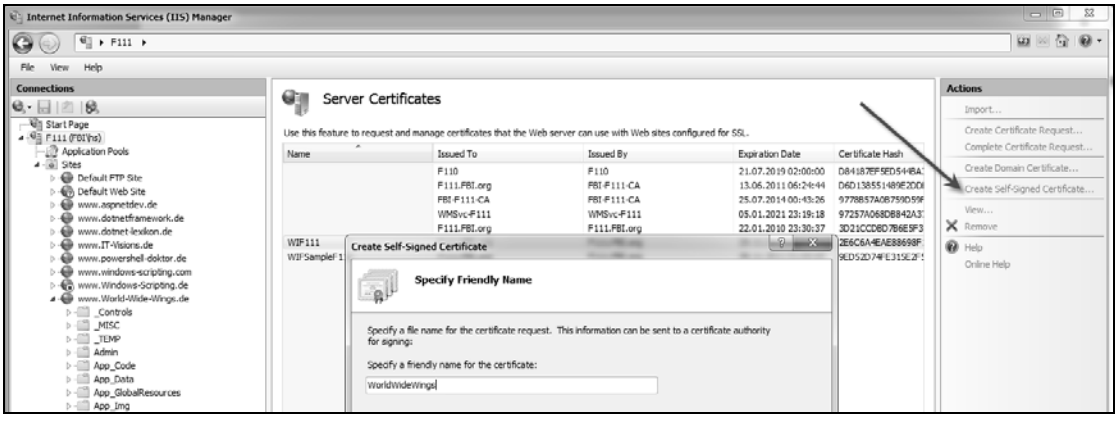

Abbildung 6.15 Anlegen eines Testzertifikats

2. Anschließend wählt man auf der Ebene des virtuellen Webservers *Bindungen* (*Bindings*) und dort *https* mit dem eingefügten Zertifikat.

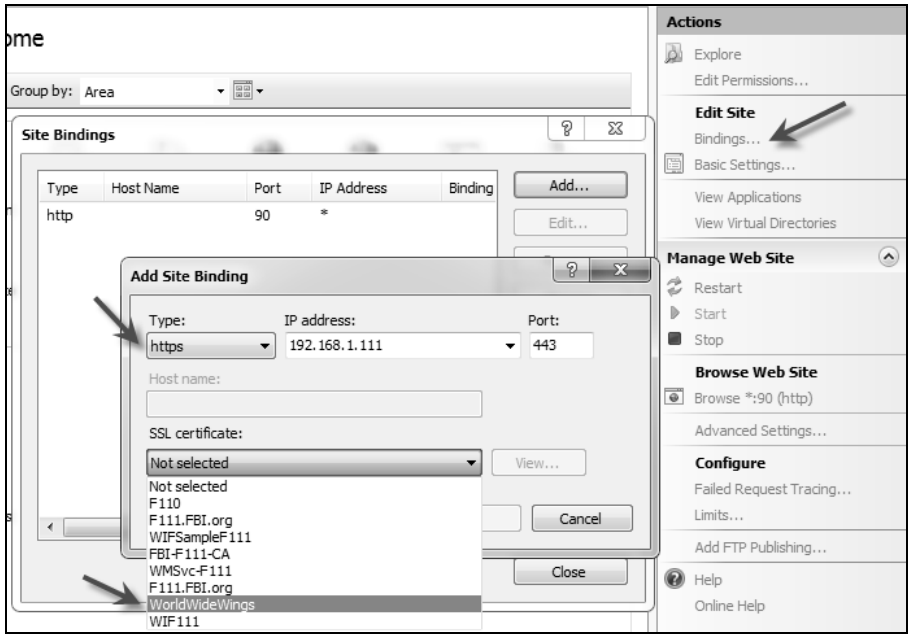

Abbildung 6.16 SSL für eine Website aktivieren

ACHTUNG Wenn die Website nur per SSL erreichbar sein soll, müssen Sie die Bindung für *http* entfernen.

TIPP Oft ist es sinnvoll, dass Benutzer, die eine Site ohne SSL betreten, dann auf SSL umgelenkt werden. In diesem Fall würde man die HTTP-Bindung belassen und in der *global.asax*-Datei eine Umlenkung festlegen:

```
protected void Application_BeginRequest(Object sender, EventArgs e)
{  if (HttpContext.Current.Request.IsSecureConnection.Equals(false))
   {
      Response.Redirect("https://" + Request.ServerVariables["HTTP_HOST"] +
      HttpContext.Current.Request.RawUrl);
   }
}
```

Die Funktion `Response.Redirect("Ziel.aspx")` ist uralt: Sie gab es schon zu Zeiten der klassischen Active Server Pages (ASP), die noch auf dem Component Object Model (COM) basierten. `Response.Redirect()` sendet den Statuscode *HTTP 302 Found* (Temporary Redirect) an den Client unter Angabe der neuen Seitenadresse.

Mehr zu URL-Umlenkungen erfahren Sie in Kapitel 39 »Adressumlenkungen« im Buchteil F »Profiwissen«.

Server für Nicht-HTTP-Protokolle

IIS war bisher ein Web-, Datei-, Mail- und Newsserver. Ab Version 7.0 versteht IIS auch TCP, MSMQ und Named Pipes und wird damit zum allgemeinen Host für Windows Communication Foundation (WCF). Neben dem bereits aus der Vorgängerversion bekannten Kernel-Mode-Listener *HTTP.sys* installiert IIS ab Version 7.x die Listener *NET.TCP*, *NET.PIPE* und *NET.MSMQ*. Beim Eintreffen einer Anfrage in einem der Listener prüft Windows Activation Service (WAS), ob es bereits einen Arbeitsprozess gibt, der die Anfrage bearbeiten kann. Sofern noch keiner vorliegt, erzeugt WAS einen passenden Prozess. Der Aktivierungsdienst kann auf Wunsch verschiedene Protokolle in einem Arbeitsprozess bedienen.

Das Hinzufügen weiterer Protokolle erfolgt über den Menüpunkt *Bindungen bearbeiten* (*Edit Bindings*) im Kontextmenü einer Website im IIS -Manager.

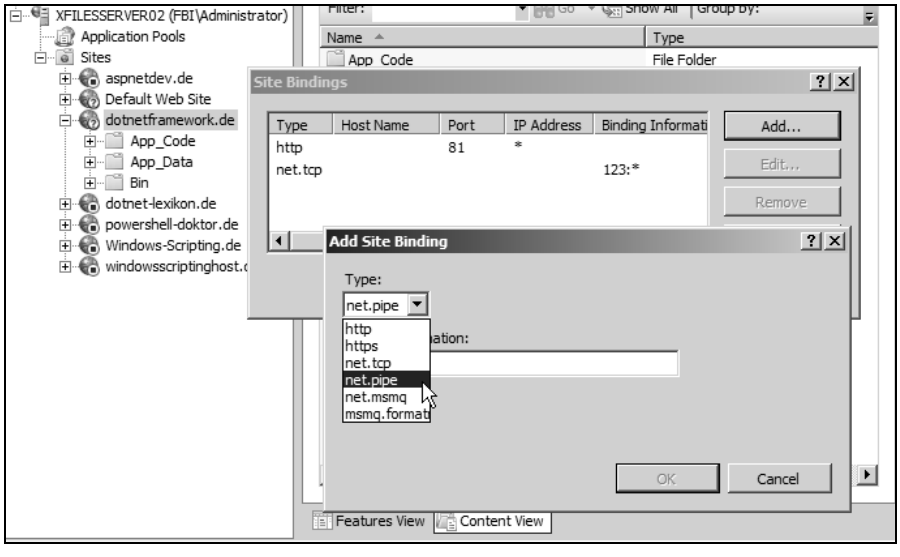

Abbildung 6.17 Aktivieren weiterer Protokolle

IIS-Virtuelle Verzeichnisse

Ein virtuelles Verzeichnis ist ein Konzept, das es erlaubt, jedes Unterverzeichnis als eine eigenständige Anwendung zu betrachten. Hierbei werden *IP-Adresse, TCP-Portnummer* und *relatives Verzeichnis* einem *Dateisystemverzeichnis* zugeordnet. Dabei muss das Dateisystemverzeichnis nicht in der gleichen Beziehung zu dem Basisverzeichnis stehen, wie es der relative Pfad in der URL angibt. Konkret bedeutet dies, dass für jedes virtuelle Verzeichnis ein beliebiger Pfad im Dateisystem gebunden werden kann. Beispielsweise sind folgende Abbildungen möglich:

http://server/ → *c:\website*

http://server/Produkte → *c:\website\Kunden\Produkte*

http://server/Neuigkeiten → *c:\website\News*

Mit virtuellen Webservern lässt sich eine logische Verzeichnisstruktur innerhalb eines virtuellen Webservers unabhängig von der physischen Verzeichnisstruktur auf dem Speichermedium etablieren. Einem physischen Verzeichnis können mehrere virtuelle Webserver und/oder Verzeichnisse zugeordnet werden.

Abbildung 6.18 veranschaulicht den Zusammenhang von virtuellen Webservern (z.B. *http://1.2.3.5:1234*), virtuellen Verzeichnissen (z.B. *http://1.2.3.5:1234/Kundendaten*) und physischen Verzeichnissen (z.B. *e:\Daten\Kunden*) an einem Beispiel. Die auf Windows 2000 Professional und Windows XP verfügbaren Varianten von IIS unterscheiden sich von den jeweiligen Serverversionen dadurch, dass es dort nur genau einen virtuellen Webserver (*Standardwebsite*) geben darf. Diese Restriktion existiert in IIS ab Version 7.x in Windows Vista, Windows 7 und Windows 8 nicht mehr.

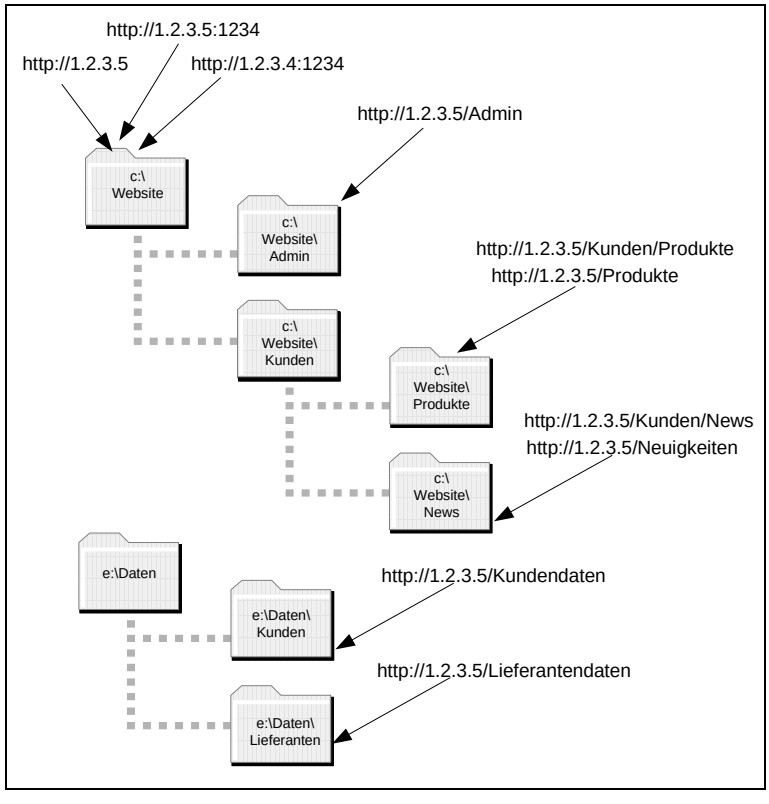

Abbildung 6.18 Das Konzept virtueller Verzeichnisse

Zum Anlegen eines virtuellen Verzeichnisses rufen Sie nach Auswahl einer Website den Eintrag *Virtuelles Verzeichnis hinzufügen* (*Add Virtual Directory*) auf. In früheren Windows-Versionen konnte man auch im Windows-Explorer in der Registerkarte *Webfreigabe* (*Web Sharing*) ein virtuelles Verzeichnis anlegen. Diese Möglichkeit bietet Microsoft seit Windows Vista nicht mehr an.

Abbildung 6.19 Anlegen eines virtuellen Verzeichnisses

Das folgende WSH-Skript legt ein virtuelles Verzeichnis in IIS an:

```
' Eingabedaten
Const RootDir = "c:\temp\MVPWeb"
Const DirName = "MVPWeb"
Const Wurzel = "IIS://EO1/w3svc/1/root" ' (1 = Standardwebserver)

' --- Info
Wscript.echo "Virtuelles IIS-Verzeichnis anlegen"
wscript.echo "Dr. Holger Schwichtenberg 2006"
wscript.echo "www.windows-scripting.de"
wscript.echo "---------------------------------"

' --- Bindung an virtuellen Webserver
Set Web = GetObject(Wurzel)
' --- Erzeugen eines neuen Verzeichnisses
Set objRootDir = Web.Create("IIsWebVirtualDir", DirName)
' --- Festlegen des Pfades
objRootDir.Path = RootDir
' --- Setzen der Eigenschaften
objRootDir.AccessScript = True
objRootDir.SetInfo
objRootDir.AccessWrite = False
objRootDir.AccessRead = True
objRootDir.AuthAnonymous = True
objRootDir.AccessExecute = True
objRootDir.AppCreate True
' --- Speichern
objRootDir.SetInfo
MsgBox "Webanwendung " & DirName & " angelegt!"
```

Listing 6.2 Skript für den Windows Script Host zum Anlegen eines neuen virtuellen Verzeichnisses (Programmiersprache: VBScript)

IIS-Anwendungen

Aus der Sicht des IIS-Webservers ist eine Anwendung etwas anderes als aus der Sicht des Webentwicklers. Für IIS ist eine Anwendung ein Element in der Konfigurationshierarchie, das auch auf ASP.NET wirkt.

Aus Sicht von IIS gehören alle Seiten, die in der gleichen *IIS-Anwendung* laufen, zu einer einzigen ASP.NET-Webanwendung. Eine IIS-Anwendung ist eine Funktion des Webservers zur Abgrenzung von dynamischen Inhalten. In IIS kann man für jede IIS-Anwendung einen eigenen Anwendungspool festlegen und damit einen eigenen Webserver-Arbeitsprozess erzeugen.

Rahmenbedingungen einer IIS-Anwendung

Es gilt:

- Eine IIS-Anwendung hat eine eigene Zustandsverwaltung und es gibt auch keine gemeinsame Authentifizierung. Sie kann Zustandsvariablen (`Session`, `Application`, vgl. Kapitel 11 »Zustandsverwaltung«) sowie Authentifizierungsdaten nicht automatisch, sondern nur manuell mit anderen IIS-Anwendungen teilen.
- Eine IIS-Anwendung besitzt eine eigene Konfigurationshierarchie und erbt keine Konfigurationseinstellungen von anderen IIS-Anwendungen, die im Konfigurationsbaum über ihr stehen (vgl. Kapitel 8 »Konfiguration«)

Eine IIS-Anwendung gehört zu genau einem IIS-Anwendungspool (gilt nur ab IIS 6.0, vgl. Abschnitt »IIS-Anwendungspools«; Seite 215)

ACHTUNG ASP.NET-Entwickler müssen beachten, dass sich durch die Aufteilung einer ASP.NET-Website in mehrere IIS-Anwendungen das Verhalten der ASP.NET-Webanwendung sehr stark ändern kann, da es keine gemeinsame Zustandsverwaltung mehr gibt und die Konfiguration anders wirkt. Bei Fehlern könnten hier z. B. zugangsgeschützte Seiten plötzlich öffentlich sein.

Anlegen einer IIS-Anwendung

Es gilt:

- Ein virtueller Webserver (alias Website) ist immer eine IIS-Anwendung
- Ein virtuelles Verzeichnis kann eine IIS-Anwendung sein
- Ein physisches Verzeichnis kann eine IIS-Anwendung sein

Eine IIS-Anwendung legt man entweder an, indem man *In Anwendung konvertieren* (*Convert to Application*) auf einem Unterverzeichnis in einer Website wählt oder *Anwendung hinzufügen* (*Add Application*). Beide Funktionen fragen nach dem Alias. Durch Vergabe eines Alias erzeugt man gleichzeitig ein virtuelles Verzeichnis.

Eine IIS-Anwendung erkennt man an dem anderen Symbol im Baum (Abbildung 6.20). Unter *Anwendung verwalten/Erweiterte Einstellungen* (*Manage Application/Advanced Settings*) kann man einen IIS-Anwendungspool zuordnen (siehe nächster Abschnitt).

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

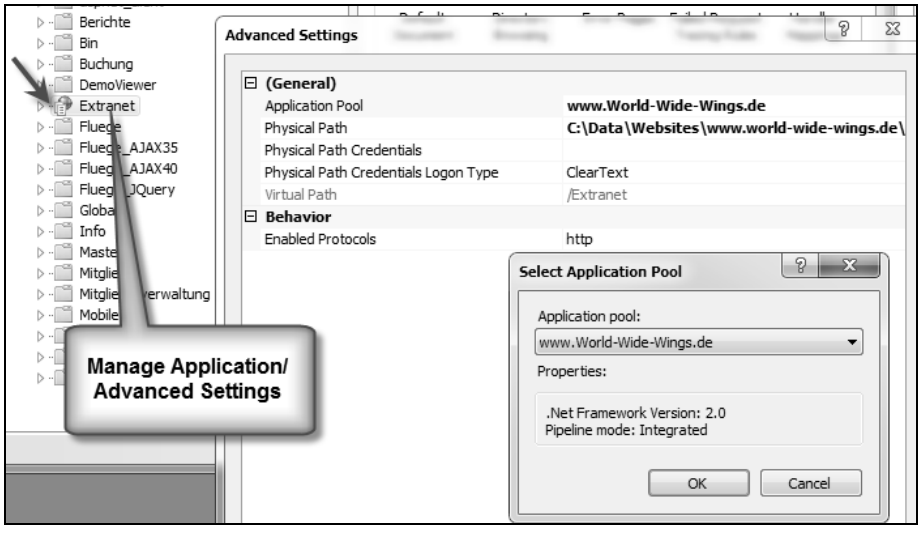

Abbildung 6.20 Einer IIS-Anwendung einen IIS-Anwendungspool zuordnen

IIS-Anwendungspools

Seit der Version 6.0 von IIS lassen sich Webanwendungen durch Anwendungspools isolieren. Ein IIS-Anwendungspool erlaubt verschiedene Einstellungen hinsichtlich Stabilität und Sicherheit.

In IIS 5.x gab es immer nur eine Warteschlange und einen Arbeitsprozess (Worker Process) für alle Webanwendungen auf einem IIS-Server. Ab IIS 6.0 sind nun beliebig viele Warteschlangen und Arbeitsprozesse möglich. Eine Kombination aus Warteschlange und Arbeitsprozess nennt man *Anwendungspool*.

Ein Anwendungspool umfasst eine oder mehrere IIS-Anwendungen. Die im Kernel-Modus arbeitende *http.sys*-Komponente verteilt Anfragen an den Webserver auf die verschiedenen Warteschlangen; die Warteschlangen werden durch Arbeitsprozesse (*w3wp.exe*) abgearbeitet.

Jedem Anwendungspool steht mindestens ein eigener Arbeitsprozess zur Verfügung. Die zur Anfrageverarbeitung notwendigen Erweiterungen (ISAPI-Filter wie ASP und ASP.NET) werden in den Arbeitsprozess geladen.

Der Absturz eines einzelnen Pools hat keine Auswirkung auf die anderen Anwendungspools und den IIS-Webdienst als Ganzes. Das Limit konfigurierbarer Anwendungspools wird im Wesentlichen durch den vorhandenen Speicher gesetzt.

Der so genannte Web Administration Service (WAS) erzeugt und überwacht alle Anwendungspools.

HINWEIS Ein Anwendungspool umfasst eine oder mehrere IIS-Anwendungen. Eine IIS-Anwendung ist dabei ein virtueller Webserver oder ein virtuelles Verzeichnis, das in der IIS-Metabase als *Anwendung* konfiguriert ist. Jeder Anwendungspool hat seine eigene Instanz von *w3wp.exe*. Es kann auch pro Anwendungspool mehr als eine Instanz des Arbeitsprozesses geben. Der IIS-Arbeitsprozess ersetzt den ASP-Arbeitsprozess, den es in IIS 5.0/5.1 gab: Er ist der .NET Runtime Host für ASPX-Seiten. Gleichzeitig beheimatet dieser Prozess aber auch andere dynamische Webtechniken, die innerhalb von IIS-Anwendungen verwendet werden, die zum gleichen Anwendungspool gehören.

Eigenschaften eines Anwendungspools

Zu den zahlreichen Einstellungen für einen IIS-Anwendungspool gehören:

- Das Benutzerkonto (die *Identität*), unter dem der IIS-Arbeitsprozess läuft, kann frei gewählt werden
- Die Anzahl der Arbeitsprozesse pro Anwendungspool kann eingestellt werden
- Die maximale Prozessornutzung eines Arbeitsprozesses kann festgelegt werden
- Das Recycling (Erneuern des Prozesses, das heißt kontrolliertes Beenden und Neustarten; in der deutschen Version von IIS schlicht *Wiederverwendung* genannt) von Prozessen unter bestimmten Bedingungen kann eingestellt werden
- Es kann festgelegt werden, dass IIS den Arbeitsprozess regelmäßig überprüft und recycelt, wenn es zu Problemen kommt. Diese Funktion wird *Pinging* genannt.

Liste der Anwendungspools

Anwendungspools werden im IIS-Manager in dem neuen Zweig *Anwendungspools* (*Application Pools*) verwaltet. Hier sieht man die vorhandenen Pools, ihren Status, die darin verwendete ASP.NET-Version sowie die Prozessidentität (Abbildung 6.21).

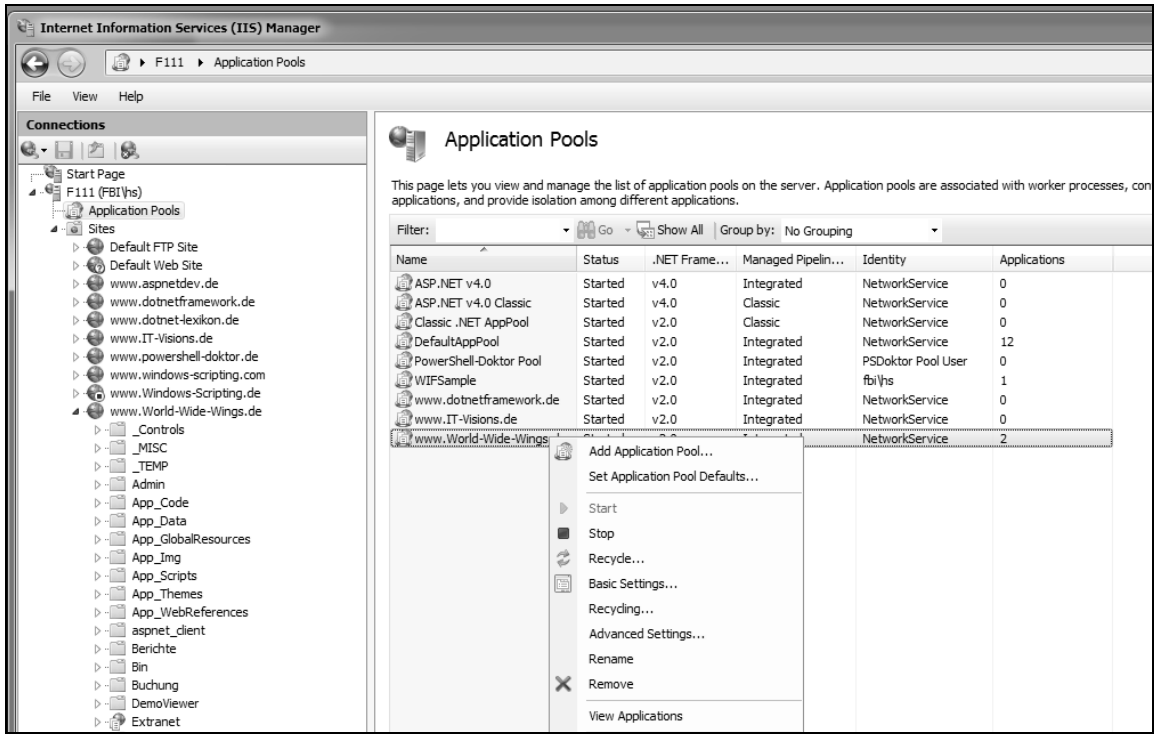

Abbildung 6.21 Anwendungspoolliste in IIS 7.x

Zuordnung von Websites und IIS-Anwendungen zu Anwendungspools

Der IIS-Manager erzeugt jeweils beim Anlegen einer Website auch automatisch einen neuen Anwendungspool unter der Prozessidentität *NetworkService*. Die Zuordnung einer Website zu einem Pool kann man ändern, indem man *Erweiterte Einstellungen/Anwendungspool* (*Advanced Settings/Application Pool*) ändert.

Eine IIS-Anwendung erhält im Standard den gleichen IIS-Anwendungspool wie die übergeordnete Website. Auch dies kann man in den erweiterten Einstellungen einer IIS-Anwendung ändern.

TIPP Eine Website kann zu mehreren verschiedenen Anwendungspools gehören, wenn sie aus mehreren IIS-Anwendungen besteht.

ASP.NET-Version

Für ASP.NET 4.0/4.5 muss man unbedingt unter *Grundeinstellungen* (*Basic Settings*) des Anwendungspools die *.NET Framework-Version* ändern.

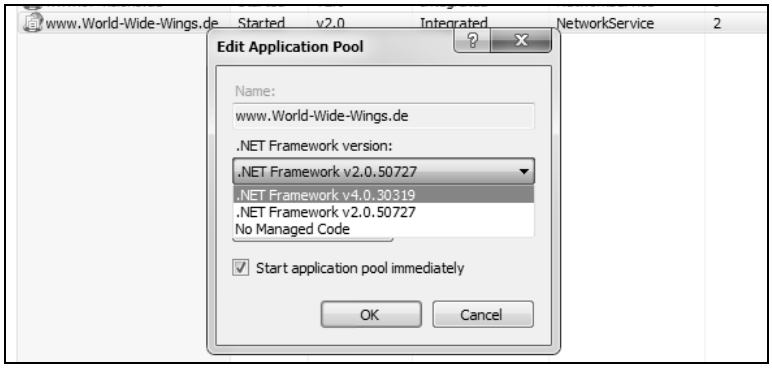

Abbildung 6.22 Ändern der ASP.NET-Version

ACHTUNG Hier verzweifeln viele: In dem Dialogfeld kann man – je nach den installierten .NET Framework-Versionen – nur zwischen ASP.NET 1.0, 1.1 und 2.0 und 4.0 wählen. ASP.NET 3.5 und ASP.NET 4.5 erscheinen dort nie. Ein Unterschied zwischen ASP.NET 2.0 und 3.5 bzw. 4.0 und 4.5 ist auf dieser Ebene auch nicht vorhanden, sodass Microsoft – zur Verwirrung der Anwender – darauf verzichtet (oder vergessen) hat, hier einen neuen Eintrag bereitzustellen bzw. hätte hier Bezug nehmen sollen auf die CLR-Version statt auf die .NET Framework-Version.

ASP.NET 4.0 ist die richtige Auswahl für ASP.NET 4.5.

Erweiterte Einstellungen

Zahlreiche weitere Einstellungen werden unter *Erweiterte Einstellungen* (*Advanced Settings*) vorgenommen (siehe Abbildung 6.23 und folgende Abschnitte).

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

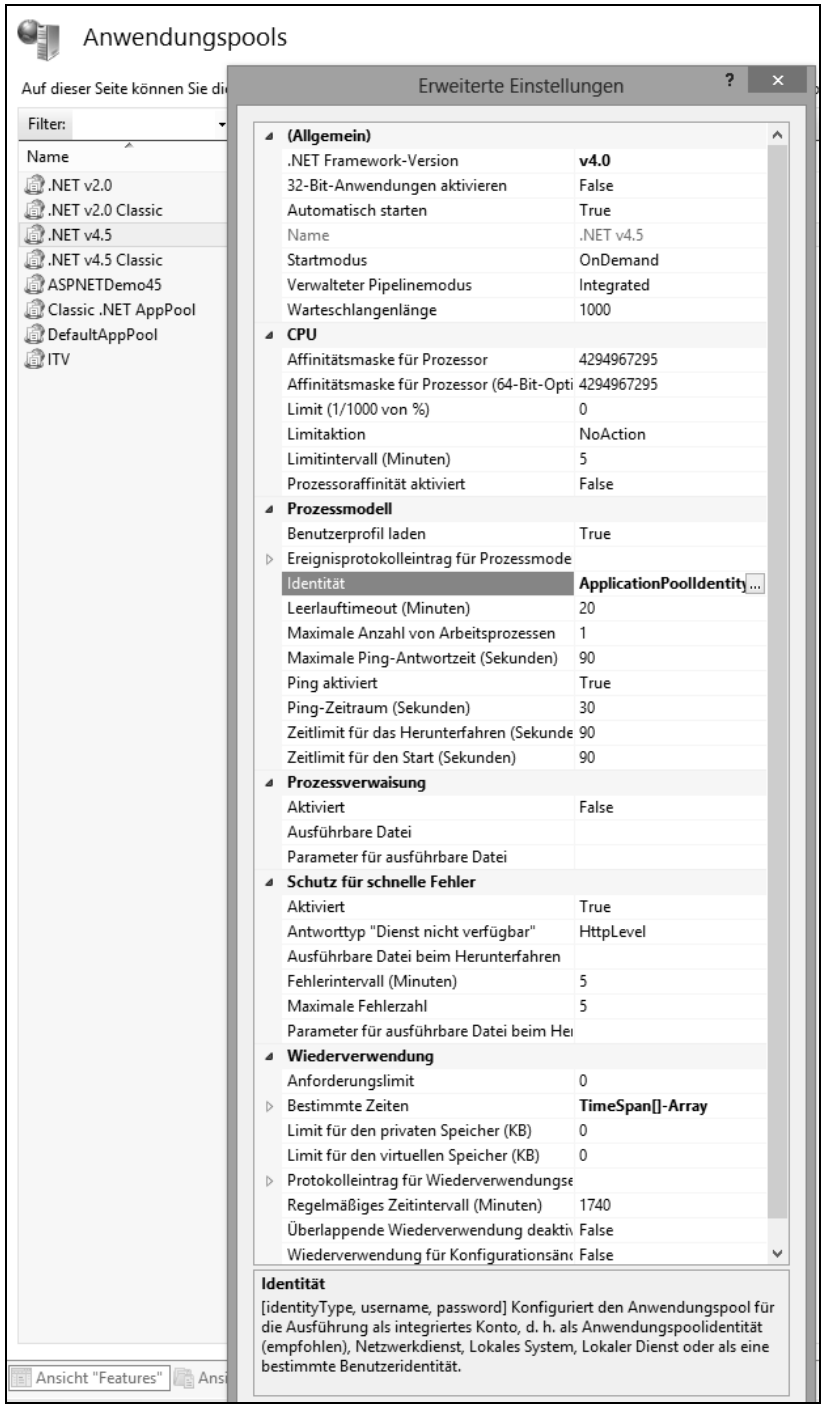

Abbildung 6.23 *Erweiterte Einstellungen* für einen IIS-Anwendungspool

Anwendungspoolidentität

Jedem IIS-Anwendungspool muss ein Benutzerkonto als Anwendungsidentität zugeordnet werden. Der dem Pool zugeordnete Arbeitsprozess (*w3wp.exe*) benutzt diese Identität zur Interaktion mit dem Windows-Betriebssystem und zum Zugriff auf andere Prozesse (z.B. einen Microsoft SQL Server-Datenbankmanagementsystemprozess).

Zur Festlegung der Identität eines Anwendungspools werden vier vordefinierte Konten angeboten:

- **LocalSystem** Alle Rechte auf dem System wie Administratoren
- **NetworkService** Mitglieder der Gruppe *Users*, Zugriff auf Netzwerk (Standard beim Anwendungspool, die durch Anlegen einer Website entstehen)
- **LocalService** Mitglieder der Gruppe *Users*, kein Zugriff auf Netzwerk
- **ApplicationPoolIdentity** Pseudobenutzer mit sehr wenigen Rechten (Standard bei manuell angelegten Anwendungspools)

ACHTUNG Bei der Installation einer Anwendung in IIS ist es regelmäßig die Identitätseinstellung, die Schwierigkeiten bereitet. Ein Entwickler, der seine Anwendung mit ASP.NET Development Server oder IIS Express entwickelt hat, testet unter seinem Benutzerkonto. Wenn die Anwendung dann auf dem IIS-Webserver eingerichtet wird, hat die Anwendung in der Standardkonfiguration möglicherweise weniger Rechte und kann nicht mehr korrekt auf Datenbanken, Dateisystem und andere Ressourcen zugreifen.

TIPP Bei einer Konfiguration der Identität sollte man ein Benutzerkonto auswählen, das im Kontext der Anwendung mit *minimal* möglichen Rechten auf dem System und Netzwerk ausgestattet ist. Wählen Sie folgenden Weg:

- Erstellen Sie ein eigenes Domänen-Benutzerkonto (das lokale Benutzerkonto, wenn kein Zugang zum Netzwerk notwendig ist) für den Anwendungspool
- Vergeben Sie dem Konto explizite Rechte in der Datenbank, im NTFS-Dateisystem und gegebenenfalls anderen Ressourcen (aber so wenig wie möglich!)

HINWEIS Weitere Informationen zum Thema »Identität« erhalten Sie in Kapitel 32 »Sicherheit«.

Wiederverwendung (Recycling)

Eine zentrale Funktion der IIS-Anwendungspools ist, dass IIS eine neue Instanz von *w3wp.exe* starten kann, wenn es zu Problemen bei der Verarbeitung kommt. Diesen Vorgang nennt man Recycling. IIS ersetzt Arbeitsprozesse eines Anwendungspools auf Basis von Konfigurationseinstellungen aufgrund eintretender Ereignisse. Der Ersatz geschieht aus Benutzersicht ohne Unterbrechungen der Webanwendungen, denn beim Recycling eines Arbeitsprozesses wird ein Ersatzprozess gestartet, der mit der Entnahme von Anfragen des Anwendungspools aus der zugehörigen Warteschlange beginnt. Der bestehende Arbeitsprozess stoppt die Annahme weiterer Webanfragen und beendet die in Bearbeitung befindlichen. Die alte Instanz wird nach Beantwortung der letzten Anfrage beendet (dies gilt, solange Sie nicht *Überlappende Wiederverwendung deaktivieren* (*Disable Overlapped Recycle*) manuell von *False* auf *True* stellen!).

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

Recycling kann automatisch erfolgen (siehe unten) oder manuell im IIS-Manager (Funktion *Wiederverwenden* (*Recycle*)) ausgelöst werden. Auch skriptbasiert bzw. über die APIs ist dies möglich.

Das automatische Recycling lässt sich festlegen:

- zu bestimmten Zeiten zwischen 00:00 und 24:00 Uhr (*Specific Times*)
- auf Basis der »Lebenszeit« des Prozesses (*Request Time Interval* – in Minuten)
- auf Basis der Zahl von Anfragen (*Request Limit*)
- auf Basis des Speicherverbrauchs (*Virtual Memory Limit*)

TIPP Das Recycling von Prozessen kann insbesondere Webanwendungen, die Probleme beim Ressourcenmanagement bereiten (z. B. zunehmender Speicherverbrauch), zu höherer Stabilität im Betrieb verhelfen.

Leistungseinstellungen

Mit der Eigenschaft *Prozessmodell/Leerlauftimeout* (*Process Model/Idle Time-out*) eines Anwendungspools wird bei festgestellter Inaktivität eines Arbeitsprozesses über einen festgelegten Zeitraum hinweg dieser gestoppt. Der Zeitraum ist in Minuten konfigurierbar und sollte sich an der Sitzungsdauer von ASP bzw. ASP.NET orientieren, um nicht versehentlich Sitzungen zu beenden.

Um einen weiteren Single Point of Failure zu vermeiden, kann einem Anwendungspool mehr als ein Arbeitsprozess zugeordnet werden (*Maximale Anzahl von Arbeitsprozessen* (*Maximum Worker Process*)). Sollte beispielsweise einer von vier konfigurierten Prozessen abstürzen, stehen dem Pool drei weitere zur Verarbeitung zur Verfügung, womit eine kontinuierliche Serviceleistung möglich ist.

Um eine Überlastung eines Anwendungspools mit Anfragen zu vermeiden, die sich in der zugeordneten Warteschlange ansammeln, kann man die *Warteschlangenlänge* (*Queue Length)* beschränken (gleichnamige Eigenschaft). Beim Überschreiten des Limits beantwortet die *http.sys*-Kernelkomponente die Anfrage mit einem Fehler »503: Service unavailable«.

Für einen Anwendungspool kann auch der maximale Prozessorverbrauch (siehe Einstellungen im Bereich *CPU*) eingestellt werden. Sofern ein Pool diese Beschränkung überschreitet, werden die Arbeitsprozesse bei entsprechender Konfiguration gestoppt und nach einem bestimmten Zeitintervall wieder gestartet. So wird der Prozessorverbrauch durch einzelne Webanwendungen kontrollierbar.

Zustandsüberwachung

Ein weiteres Leistungsmerkmal von IIS ermöglicht die Kontrolle laufender und das Aufspüren von nicht mehr reagierenden Arbeitsprozessen. Bei Aktivierung von *Process Pinging* schickt Web Administration Service (WAS) in definierbaren Abständen eine Kontrollnachricht an den Anwendungspool bzw. den zugeordneten Prozess. Hierbei kann im IIS-Manager das Ping-Intervall festgelegt werden. Wird auf die Nachricht von WAS nicht innerhalb einer konfigurierten Zeit geantwortet, beginnt WAS mit dem Recycling des Arbeitsprozesses.

Sollte das Recycling innerhalb eines einstellbaren Zeitraums wiederholt misslingen bzw. der Arbeitsprozess eines Anwendungspools innerhalb eines einstellbaren Zeitraums wiederholt nicht reagieren, wird der Pool deaktiviert und weitere Anfragen an den Pool von der *http.sys*-Systemkomponente mit einem Fehler »503: Service unavailable« beantwortet.

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

Um blockierenden Prozessen beim Starten oder Stoppen eines Anwendungspools vorzubeugen, ist es möglich, ein Zeitlimit für die Initialisierung bzw. die Beendigung eines Arbeitsprozesses anzugeben. Bei Überschreitung der Beschränkung wird der Prozess durch WAS terminiert und eine Fehlermeldung in das Windows Server 200-Ereignisprotokoll geschrieben.

IIS-Autostart

ASP.NET-Webanwendungen unterliegen verschiedenen Kompilierungsschritten. Sofern nicht mit komplett vorkompilierten Seiten gearbeitet wird, erfolgt ein Teil der Kompilierung immer auf dem Zielsystem beim ersten Aufruf. Selbst im komplett vorkompilierten Fall sind einige Initialisierungsschritte notwendig (z.B. Anlegen des Schattenkopieverzeichnisses). Beim ersten Aufruf einer Website nach einer Aktualisierung fällt daher in der Regel die Performanz schlecht aus. Zudem gibt es Webanwendungen, die größere Datenmengen in einen Cache laden. Auch dies wirkt negativ auf den ersten Benutzer, in diesem Fall den ersten Benutzer nach jedem Neustart des Webserverprozesses.

ASP.NET seit Version 4.0 unterstützt das Aufwärmen von Webanwendungen schon beim Prozessstart des Webservers (Start der w3wp.exe für die Webanwendung). Der Entwickler kann in einem so genannten Preload-Provider (alias: Autostart Provider) Programmcode festlegen, der noch vor dem ersten Besuch während der Initialisierung der Webwendungen (Start der w3wp.exe, auch nach einem Recycling) ausgeführt werden soll.

Es sind folgende Voraussetzungen notwendig:

- IIS ab 7.5 (enthalten in Windows Server 2008 R2 und Windows 7)
- ASP.NET ab 4.0
- Der Anwendungspool in IIS muss auf ASP.NET 4.0 oder 4.5 konfiguriert sein
- Der Anwendungspool in IIS muss auf `startMode="AlwaysRunning"` konfiguriert sein
- Implementierung der Methode `Preload()` in einer Klasse, die die Schnittstelle `System.Web.Hosting.IProcessHostPreloadClient` realisiert
- In IIS muss diese Klasse unter `<serviceAutoStartProviders>` hinterlegt werden
- Die IIS-Website muss mit `serviceAutoStartEnabled="true"` und `serviceAutoStartProvider=`*Name* konfiguriert sein. Dabei muss sich *Name* auf einen der in `<serviceAutoStartProviders>` hinterlegten Namen beziehen. Im World Wide Wings-Beispiel sieht dies so aus: `serviceAutoStartProvider="WorldWideWingsAutoStart"`.

HINWEIS Die IIS-Einstellungen kann man leider nicht alle über den IIS-Manager vornehmen. Man muss die Datei `applicationHost.config` manuell bearbeiten.

```
<system.applicationHost>
<applicationPools>
…
    <add
        name="World Wide Wings"
        managedRuntimeVersion="v4.0"
        startMode="AlwaysRunning"
    />
…
</applicationPools>
```

…

```
<sites>
…
    <site name="World Wide Wings" id="4">
        <application path="/" applicationPool="ASPNEt4ODemos"
            serviceAutoStartEnabled="true"
            serviceAutoStartProvider="WorldWideWingsAutoStart">
                <virtualDirectory path="/"
                physicalPath="H:\TFS\Demo\NET4\ASPNET4Demos\ASPNET4Demos" />
        </application>
        <bindings>
            <binding protocol="http" bindingInformation="*:111:" />
        </bindings>
    </site>
…

<serviceAutoStartProviders>
    <add name="WorldWideWingsAutoStart"
        type="ASPNET4Demos_LIB.WorldWideWingsAutoStart, ASPNET4Demos_LIB" />
</serviceAutoStartProviders>
```

Listing 6.3 Einträge in der Datei *applicationHost.config* von IIS

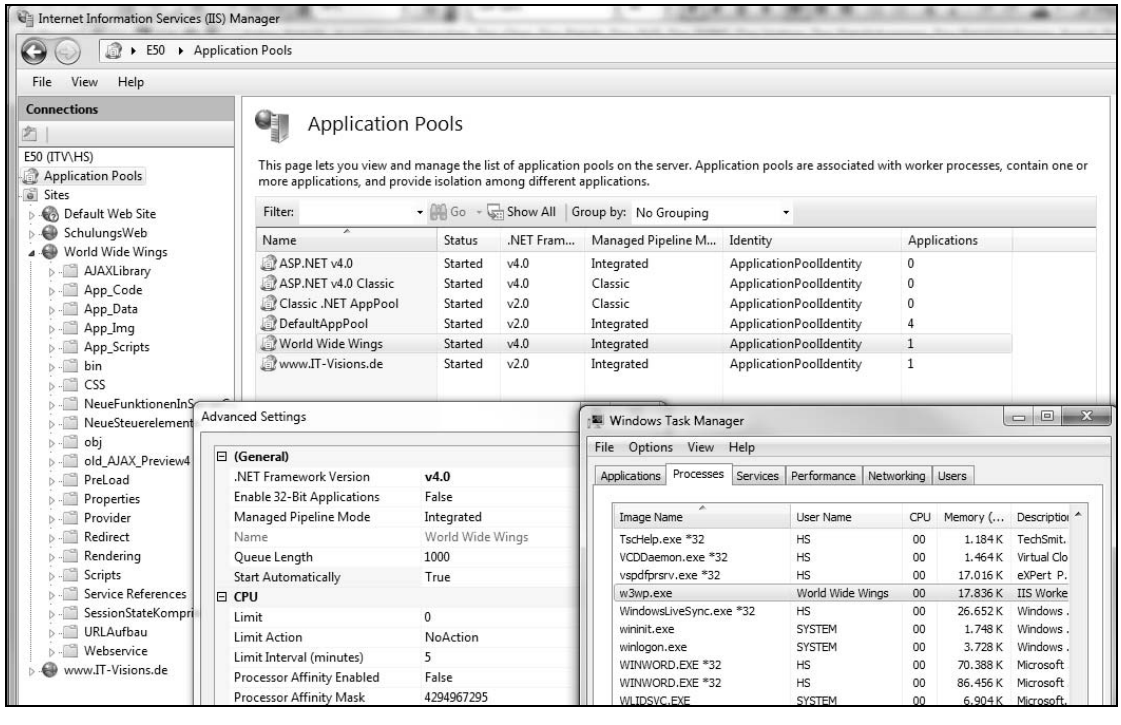

Abbildung 6.24 Eine ASP.NET-Website wird noch vor dem ersten Besucher gestartet. Man sieht deutlich den Webserverprozess (*w3wp.exe*) im Windows Task-Manager

TIPP In einem solchen Preload-Provider (alias Autostart-Provider) könnte z. B. die zeitaufwändige Initialisierung eines größeren ADO.NET Entity Framework-Modells stehen. Die lästige Zusatzwartezeit für den ersten Benutzer verlagert sich damit auf den Bootvorgang des Servers.

Das folgende Listing zeigt eine Implementierung eines Preload-Providers, der ein Entity Framework-Modell initialisiert und eine erste Speicheraktion ausführt; dies führt dazu, dass alle folgenden Operationen schneller sind. Zudem erzeugt der Preload-Provider einen Protokolleintrag im Dateisystem und in der Datenbank.

```
using System;

namespace ASPNET4Demos_LIB
{  /// <summary>
  ///   ASP.NET 4.0 AutoStart-Provider (alias Preload-Provider)
  /// </summary>
  public class WorldWideWingsAutoStart : System.Web.Hosting.IProcessHostPreloadClient
  {
    public void Preload(string[] parameters)
    {
      // Protokollierung
      System.IO.StreamWriter sw =
          new System.IO.StreamWriter(@"c:\temp\autostartlog.txt",
          true);
      sw.WriteLine("Autostart beginning: " + DateTime.Now);
      if (parameters != null) sw.WriteLine(String.Join(";", parameters));

      // Erste Instanz des EF-Modells
      ASPNET4Demos_LIB.WWWings6Entities modell =
        new ASPNET4Demos_LIB.WWWings6Entities();
      // Erste Speicheroperation

      var p = new Protokoll();
      p.Zeit = DateTime.Now;
      p.Text = "ASP.NET Webapp Preload";
      modell.AddToProtokoll(p);
      modell.SaveChanges();

      // Protokollierung
      sw.WriteLine("Autostart completed: " + DateTime.Now);
      sw.Close();

    }
  }
}
```

Listing 6.4 Beispiel eines `Preload`-Providers

ACHTUNG Ein Fehler in der `Preload()`-Routine sorgt dafür, dass der IIS Application Pool anhält: »There was an error during processing of the managed application service auto-start«. Den Fehler findet man dann im Windows-Ereignisprotokoll (Rubrik *Application*).

IIS-Verarbeitungspipeline

Neben der Hochzeit auf Konfigurationsebene haben IIS und ASP.NET auch auf der Ebene der Verarbeitung eines Seitenabrufs über die Request Pipeline zueinander gefunden. In den früheren IIS-Versionen kümmerte sich zunächst IIS um die Anfrage und übergab sie dann an *aspnet_isapi.dll*; die von ASP.NET erzeugte Antwort ging wieder zurück an den IIS-Webserver, der sie an den Client gesendet hat. In dem neuen Integrated Application Pool Mode lassen sich die Module in beliebiger Reihenfolge nacheinander ausführen, unabhängig davon, ob es sich um Module in verwaltetem Code (z.B. *ASP.NET-HTTP-Handler* und *ASP.NET-HTTP-Module*) oder in nicht verwaltetem Code wie bei den Win32-ISAPI-Modulen handelt. Bisherige Doppelarbeiten, z.B. im Bereich der Authentifizierung für den Webserver und für ASP.NET, entfallen dadurch.

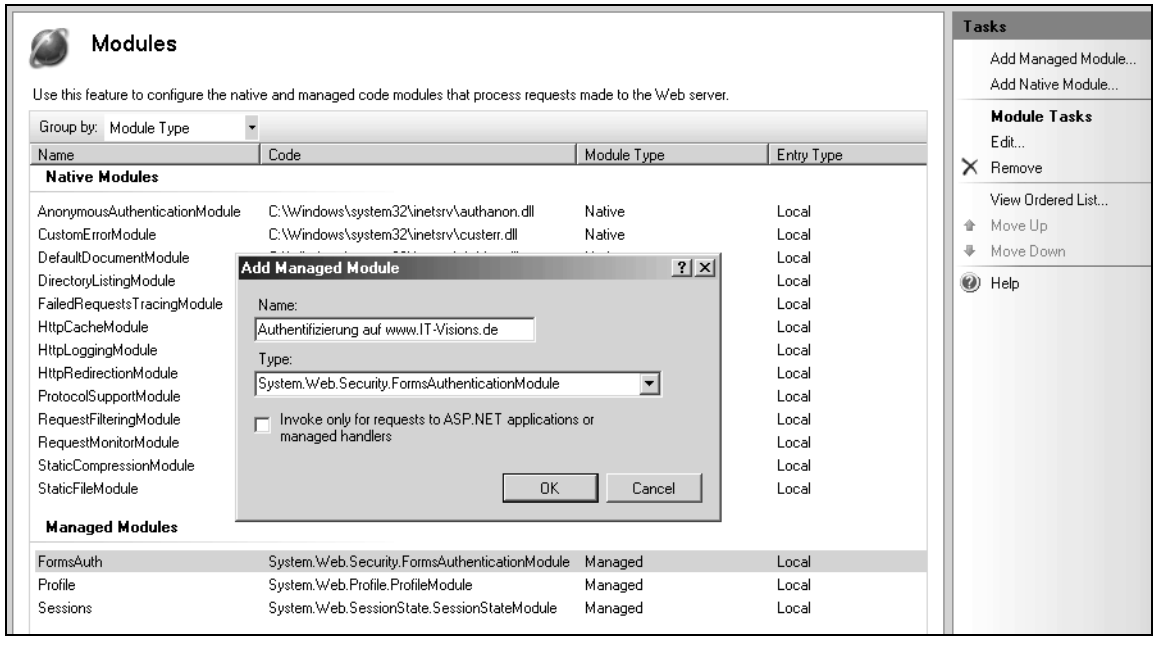

Abbildung 6.25 Aktivierung einzelner Module auf Siteebene

Auch die Administration wird einfacher, da sie nicht mehr zwischen IIS- und ASP.NET-Modulen unterscheidet. IIS verwaltet alle Module im Element `<modules>` in den Konfigurationsdateien. In dem neuen Modell sind die aus ASP.NET stammenden Konfigurationselemente `<httpModules>` und `<httpHandlers>` ohne Bedeutung. Daraus ergibt sich, was Microsoft *Unified Request Pipeline* nennt: eine gemeinsame Aufrufkette für die beiden Modultypen. Um die Kompatibilität zu wahren, lässt sich IIS ab Version 7.x auch in den *ISAPI Application Pool Mode* schalten. Dann funktioniert die Pipeline wieder wie in IIS 6.0.

Ein weiterer Vorteil des neuen Verarbeitungsmodells ist, dass für die Erweiterung von IIS keine ISAPI-Erweiterungen – die nur in C++ entwickelt werden können – mehr notwendig sind. IIS ab Version 7.x erlaubt die Erweiterung mithilfe der aus ASP.NET bekannten Programmiermodelle Managed ASP.NET-Handler und Managed ASP.NET-Modul; dabei kann eine beliebige .NET-Sprache zum Einsatz kommen. Dadurch wird es beispielsweise möglich, die Protokollierung oder die vom Webserver erzeugte Verzeichnisauflistung den eigenen Bedürfnissen anzupassen. Microsoft will damit den Webserver auch für Erweiterungen von Drittanbietern öffnen.

Internet Information Services (IIS) Express

Visual Studio 2012 verwendet zum Ausführen und Debugging von Webprojekten nun bei den mitgelieferten Projektvorlagen im Standard »IIS Express 8.0« anstelle des »Visual Studio Web Development Server« (alias »ASP.NET Web Development Server« – es werden verschiedene Bezeichnungen in der Oberfläche verwendet).

»IIS Express« ist eine leichtgewichtige Variante von »Internet Information Services« (IIS). Dabei hat IIS Express die gleichen Funktionen, läuft aber nicht als Windows-Systemdienst, sondern als einzeln startbare Anwendung (iiexpress.exe). Microsoft will damit erreichen, dass Entwickler nicht mehr so oft in die Falle laufen, dass eine Webanwendung in der Entwicklungsumgebung sich anders verhält als auf dem Zielsystem.

HINWEIS Während der große Bruder IIS 8.0 nur auf Windows 8 und Windows Server 2012 läuft, kann man IIS Express 8.0 auch auf Windows Vista, Windows 7, Windows Server 2008 und 2008 R2 verwenden. Allerdings sind dann einige Funktionen (Websockets, Zentraler Zertifikatsspeicher, Server Name Indication (SNI)) nicht verfügbar.

Visual Studio startet IIS Express bei Bedarf. IIS Express ist genauso wie »Visual Studio Web Development Server« über ein Symbol in der Taskleiste beendbar (Abbildung 6.26). Einstellungen für IIS Express nimmt man in den Eigenschaften der Website vor (siehe ebenfalls Abbildung 6.26).

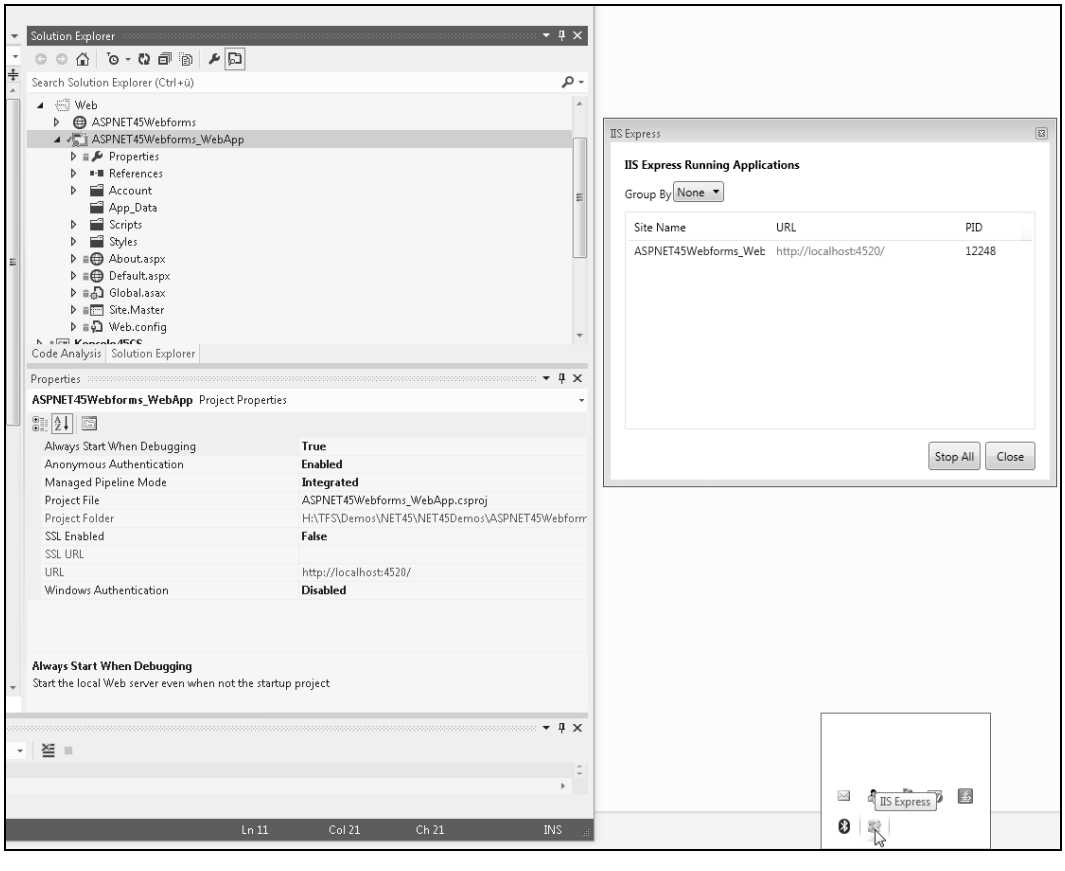

Abbildung 6.26 Fenster von IIS Express, das über ein Symbol in der Windows-Taskleiste aufrufbar ist

Konfiguration des Webservers in der Entwicklungsumgebung

Welchen Webserver die Entwicklungsumgebung Visual Studio beim Start des Debuggings einer Webanwendung verwendet, legt man bei Webprojekten abhängig vom verwendeten Projektmodell fest. Beim …

- *Webanwendungsmodell* (WAP) über die Eigenschaften des Projekts auf der Registerkarte *Web*
- *Websitemodell* (WSP) über die Eigenschaftenseiten der Website (auf der Position des Stammverzeichnisses) auf der Registerkarte *Startoptionen*

TIPP Eine bestimmte *Startseite* (*Specific page*) kann man als Alternative zur Registerkarte *Web* bzw. zu den Eigenschaftenseiten auch sehr einfach über das Kontextmenü einer Seite im Projektmappen-Explorer festlegen. Hierzu steht der Eintrag *Als Startseite festlegen* (*Set as Start Page*) zur Verfügung.

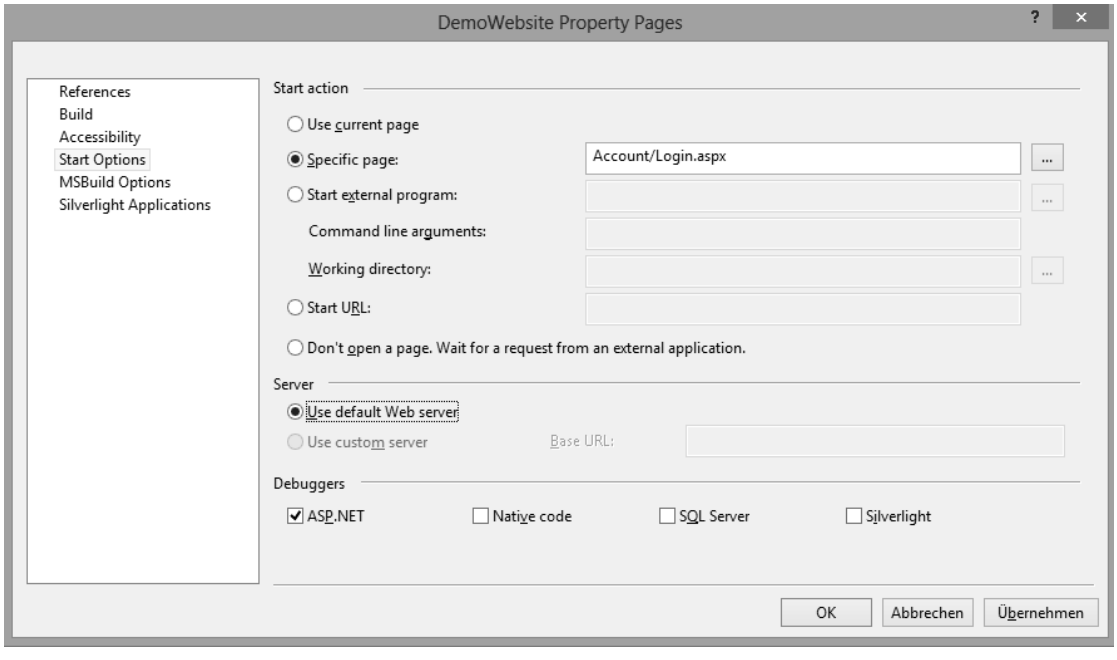

Abbildung 6.27 Festlegung der Startoptionen für eine Website (WSP) in Visual Studio

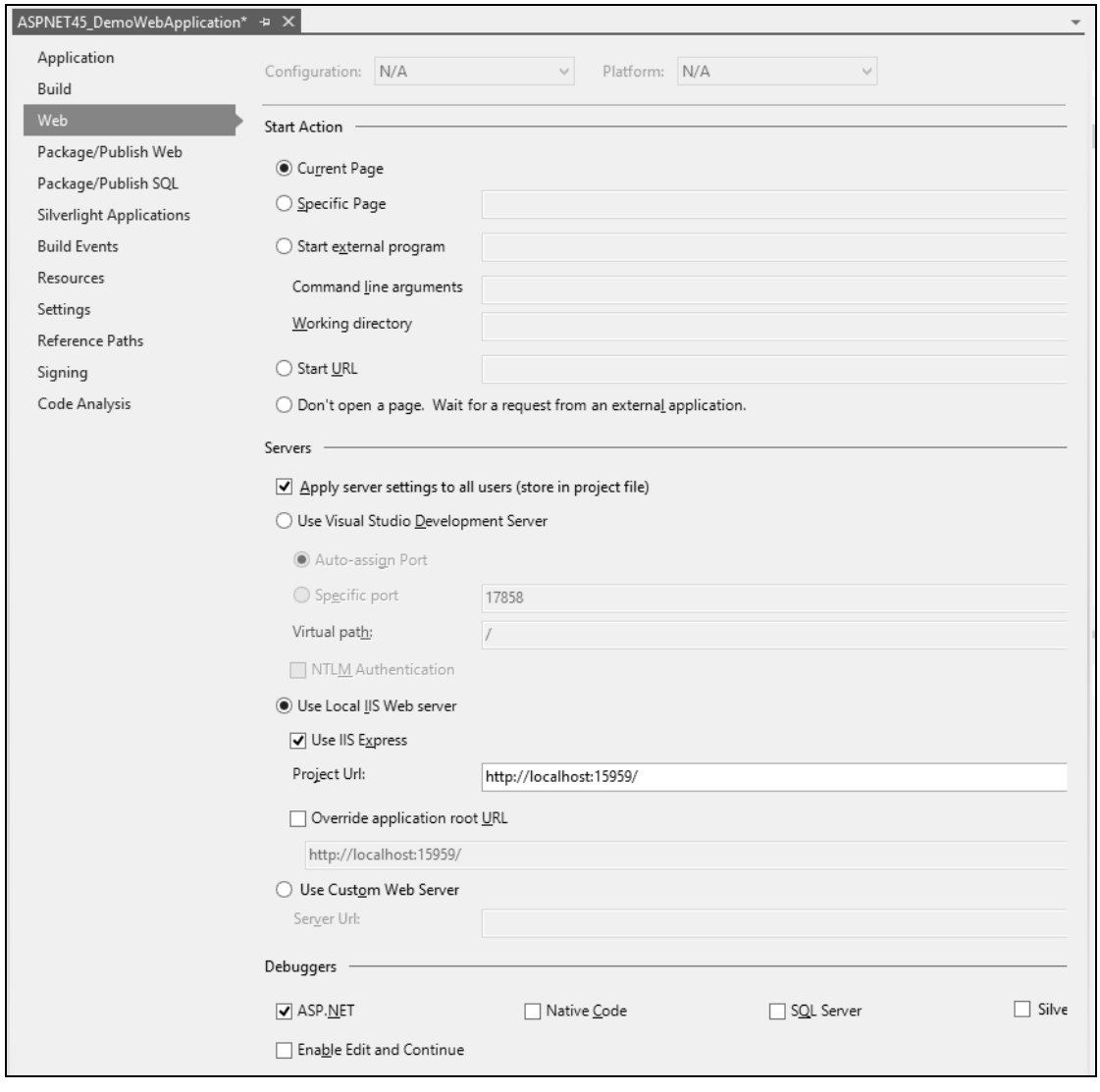

Abbildung 6.28 Festlegung der Startoptionen für eine Webanwendung (WAP) in Visual Studio

Für Webanwendungsprojekte kann man den Wechsel zwischen IIS Express oder ASP.NET Development Server über die Projekteigenschaften steuern. Für Website-Projekte, die keine Projektdatei besitzen, liegt diese Einstellung in der *.sln*-Datei. Hier muss man bei einem Wechsel den Eintrag `UseIISExpress = "true"` hinzufügen bzw. entfernen. Visual Studio unterstützt dies durch die Funktionen *IIS Express verwenden* (*Use IIS Express*) oder *Visual Studio Development Server verwenden* (*Use Visual Studio Development Server*) im Kontextmenü des Stammverzeichnisses einer Website.

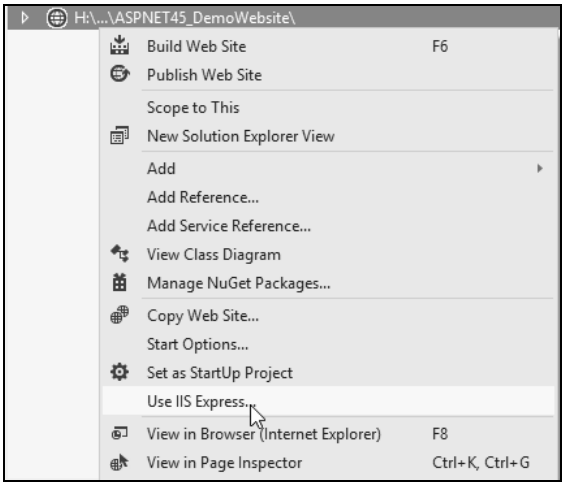

Abbildung 6.29 Umschalten auf IIS Express im Kontextmenü einer Website

TIPP Ob im Standard für neue Webprojekte der Webserver IIS Express oder aber ASP.NET Development Server verwendet wird, steuert eine Einstellung unter *Tools/Optionen/Projekte und Projektmappen/Webprojekte* (*Tools/Options/Projects and Solutions/Web Projects*).

Apache-Webserver

Für die Integration von ASP.NET in den Apache-Webserver gibt es verschiedene Varianten:

- Verwendung des in Mono enthaltenen Apache-Moduls (*mod_mono*) [MONO04]
- Nutzung von Cassini zusammen mit Apache [CPR01]
- Nutzung des kommerziellen *mod_asp.net* der Firma Covalent [COV01]

Webhosting

Dieser Abschnitt gibt einen kurzen Überblick über den Webhostingmarkt für ASP.NET.

Formen des Webhosting

Beim *gemeinsamen Hosting (Shared Hosting)* teilen sich zahlreiche Kunden ein physisches System. Der Webserver ist hier ein virtueller Webserver, die Datenbank nur eine Datei im gemeinsamen Datenbankserverprozess. Die Möglichkeiten des Kunden sind auf das Einspielen von Skripts, Daten und – vielleicht – Programmbibliotheken beschränkt. Den Rest kann nur der Provider erledigen.

Wer auf seinem System »alles« machen will, z.B. außer HTTP und FTP andere Serverdienste anbieten und dafür eigene Software installieren, benötigt den *root*-Zugriff. Der aus der Linux-Welt stammende Begriff hat sich auch beim Windows-basierten Hosting für eine Zugriffsform etabliert, bei welcher der Kunde die Administrator-Rolle über das System erhält. Manche Anbieter sprechen auch vom *dedizierten Server* oder *gemieteten Server*.

Root Hosting ist grundsätzlich in zwei Formen möglich. Entweder steckt der Provider für jeden Kunden ein eigenes Server-Blade in den Schrank oder er virtualisiert das Betriebssystem mit Lösungen wie VMWare oder Microsoft Virtual Server.

Verwendete Software

Als Betriebssystem kommt bei den Windows-basierten Webhostingangeboten Windows Server 2012 oder 2008 in der Standard- oder der günstigeren Web-Edition vor. Bei den Webservern ist die Sache klar: Apache ist bei allen Linux-Angeboten installiert, während in der Windows-Welt Microsoft IIS zum Lieferumfang des Betriebssystems gehört. Bei den Datenbanken dominiert MySQL, denn auch ein beträchtlicher Teil der Windows-Hoster liefert den freien Datenbankserver mit. Beim Hosting von Microsoft SQL Server hat man die Wahl zwischen dem kostenfreien Microsoft SQL Server Express oder dem kommerziellen SQL Server 2008/2012.

Zur Verbindung von Webserver und Datenbank braucht man eine Skript- oder Programmiersprache. Bei den Linux-Angeboten ist immer PHP vorinstalliert, manchmal auch Python, Perl und Java Server Pages/Java-Servlets sowie die für JEE-Webanwendungen notwendigen Web- und Anwendungscontainer. In der Windows-Welt gehören Active Server Pages (ASP) und ASP.NET immer dazu, denn diese (sprachneutralen) Programmierframeworks sind Bestandteil des IIS-Webservers.

Konfiguration

Fast immer steht dem Kunden eine Weboberfläche zur Konfiguration zur Verfügung. Beim Shared Hosting ist dies oft die einzige Zugangsmöglichkeit, z.B. um Rechte auf einem Verzeichnis so zu setzen, dass die Webanwendung dort Dateien anlegen darf. Bei Anbietern, die Frontpage Server Extensions unterstützen, kann man auch mit Microsoft Frontpage einige Einstellungen vornehmen.

Alle Anbieter von dedizierten Servern erlauben nicht nur das Einspielen von Skripts, sondern auch die Installation eigener Software. Auch wenn der Kunde damit Lücken in der vorinstallierten Software ausbessern kann, ist die Bedeutung des mitgelieferten Softwarepakets nicht zu unterschätzen, denn das spart Konfigurationsaufwand und in vielen Fällen auch Lizenzkosten. Bei dedizierten Servern gehört auch die freie Entscheidung, wann und wie oft der Kunde das System neu starten will, zum Standard. Die Datensicherungslösung ist hingegen uneinheitlich. Einige Provider übernehmen Datensicherung und Datenwiederherstellung, andere stellen bei der vorinstallierten Software eine Backupanwendung zur Verfügung, die der Kunde selbst steuern kann. Interessant ist, dass einige Anbieter ein Backup gar nicht vorsehen. Das Einspielen von Updates und Patches gestatten die meisten Anbieter dem Kunden.

Da der Kunde keinen physischen Zugang zu der gemieteten Hardware hat, muss man ihm entfernte Verwaltungsoptionen bieten. In der Linux-Welt ist der ssh-Zugang Standard. Viele Anbieter gestatten auch FTP. In der Windows-Welt ist ssh selten, denn Microsoft hat hier andere Lösungen. Das zentrale Verwaltungsprotokoll von Windows ist Remote Procedure Call (RPC). Mit RPC greifen die Windows-Kommandozeilen-

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)

werkzeuge und grafische Werkzeuge wie die Microsoft Management Console (MMC) auf die Systemdienste des Betriebssystems zu. RPC kann auch über das Internet genutzt werden, um mit den Windows-eigenen Werkzeugen ein entferntes System zu verwalten. Voraussetzung ist jedoch eine nicht unerhebliche Anzahl offener Ports. Nur etwa die Hälfte aller Provider lässt diesen Zugangsweg zu, denn in der Vergangenheit nutzten viele Würmer Lücken in RPC aus. Aus Kundensicht ist ein RPC-Zugang hilfreich, denn man kann durch individuell zusammengestellte MMC-Konsolen ganze Serverfarmen über eine Bildschirmmaske verwalten.

Fast alle Anbieter von dediziertem Hosting gestatten hingegen Windows Terminal Services, mit denen der Kunde den Bildschirm des Servers auf einen entfernten PC projizieren kann. Windows Terminal Services basieren auf Remote Desktop Protocol (RDP) und fordern einen Zugang zu Port 3389. Im Vergleich zum RPC-Protokoll hat man mit einem RDP-Zugang noch mehr Möglichkeiten; die Verwaltung von Server-Farmen fällt allerdings schwerer, denn man hat für jeden zu verwaltenden Server ein separates Terminal-Fenster auf dem Bildschirm.

Für den Fall, dass ein Kunde sich von allen Zugängen ausgesperrt hat (z.B. durch falsche Einstellungen der Windows Firewall), ist eine administrative Weboberfläche auf einem Webserver des Providers sehr sinnvoll. Man könnte vermuten, dass eine solche Webschnittstelle zur Verwaltung des Systems (z.B. zum Neustart) zum Standard gehört. Eine Erhebung [WMHS01] zeigt aber, dass bei Weitem nicht alle Anbieter diese zur Verfügung stellen.

Preise

Obwohl ASP.NET gegenüber PHP Vorteile in der Entwicklungsproduktivität besitzt (siehe Vergleich in [JK01]), hat Microsoft es schwer, WIMA (Windows, Internet Information Services, Microsoft SQL Server/MSDE, ASP.NET) gegen LAMP (Linux, Apache, MySQL, PHP) im Hostingmarkt durchzusetzen. Die Gründe dafür liegen nicht nur in dem lange Zeit als *Käferfänger* bekannten Redmonder Webserver IIS, sondern auch darin, dass bei WIMA zumindest das Betriebssystem Anschaffungskosten verursacht, während LAMP völlig »frei« ist. Bei Webhostingprovidern galt und gilt die Windows-Plattform zudem als schwerer automatisierbar. Folglich waren Windows-basierte Hostingangebote immer deutlich teurer als die Linux-Alternative.

Mittlerweise bemüht sich Microsoft jedoch um günstigere Hostingangebote. Unter [MS12] bietet Microsoft den Hostern ein Rundumpaket mit Skripts und Verwaltungsanwendungen. In Deutschland existieren inzwischen Angebote für Hosting auf virtuellen Webservern für rund 10 Euro im Monat [MS11]. International gibt es noch günstigere Angebote (ab 2 Euro im Monat) [MS03].

HINWEIS Die Website zu diesem Buch (*www.aspnetdev.de*) wird bei der Firma QualityHosting (*http://www.qualityhosting.de*) betrieben und von dieser gesponsert.

Dr. Holger Schwichtenberg: Microsoft ASP.NET 4.5 mit Visual C# 2012 - Das Entwicklerbuch,
Microsoft Press 2013 (ISBN 978-3-86645-570-2)