

ClickOnce unter .NET Framework 2.0 und Visual Studio 2005

# Einmal geklickt, schon installiert

Visual Studio und .NET Framework 2.0 bescheren dem Entwickler eine neue Möglichkeit, seine Programme an den Mann, den Anwender zu bringen. Mit ClickOnce lassen sie sich auf einem Server bereitstellen und vom Nutzer installieren, wie es jetzt schon mit Windows-Updates möglich ist.

**W**arum werden heute Geschäfts-anwendungen meist als Webanwendungen unter ASP.NET implementiert? Ein großer Vorteil dürfte das Installationsmodell sein, da hierbei Anwendungs-Updates zentral auf einem Webserver bereitgestellt werden können und die vielen Clients sofort mit der aktualisierten Version arbeiten können.

Dabei ist das Entwickeln von Webanwendungen gar nicht so komfortabel, gemessen an den Vorteilen einer leistungsfähigen Windows-Forms-Anwendung. Webanwendungen basieren auf dem HTTP-Protokoll, das von Natur aus statuslos ist. Daher muss sich der Programmierer hier

immer um die Statusverwaltung kümmern. Zudem muss dieses Programmiermodell das Request-/Response-Prinzip des Internets berücksichtigen, das ebenfalls längst nicht so bequem ist wie die Event- und Nachrichtenmechanismen von Windows Forms.

Trotzdem werden viele Geschäftsanwendungen als Webanwendungen entwickelt. Der Grund liegt in einem entscheidenden Vorteil der Webanwendungen: dem einfachen Bereitstellen (Deployment). Denn Webanwendungen laufen immer auf einem zentralen Webserver. Dieser – im Fall von ASP.NET ist das IIS – stellt die nötige Technik für die Funktionalität der Anwendung bereit.

Der Client dagegen, also der Webbrowser, fordert eine Seite vom Webserver an und sorgt im Allgemeinen lediglich dafür, dass diese lokal dargestellt wird. Solange nicht spezielle Browser-Plug-Ins für die Anwendung nötig sind, ist es völlig egal, welche Software-Komponenten auf dem Client installiert sind, denn das eigentliche lokale Betriebssystem für die Webanwendung ist der Internet Explorer.

Das ist sehr bequem, denn bei der Entwicklung muss sich der Programmierer nicht um Details des Clients kümmern. Er muss nur wissen, welche Version des Internet Explorers installiert ist, und alles andere erledigt ASP.NET.

Fehlfunktionen aufgrund falscher Software-Komponenten kommen praktisch nicht vor. Bei einem Update der Anwendung muss einfach nur die Programmversion auf dem Webserver ersetzt werden. Beim nächsten Zugriff auf die Webanwendung nutzt der Browser automatisch die aktualisierte Version. Dadurch lassen sich Hunderttausende Clients innerhalb von Sekunden aktuali-

sieren. Diese auf jedem PC einzeln „auf Stand“ zu bringen, wie es bei Windows-Forms-Anwendungen geschieht, würde einen nicht akzeptablen Zeitaufwand erfordern.

So gesehen haben Webanwendungen eigentlich nur Vorteile. Der Teufel liegt jedoch im Detail. Zwar lässt sich jede erdenkliche Anwendung prinzipiell als Webanwendung implementieren. Die Frage ist nur: mit welchem Entwicklungsaufwand und mit welchen Abstrichen beim Benutzerkomfort? Das HTTP und das Request-/Response-Modell des Internets machen das Entwickeln deutlich komplizierter und zeitaufwändiger.

Wie sieht es aber mit der Benutzerfreundlichkeit einer Webanwendung aus? Mit viel Aufwand und JavaScript bekommen Sie alles so hin, wie Sie es von Windows-Anwendungen her gewohnt sind. Aber wer will schon Hunderte von Stunden in JavaScript-Entwicklungen investieren, wenn er das Ganze unter Windows Forms ohne Entwicklungsaufwand zum Nulltarif geboten bekommt?

Microsoft hat im Lauf der Zeit dieses Problem erkannt und sich dem Deployment von Windows-Anwendungen gewidmet. Bisheriges Ergebnis ist unter Visual Studio 2003 das so genannte No-Touch-Deployment, mit dessen Hilfe sich bereits ein vereinfachtes, zentrales Bereitstellen von Windows-Anwendungen erledigen lässt. Allerdings ist dieser Mechanismus noch nicht ganz ausgereift und außerdem anscheinend in Vergessenheit geraten.

Aber Tote leben länger. Unter Visual Studio 2005 erhebt diese Technologie unter dem Namen ClickOnce wieder auf, um zahlreiche Features verbessert und nun tatsächlich in der Lage, Windows-

## Auf einen Blick

### Autor

#### Klaus Aschenbrenner

arbeitet als Software Architect und Consultant bei der Firma TechTalk in Wien. Er beschäftigt sich bereits seit mehreren Jahren mit der Windows-Programmierung und seit der Beta 1 Version mit dem .NET Framework. Nähere Informationen zu seiner Person finden Sie auf seiner Homepage [www.csharp.at](http://www.csharp.at).



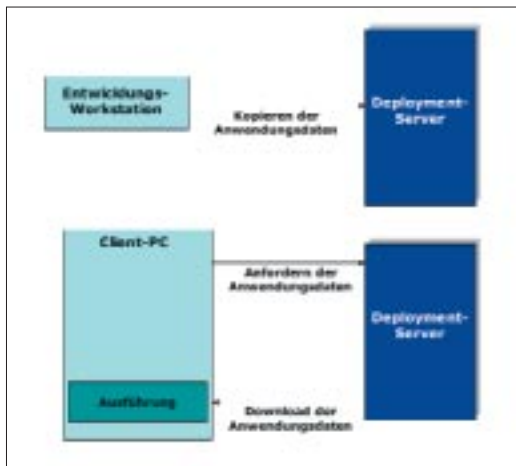
dotnetpro.code  
A0412ClickOnce



Sprachen VS.NET

Technik .NET Framework 2.0

Voraussetzungen Visual Studio 2005  
Beta 1



**Abbildung 1** Die grundlegenden Schritte beim Bereitstellen einer ClickOnce-Anwendung.

Anwendungen zentral zu verteilen und zu aktualisieren.

## Überblick über ClickOnce

ClickOnce ist Teil des .NET Frameworks 2.0 und kann Windows-basierte Anwendungen auf dem Desktop des Benutzers bereitstellen, indem die Anwendungsdaten auf einem Server (auch Deployment-Server genannt) zur Verfügung gestellt werden. Dem Benutzer wird dann ein Link zu den Programmdateien zur Verfügung gestellt. Öffnet der Benutzer diesen Link, werden die Programmdateien auf den Client-PC heruntergeladen und ausgeführt. Das Programm wird dabei in einer so genannten „Secure Sandbox“ ausgeführt, die von .NET durch CAS (Code Access Security) bereitgestellt wird. Dabei handelt es sich um eine Anwendungsdomäne, die nur wenige und außerdem eingeschränkte Rechte für die Anwendung zur Verfügung stellt.

Steht eine neue Programmversion auf dem Server bereit, können die Clients dies automatisch feststellen. Die neue Programmversion lässt sich dann im Hintergrund herunterladen und installieren. Dabei gibt es eine Reihe von Konfigurationsmöglichkeiten, um die Funktionsweise des Update-Vorgangs zu beeinflussen. ClickOnce bietet außerdem die Möglichkeit an, Anwendungen so einzurichten, dass der Nutzer auch offline mit ihnen arbeiten kann, sie also ohne Verbindung zum Netzwerk zur Verfügung stehen. Dazu wird zum Beispiel ein Programmpunkt im Startmenü angelegt, der die Offline-Variante der Anwendung startet.

ClickOnce basiert auf einer Reihe von Technologien, die bereits im .NET Framework vorhanden sind. Longhorn, die kommende Windows-Version, soll ClickOnce um weitere Features bereichern und eine nahtlose Integration mit dem Desktop ermöglichen.

Ein Design-Ziel von ClickOnce ist ein vertrauenswürdiges Modell, damit der Benutzer Windows-Anwendungen von einem zentralen Server herunterladen und lokal ausführen kann, ohne dazu Administratorberechtigungen zu benötigen. Die Anwendun-

gen werden dabei in einem privaten Bereich des aktuellen Benutzerprofils installiert – der schon erwähnten Secure Sandbox –, damit es nicht zu Problemen mit anderen installierten Programmen kommt.

Das erstmalige Einrichten einer ClickOnce-Anwendungen ist einfach und schnell erledigt. Wenn aber eine schon eingerichtete Anwendung aktualisiert werden muss, stehen eine Reihe von Optionen bereit, mit deren Hilfe der Administrator den Server konfigurieren und den Vorgang zentral steuern kann.

Abbildung 1 zeigt den grundlegenden Ablauf beim Einrichten einer ClickOnce-Anwendung beim Benutzer. Nachdem die Anwendung entwickelt und getestet wurde, kann sie auf einem Server zur Verfügung gestellt werden. Dazu werden die Anwendung, das so genannte Deployment-Manifest und das Application-Manifest auf den Server kopiert. Bei den Manifestdateien handelt es sich um XML-Konfigurationsdateien, die die Installation der Anwendung steuern. Das

Application-Manifest wird vom Entwickler der Anwendung erstellt und beschreibt, aus welchen Dateien diese besteht, welche Abhängigkeiten vorliegen und welche Security-Einstellungen getroffen sind. Dies wird automatisiert durch Visual Studio 2005 erledigt.

Das Deployment-Manifest wird meistens durch den Administrator erstellt und gewartet, der für das Einrichten und Updaten der Anwendungen zuständig ist. Die Datei beschreibt, wie die Anwendung auf die Clients installiert wird und welche Update-Optionen zur Verfügung stehen. Auch die Manifestdatei kann von Visual Studio 2005 automatisch erstellt werden.

Nachdem Anwendung und Manifestdateien auf den Server kopiert wurden, kann den Anwendern ein URL mitgeteilt werden, mit dessen Hilfe sie die Anwendung lokal installieren können. Wenn der Benutzer auf den Link klickt, erkennt die lokale .NET Runtime die Dateinamenerweiterung und führt ein ClickOnce-Deployment aus. Darunter fallen der Download und das Ausführen der Anwendung. Damit wäre auch schon die einzige Anforderung genannt, die ClickOnce benötigt: das .NET Framework 2.0 muss auf dem Client installiert sein. Dieses wiederum lässt sich über den System Management Server (SMS) firmenweit auf den Maschinen bereitstellen.

Die Anwendungsdateien, die vom Client heruntergeladen werden, werden unter einem verschlüsselten Pfad im Benutzerprofil des aktuell angemeldeten Benutzers gespeichert (*C:\Documents and Settings\MyUserProfile\Local Settings\Apps*). Dadurch kann es nicht zu Komplikationen mit anderen installierten Anwendungen kommen. Anschließend wird die Anwendung in der Secure Sandbox ausgeführt. Dabei werden nur eingeschränkte Rechte zur Verfügung gestellt, die aufgrund der Herkunft der Anwendung (Intranet oder Internet) bestimmt

## Tabelle I

### Die Installation von Web-, ClickOnce- und MSI-Anwendungen im Vergleich.

Merkmals	Web	ClickOnce	MSI
Auto-Deployment	+	+	-
Installation/Ausführung pro Benutzer	+	+	-
Offline-Fähigkeit	-	+	+
Windows Shell Integration	-	+	+
Uneingeschränkte Installation	-	-	+



**Tabelle 2**

**Die ClickOnce-Einstellungen der Projekteigenschaften in Visual Studio 2005.**

Option	Beschreibung
Publishing Location	Gibt den Ort an, an den die Anwendung kopiert werden soll. Dabei stehen die folgenden Pfade zur Auswahl: <ul style="list-style-type: none"> <li>• Dateipfad: c:\Deploy\MyApp</li> <li>• Fileshare: \\MyServer\MyApp</li> <li>• FTP-Server: ftp://ftp.csharp.at</li> <li>• Website: www.MyDomain.com/MyApp</li> </ul>
Installation URL	Gibt eine Website an, von der die Anwendung installiert werden kann, zum Beispiel ein Staging-Server. (optional)
Documentation and Support URL	Gibt eine Website an, auf der die Benutzer Support-Informationen über die Anwendung finden. Dieser URL wird im <i>Software-Control</i> der Systemeinstellungen angezeigt. (optional)
The application is available online only	Die Anwendung muss jedes Mal über den URL gestartet werden.
The application is available offline as well	Die Anwendung steht auch offline zur Verfügung. Es wird ein Startmenüeintrag und ein Eintrag im <i>Software-Control</i> angelegt.
For CD installations, automatically start Setup when CD is inserted	Es wird die Datei <i>autorun.inf</i> erzeugt, über die eine CD automatisch beim Einlegen gestartet werden kann.
Publish Version	Gibt die Versionsnummer an, die auf den Deployment-Server kopiert werden soll. Bei jedem Kopiervorgang auf den Server wird die Revision-Nummer erhöht.

**Tabelle 3**

**Die möglichen Optionen, die das Verhalten der Software bei einem Update festlegen.**

Option	Beschreibung
The application should check for updates	Die bereitgestellte Anwendung kann per ClickOnce aktualisiert werden.
Before the application starts	Bei jedem Start der Anwendung wird geprüft, ob ein Update vorliegt.
In the background as the application is running	Während der Ausführung des Programms wird geprüft, ob ein Update vorliegt.
Check every time the application runs	Es wird im Hintergrund geprüft, ob ein Update vorliegt.
Check every ...	Ein Zeitintervall, in welchem das Vorhandensein eines Updates ermittelt wird.
Allow the users to choose whether to accept the update	Hier können Sie festlegen, ob es sich um ein verpflichtendes Update handelt. Wenn ja, wird der Benutzer nicht gefragt, ob er es installieren will.
Update location	Hier können Sie den Pfad angeben, von dem das Update installiert werden soll.

werden. Diese Sicherheitseinstellungen können natürlich mit einer benutzerdefinierten Policy überschrieben werden, welche erweiterte Rechte aufgrund des Hashs, des Publishers oder des Strong Names zur Verfügung stellt. Solche benutzerdefinierten Policies müssen jedoch separat auf den Clients eingerichtet werden, etwa über ein MSI-Package.

Soll die Anwendung nicht offline zur Verfügung stehen, endet hier das Deploy-

ment. Wenn der Benutzer die Anwendung das nächste Mal starten will, klickt er wieder auf den Link, und der Vorgang beginnt von vorn. Die Anwendung wird allerdings nur dann neu vom Server geladen, wenn ein Update zur Verfügung steht. Dazu werden einfach die Zeitstempel der lokal gespeicherten Version und der Version auf dem Server verglichen.

Soll die Anwendung auch offline verfügbar sein, schließen sich weitere Schritt-

te an den Download an. Als Erstes wird für die Anwendung ein neuer Eintrag im Startmenü hinzugefügt. Dadurch lässt sich die Anwendung auch starten, wenn keine Verbindung zum Netzwerk besteht. Sodann wird ein neuer Eintrag unter *Software* in der Systemsteuerung hinzugefügt. Dadurch ist es möglich, zu einer früher installierten Version der Anwendung zurückzukehren oder die Anwendung komplett vom Client-PC zu löschen – vorausgesetzt, es handelte sich nicht um ein verpflichtendes Update.

Das Aktualisieren von ClickOnce-Anwendungen ist umfangreicher als das erste Einrichten, da es hierbei mehr Optionen gibt. Darunter fällt beispielsweise, ob die Suche nach neuen Updates automatisch oder manuell geschehen soll und ob dies vor dem Start der Anwendung oder während ihres Betriebs im Hintergrund passieren soll. Des Weiteren besteht auch die Möglichkeit, Aktualisierungen programmatisch über das ClickOnce-API anzustoßen.

### Das Design von ClickOnce-Anwendungen

Das Ziel von ClickOnce ist es, das Deployment von Windows-Anwendungen zentral zu verwalten und zu vereinfachen. Es soll aber nicht die aktuelle Installationstechnologie der MSI-Packages ersetzen. ClickOnce ist auch nicht für alle Arten von Windows-Anwendungen gedacht. Wie bereits erwähnt, stehen einer ClickOnce-Anwendung keine Administrationsrechte zur Verfügung. Daher sind während der Einrichtung auch keine Operationen erlaubt, die solche Rechte benötigen. Darunter zählen zum Beispiel Dateizugriffe oder der Zugriff auf die Registrierungsdatenbank.

Auch während der Installation stehen keine Administrationsrechte zur Verfügung. Daher ist es nicht möglich, gemeinsam genutzte Assemblies im GAC (Global Assembly Cache) zu installieren, Registrierungseinträge hinzuzufügen, Windows Services oder gar COM+-Komponenten einzurichten. Dies ist bereits beim Design der Anwendung zu berücksichtigen.

Sie können Ihre Anwendungen allerdings so entwerfen, dass all diese Punkte während einer Installation nicht notwendig sind. Das Hauptaugenmerk einer ClickOnce-Anwendung liegt im Grunde bei der Präsentation einer „Rich Client UI“ und bei der Interaktion mit Services



**Abbildung 2** Die ClickOnce-Optionen eines Programms sind Teil der Projekteigenschaften.

im Netzwerk, beispielsweise mit Web Services. Es handelt sich bei einer ClickOnce-Anwendung also um einen traditionellen Smart Client.

Darüber hinaus bestehen für ClickOnce-Anwendungen keine bestimmten Anforderungen bei der Entwicklung. Die komplette Funktionalität, die ClickOnce zur Verfügung steht, wird von der .NET Runtime angeboten, ohne dass die Anwendung in irgendeiner Weise angepasst werden muss. ClickOnce kann somit automatisiert ohne Code-Änderungen zu bestehenden Anwendungen hinzugefügt werden.

Wenn eine Anwendung für ClickOnce vorbereitet wird, werden auch eine *setup.exe* und alle benötigten Komponenten wie etwa SQL Server 2005 Express Edition, MDAC, DirectX und andere vorbereitet. Dadurch ist es auch möglich, dass eine ClickOnce-Anwendung auf einer CD ausgeliefert werden kann. Tabelle 1 zeigt die Unterschiede zwischen der Installation einem Web-, einem ClickOnce- und einem MSI-Deployment.

### Eine Beispielanwendung

So viel zu den Grundlagen von ClickOnce. Doch es wird Zeit, die Theorie in die Praxis umzusetzen. Dazu können Sie eine neue Windows-Anwendung unter Visual Studio 2005 anlegen und in den Projekteinstellungen die ClickOnce-Optionen festlegen (Abbildung 2). Die Bedeutung der einzelnen Optionen erklärt Tabelle 2.

Alle Einstellungen, die Sie hier vornehmen, werden gespeichert und stehen dann beim eigentlichen Kopiervorgang auf den Deployment-Server als Standardwerte zur Verfügung. Für die Beispielanwendung brauchen Sie nur die *Publi-*

*shing Location* und die Option *The Application is available offline as well* anzugeben. Dadurch werden für die Anwendung ein Startmenüeintrag und ein Eintrag in den Systemeinstellungen unter *Software* hinzugefügt. Die Anwendung ist nun auch ohne Verbindung zum Netzwerk verfügbar.

Durch Klick auf den Button *Application Files...* öffnet sich der Dialog, den Sie in Abbildung 3 sehen. Hier können Sie angeben, welche Dateien auf den Server kopiert werden sollen. Im vorliegenden Fall wurden zwei Dateien als notwendig gekennzeichnet: die *exe*-Datei der Anwendung und eine Klassenbibliothek, die der Projektmappe hinzugefügt wurde. ClickOnce ermittelt alle Abhängigkeiten der Anwendung selbstständig und zeigt sie in diesem Dialog an. Die PDB-Dateien mit den Debuginformationen der Anwendung werden hier nicht als notwendig gekennzeichnet, da sie zum Ausführen des Programms nicht benötigt werden.



**Abbildung 3** ClickOnce zeigt die zur Installation notwendigen Dateien an.

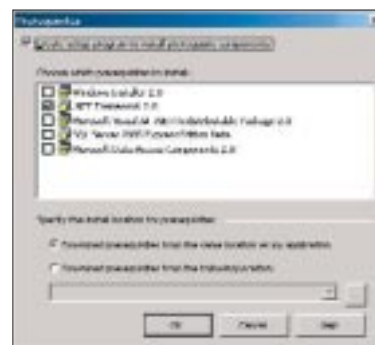
Über die Schaltfläche *Prerequisites* lassen sich die Komponenten auswählen, die ebenfalls auf den Deployment-Server kopiert werden sollen.

Zurzeit stehen die folgenden Komponenten zur Auswahl:

- Windows Installer 2.0,
- .NET Framework 2.0,
- Microsoft Visual J# .NET Redistributable Package 2.0,
- SQL Server 2005 Express Edition Beta,
- Microsoft Data Access Components 2.8.

Sehr interessant ist der Dialog, der sich hinter der Schaltfläche *Updates...* verbirgt (Abbildung 5). Über diesen Dialog können Sie festlegen, wie sich die Anwendung bei einem Update verhalten soll. Dazu stehen die Optionen zur Auswahl, die Sie in Tabelle 3 sehen.

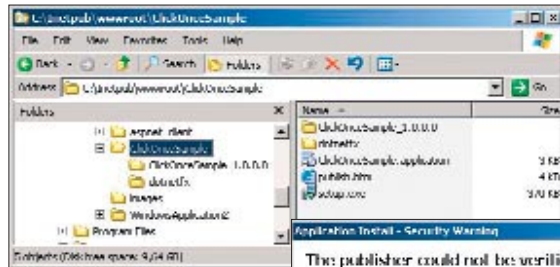
Wenn alle Einstellungen feststehen, kann der Kopiervorgang beginnen. Dazu steht der Menüpunkt *Build/Publish Solution* zur Verfügung, der den Publishing-Assistenten startet. Dieser Assistent führt



**Abbildung 4** Zusätzlich zur Anwendung lassen sich bestimmte Standardkomponenten bereitstellen.



**Abbildung 5** Dieser Dialog legt die Update-Optionen fest.



**Abbildung 6** Die auf dem Server erzeugte Ordnerstruktur für die ClickOnce-Anwendung.

**Abbildung 7** Bei einem unbekannten Publisher muss der Anwender selbst über die Vertrauenswürdigkeit entscheiden.



durch die verschiedenen Schritte, die notwendig sind, um die Anwendung auf den Server zu kopieren. Dabei werden die Projekteinstellungen der Registerkarte *Publish* verwendet.

Nach dem Kopieren wird eine Website angezeigt, über welche die Anwendung direkt installiert werden kann. Hierbei wird ein Link auf das Deployment-Manifest der Anwendung zur Verfügung gestellt (<http://localhost/ClickOnceSample/ClickOnceSample.application>). Das Deployment-Manifest hat dabei die Dateinamenerweiterung *.application*. Das .NET Framework erkennt diese Erweiterung beim Öffnen und startet die Installation. Bevor wir die Anwendung installieren, möchte ich zeigen, was auf den Deployment-Server kopiert wurde und wie das Deployment- und das Application-Manifest aufgebaut sind.

Um überhaupt von einem Deployment-Server aus eine Anwendung installieren zu können, wird auf ihm ein virtuelles Verzeichnis für die notwendigen Dateien der Anwendung erstellt. Die Abbildung 6 zeigt die Verzeichnisstruktur, die für unsere Anwendung am Deployment-Server erstellt wurde.

Im Verzeichnis *dotnetfx* befindet sich das .NET Framework 2.0, da dies im Dialog *Prerequisites* ausgewählt wurde. So kann das .NET Framework 2.0 direkt von den Clients von diesem Pfad geladen und installiert werden. Bei der ersten Installation wird aus dem Anwendungsnamen, einem Unterstrich und der Versionsnummer 1.0.0.0 der Pfad zusammengesetzt, in den die Anwendungsdaten kopiert werden (*ClickOnceSample\_1.0.0.0*). In diesem Verzeichnis befindet sich dann

das eigentliche Programm, bestehend aus einer *exe*-Datei, einer Klassenbibliothek und dem Application-Manifest *ClickOnceSample.exe.manifest*.

Im Root-Verzeichnis befinden sich drei Dateien: die Datei *setup.exe*, welche die Installation zum Beispiel von einer CD starten kann, die Datei *publish.htm*, die nach dem Publishing-Assistenten angezeigt wird, und die Datei *ClickOnceSample.application*, das Deployment-Manifest.

Das Deployment-Manifest ist eine XML-Datei, die Setup und Update der Anwendung konfiguriert. Listing 1 enthält eine vereinfachte Darstellung dieser Datei.

Das Element `<assemblyIdentity>` enthält einen Verweis auf das Deployment-Manifest, und die Versionsnummer der Anwendung (1.0.0.0). `<deployment install="true">` bedeutet, dass die Anwendung auf dem lokalen PC installiert wird und Einträge im Startmenü und im *Software-Control* der Systemeinstellungen angelegt werden. Unter dem Knoten `<update>` wird festgelegt, wie sich die Anwendung bei einem Update verhalten soll. `<beforeApplicationStartup>` definiert, dass bei Anwendungsstart etwaige Updates gesucht werden sollen.

Unter dem Knoten `<dependency>` ist anschließend eine Referenz auf das Application-Deployment enthalten (die Datei *ClickOnceSample.exe.manifest*). Hier wird über das Element `<dependentAssembly>` festgelegt, welche Version der Anwendung die aktuelle ist. Wird hier auf ein anderes Application-Manifest verwiesen, bedeutet dies für den Client, dass ein Update der Anwendung vorliegt, das installiert werden muss. Der wichtigste Knoten ist `<Signature>`. Er enthält eine

### Listing 1

#### Das Deployment-Manifest.

```
<asmv1:assembly>
  <assemblyIdentity
    name="ClickOnceSample.application"
    version="1.0.0.0"
    publicKeyToken="..."
    language="neutral"
    processorArchitecture="msil" />
  <deployment install="true">
    <subscription>
      <update>
        <beforeApplicationStartup />
      </update>
    </subscription>
    ...
  </deployment>
  <dependency>
    <dependentAssembly
      codebase="ClickOnceSample_1.0.0.0\
        ClickOnceSample.exe.manifest"
      size="3513" />
    <assemblyIdentity
      name="ClickOnceSample.exe"
      version="1.0.0.0"
      publicKeyToken="..."
      language="neutral"
      processorArchitecture="msil" />
    ...
  </dependency>
  <Signature>
    ...
  </Signature>
</asmv1:assembly>
```

digitale Signatur, die das komplette Deployment-Manifest kennzeichnet. Sie können es daher nicht manuell ändern, sondern nur über den Eigenschaftendialog in Visual Studio 2005 oder über das Framework-Tool *mage.exe*.

Ein vereinfachtes Deployment-Manifest, das die Anwendung näher beschreibt, sehen Sie in Listing 2. Der Knoten `<entryPoint>` gibt an, welche Datei der Einstiegspunkt in die Anwendung ist. Über den Knoten `<commandLine>` können ihr Startparameter mitgegeben werden. `<dependency>` beschreibt die Komponenten der Anwendung, hier die *.exe*-Datei, die Klassenbibliothek sowie die Hauptkomponenten des .NET Frameworks. Zum Schluss sehen Sie wieder den Knoten `<Signature>`, der das komplette Application-Manifest signiert.

#### Die Installation der Anwendung

Nun ist es langsam an der Zeit, die Beispielanwendung zu installieren. Dazu

## Listing 2

### Das Application-Manifest.

```
<asmv1:assembly>
  <assemblyIdentity name="ClickOnceSample.exe"
    version="1.0.0.0"
    publicKeyToken="..."
    language="neutral"
    processorArchitecture="msil" />
  <entryPoint>
    <assemblyIdentity
      name="ClickOnceSample"
      version="1.0.0.0"
      language="neutral"
      processorArchitecture="msil" />
    <commandLine
      file="ClickOnceSample.exe"
      parameters="" />
  </entryPoint>
  <trustInfo>...</trustInfo>
  <dependency>
    <dependentAssembly
      codebase="ClickOnceSample.exe"
      size="20480">
      <assemblyIdentity
        name="ClickOnceSample"
        version="1.0.0.0"
        language="neutral"
        processorArchitecture="msil" />
      </dependentAssembly>
    </dependency>
    <dependentAssembly
      codebase="ClickOnceSampleLibrary.exe"
      size="16384">
      <assemblyIdentity
        name="ClickOnceSample"
        version="1.0.0.0"
        language="neutral"
        processorArchitecture="msil" />
      </dependentAssembly>
    </dependency>
    <dependentAssembly preRequisite="true">
      <assemblyIdentity
        name="Microsoft-Windows-CLRCoreComp"
        version="2.0.3600.0" />
      </dependentAssembly>
    </dependency>
  </dependency>
  <signature>...</signature>
</asmv1:assembly>
```

müssen Sie nur die Datei *publish.htm* im Internet Explorer oder das Deployment-Manifest über einen Doppelklick öffnen. Als Erstes müssen Sie die Installation der Anwendung bestätigen, da deren Publisher – in diesem Fall sind das Sie selbst – als nicht vertrauenswürdig eingestuft wird.

Der Klick auf den Button *Install* richtet die Anwendung im lokalen Cache sowie die Systemeinträge ein und startet das Programm. Anschließend können Sie das Programm wie eine Windows-Forms-Anwendung aus dem Startmenü aktivieren – auch wenn keine Verbindung zum Netzwerk besteht.

Hat alles geklappt? Dann erstellen Sie nun eine neue Version der Anwendung, zum Beispiel einfach nur mit einer anderen Hintergrundfarbe des Formulars. Wenn Sie danach den Publishing-Assistenten erneut ausführen, wird auf dem Server ein neuer Ordner mit der neuen Programmversion *ClickOnceSample\_1.0.0.1* angelegt und das Deployment-Manifest entsprechend geändert, damit es auf das Application-Manifest der neuen Programmversion zeigt (Abbildung 8).

Wenn Sie nun über das Startmenü auf dem Client-PC die Anwendung starten, erscheint ein Dialog und informiert Sie darüber, dass ein Update des Programms zur

Verfügung steht. Bestätigen Sie dies, wird das Update heruntergeladen und die aktualisierte Anwendung gestartet. Außerdem haben Sie über die Systemeinstellungen nun sogar die Möglichkeit, die vorherige Version der Anwendung wiederherzustellen, wenn es sich nicht um ein verpflichtendes Update handelt hat.

### Fazit

Zurück zum Ausgangspunkt: Der Vorteil von Webanwendungen hinsichtlich der einfachen zentralen Installation bekommt mit ClickOnce unter Visual Studio 2005 einen starken Konkurrenten. Die Technologie erlaubt es Ihnen, Ihre Windows-Anwendungen ebenso wie Webanwendungen von einer zentralen Stelle aus zur Installation bereitzustellen. Dabei steht Ihnen Visual Studio 2005 zur Seite und macht Ihre Windows-Programme „ClickOnce-ready“. Der Clou dabei ist, dass Sie keine einzige Code-Zeile dafür ändern müssen. Überlegen Sie sich bei der nächsten Anwendung, ob es sich wirklich lohnt, sie als Webanwendung unter ASP.NET zu entwickeln. |||||



**Abbildung 8** Die Einrichtung einer neuen Programmversion erweitert die Installationsordner des Servers.