B30751

dotnetpro

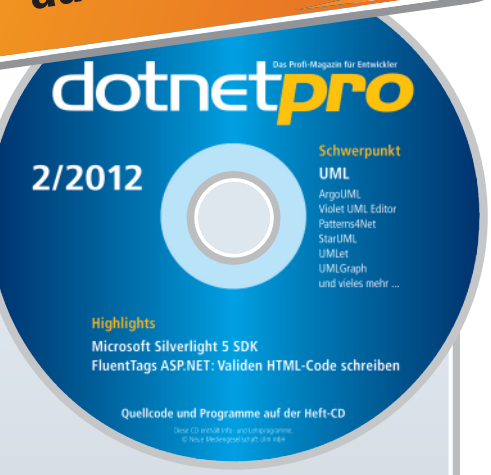
Das Profi-Magazin für Entwickler

Ausgabe 2/2012

14,90 € Deutschland 14,90 € Österreich 16,40 € Schweiz 29,80 sfr

Quellcode und Tools auf der CD im Heft

Jobs für Entwickler ab S. 143



Microsoft Azure schrittweise nutzen

- Cloud schon in der Architektur berücksichtigen S. 22
- Persistenz unter Microsoft Azure S. 28
- AppFabric: Modulare Middleware S. 38

Dynamic Language Runtime

Das Wundermittel für int

Schwerpunkt Tools für UML

- **ArgoUML**
  Open-Source-UML-Modellierungswerkzeug für UML 1.4
- **Violet UML Editor**
  Kostenloser, plattformübergreifender Diagrammeditor
- **UMLet**
  Open-Source-UML-Werkzeug mit einfacher Bedieneroberfläche
- **UMLGraph**
  Zeichnen von Klas
- **T**
  Editor für 14 inklusive UML-Modellierungswerkzeugen

witch im Praxiseinsatz

Für Standardszenarien genau das passende Werkzeug S. 92

Test-Pyramide bringt Sicherheit

Qualitätssicherung in Windows-Embedded-Projekten S. 72

Behavior Driven Development

Lebende Spezifikationen mit Gherkin und SpecFlow S. 66

INFO-Programm gemäß § 14 JuSchG

4 191709 414901 02

Cloud-fähige HTML5-Anwendungen mit Visual WebGui

Eine Sache von zehn Minuten

Anwendungen für Web und Cloud schreiben, ohne sich dabei in dem Gewirr erforderlicher Technologien zu verstricken: Das klappt jetzt mit Visual WebGui, wie dieser Workshop anschaulich zeigt.

Auf einen Blick

Günter Hofmann leitet den Geschäftsbereich Software Services bei fecher. Die dort für die weltweiten Kunden erarbeiteten Softwarestrategien reichen von der Entwicklung individueller Softwarekomponenten auf Basis des .NET-Frameworks bis zur Migration ganzer Legacy-Anwendungen und Datenbanken.

Inhalt

- Vorstellung des Open-Source-Frameworks Visual WebGui.
- Windows-Forms-Anwendungen laufen automatisch auch im Web.
- Eine Beispielanwendung mit Datenbankzugriff.

dnpCode

A1202VisualWebGui

fecher e.Kfm.
Seestr. 2-4
63110 Rodgau
Telefon (06106) 605-0
Telefax (06106) 605-200
www.fecher.eu
Guenter.Hofmann@fecher.eu

Jeder, der sich schon einmal daran versucht hat, ernsthafte Businessanwendungen für die Cloud zu schreiben, kennt das Problem: Einerseits soll die Software ohne Clientinstallation direkt im Browser laufen und idealerweise auch mobile Geräte wie Tablets und Smartphones unterstützen. Andererseits soll sie ohne Komforteinbußen die volle Funktionalität einer echten Desktopanwendung oder mobilen App bieten. Und vor allem will man sich bei Entwicklung und Betrieb nicht in einem ganzen Gewirr von Technologien verstricken. Abhilfe verspricht hier das Open-Source-Framework Visual WebGui, das in der aktuellen Version 6.4 ganz auf HTML5 setzt – und den Entwickler davon nichts merken lässt.

Die heutigen Internettechnologien stammen bekanntlich aus den Achtzigerjahren des vorigen Jahrhunderts und wurden dafür entwickelt, einzelne statische, textbasierte Informationsseiten zu veröffentlichen und über Links miteinander zu verknüpfen. Auch dreißig Jahre Weiterentwicklung der Webstandards konnten das nicht grundsätzlich ändern. So gilt es unter Entwicklern mittlerweile als Binsenweisheit, dass der Einsatz von Webtechnologie als grafischer Benutzeroberfläche für dynamische Geschäftsanwendungen zwar wünschenswert, aber nur mit erheblichem Aufwand und unter Inkaufnahme vielfältiger Kompromisse machbar ist.

Die Open-Source-Welt setzt für diese Rich Internet Applications (RIAs) konsequent auf die LAMP-Technologien, also die Kombination von Linux, Apache, mySQL und PHP oder Perl. Bei Microsoft aber scheint die Strategie nicht mehr so klar: ASP wurde 2002 zu ASP.NET weiterentwickelt und 2007 um Silverlight ergänzt, das als Browser-Plug-in die Ausführung von RIAs übernimmt und auch die Basis von Windows Phone 7 bildet. Wegen seines proprietären Charakters konnte sich Silverlight aber nicht wie geplant durchsetzen; seit der Keynote von Steve Ballmer auf der Entwicklerkonferenz PDC 2010 ist davon auszugehen, dass es strategisch von HTML5 abgelöst wird.

Das Beste aus Open-Source- und Microsoft-Welt

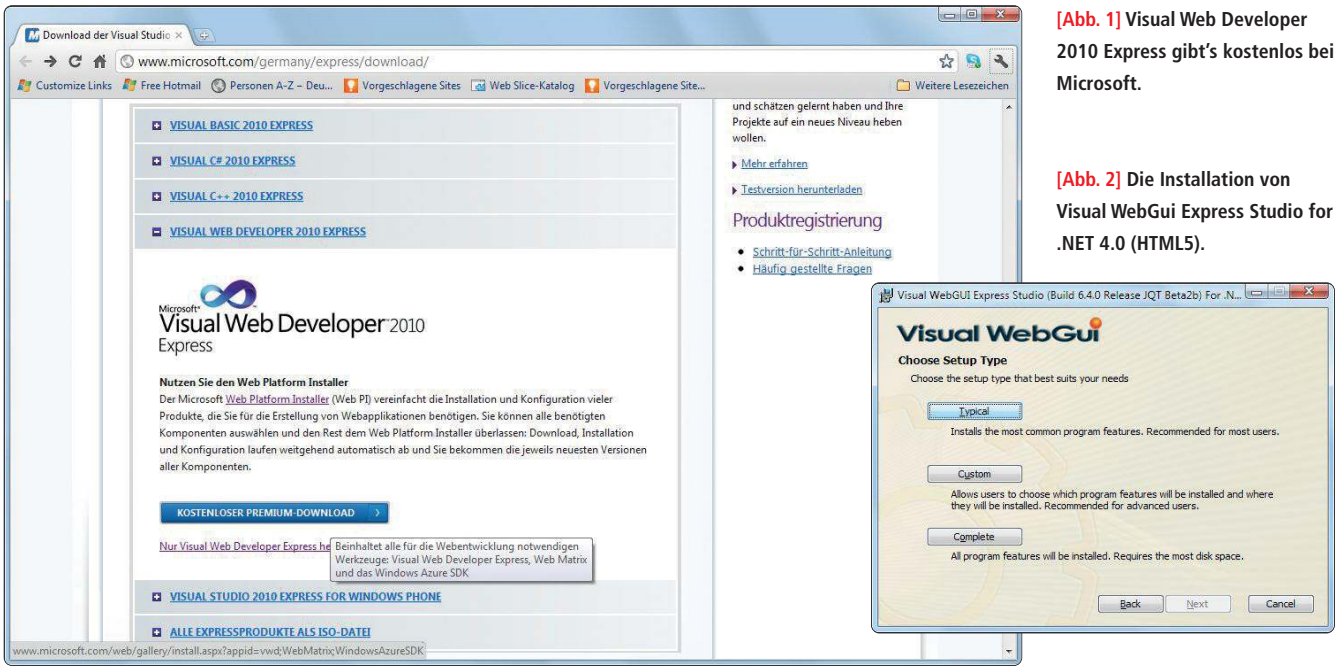
Für den .NET-Entwickler stellt sich damit erneut die Frage, wie er den Brückenschlag von der gewohnten Entwicklung von Windows-Forms-Anwendungen hin zur RIA vollziehen soll. Einen zumindest aus Programmierersicht erfrischend einfachen Ansatz liefert Visual WebGui von der israelischen Softwareschmiede Gizmox: Das Open-Source-Framework setzt auf ASP.NET, Ajax und die jQuery-Bibliothek, um auch in einer verteilten Umgebung von Webbrowser, Webserver, Applikationsserver und Datenbank genau die Funktionen des Windows-Form-APIs bereitzustellen. Dazu klinkt sich der Entwickler in das vertraute Visual Studio ein, und das Framework kümmert sich ohne weiteres Zutun selbst darum, welcher Code später wo abläuft und wie die einzelnen Komponenten sicher zusammenarbeiten. In der Theorie sollte damit also jeder, der schon einmal eine Windows-Forms-Anwendung entwickelt hat, ASP.NET-Applikationen schreiben können, die vom Smartphone über iPad und andere Tablets bis zum Desktop ohne Softwareinstallation oder Plug-ins im Browser ablaufen.

Dank eingebauter Virtualisierungstechnik kann die Serverplattform vom lokalen PC über große Unternehmensnetzwerke bis zur Cloud-Konfiguration reichen, Windows Azure sowie die Datenbankplattform Azure SQL werden dazu transparent unterstützt. Ob das in der Praxis tatsächlich so einfach funktioniert, soll ein kleiner Workshop an einem Beispiel zeigen. Alles, was dazu benötigt wird, ist kostenfrei erhältlich: Visual Studio 2010 Express kann man bei Microsoft laden, und die dazu passende Visual-WebGui-Express-Studio-Version für .NET 4.0 und HTML5 finden Sie auf der Heft-CD.

Die Praxis soll es zeigen

Wer die Express-Version von Visual Studio 2010 noch nicht installiert hat, kann sie über [1] beziehen. Dort werden verschiedene Varianten angeboten, benötigt wird der Visual Web Developer 2010 Express **(Abbildung 1)**. Dieser installiert auch gleich die weiteren Microsoft-Komponenten wie Webserver und SQL-Server, sodass neben der Entwicklungs- auch gleich die vollständige Testumgebung zur Verfügung steht.

Nun fehlt nur noch Visual WebGui, das Sie auf der Heft-CD finden **(Abbildung 2)**. Beim Anklicken öffnet sich ein Installationsfenster, in dem man sinnvollerweise *Typical* auswählt, um alle erforderlichen Standardkomponenten zu erhalten. Allerdings lässt sich diese kostenlose Version von Visual WebGui nur mit Visual Studio Express



[Abb. 1] Visual Web Developer 2010 Express gibt's kostenlos bei Microsoft.

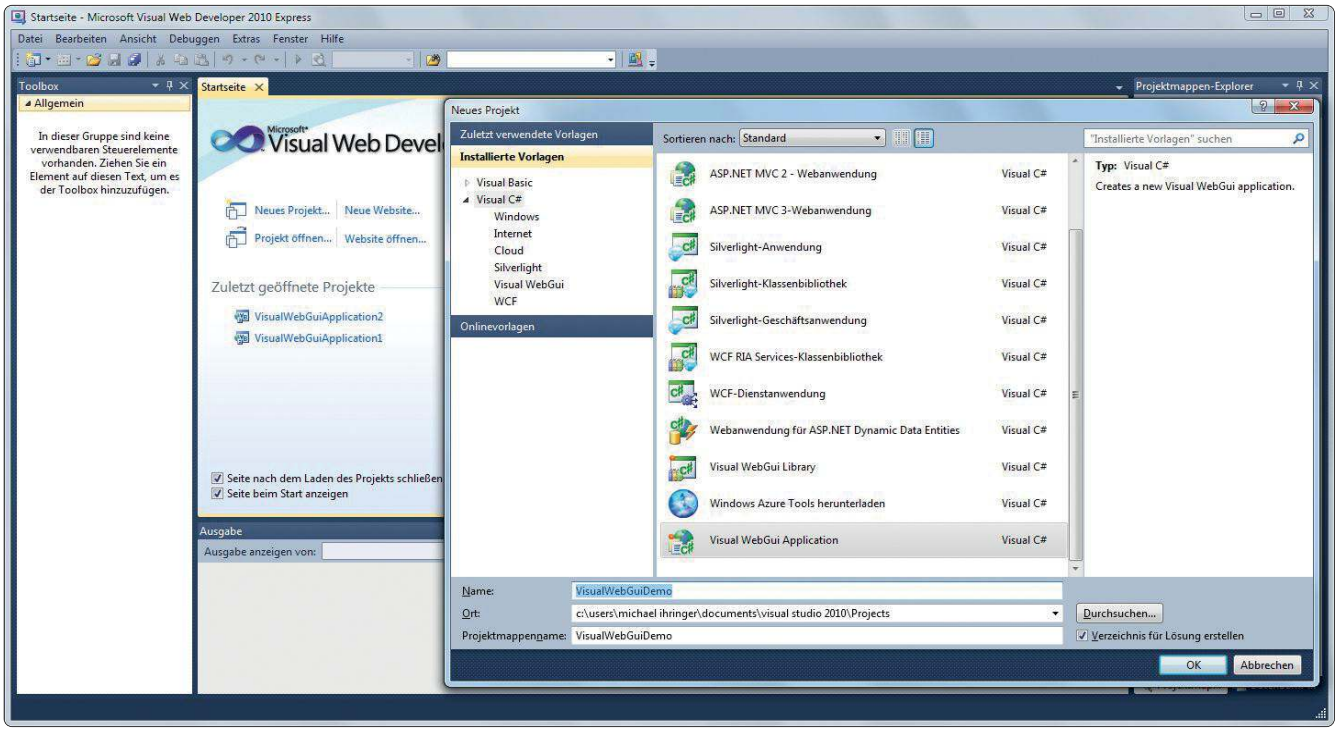
[Abb. 2] Die Installation von Visual WebGui Express Studio for .NET 4.0 (HTML5).

einsetzen. Wer über Visual Studio Professional verfügt (und die Express-Version nicht zusätzlich installieren möchte), findet unter [2] eine 14-Tage-Testversion von Visual WebGui ProStudio for .NET (HTML5), das mit dieser lizenzpflichtigen Version läuft. Bei der Auswahl des Downloads ist dann unbedingt darauf zu achten, dass man die neueste Version für .NET 4.0 und Visual WebGui .NET HTML5 nimmt. Ein Besuch der Website lohnt sich übrigens in jedem Fall, da dort auch die vollständige Dokumentation geladen werden kann.

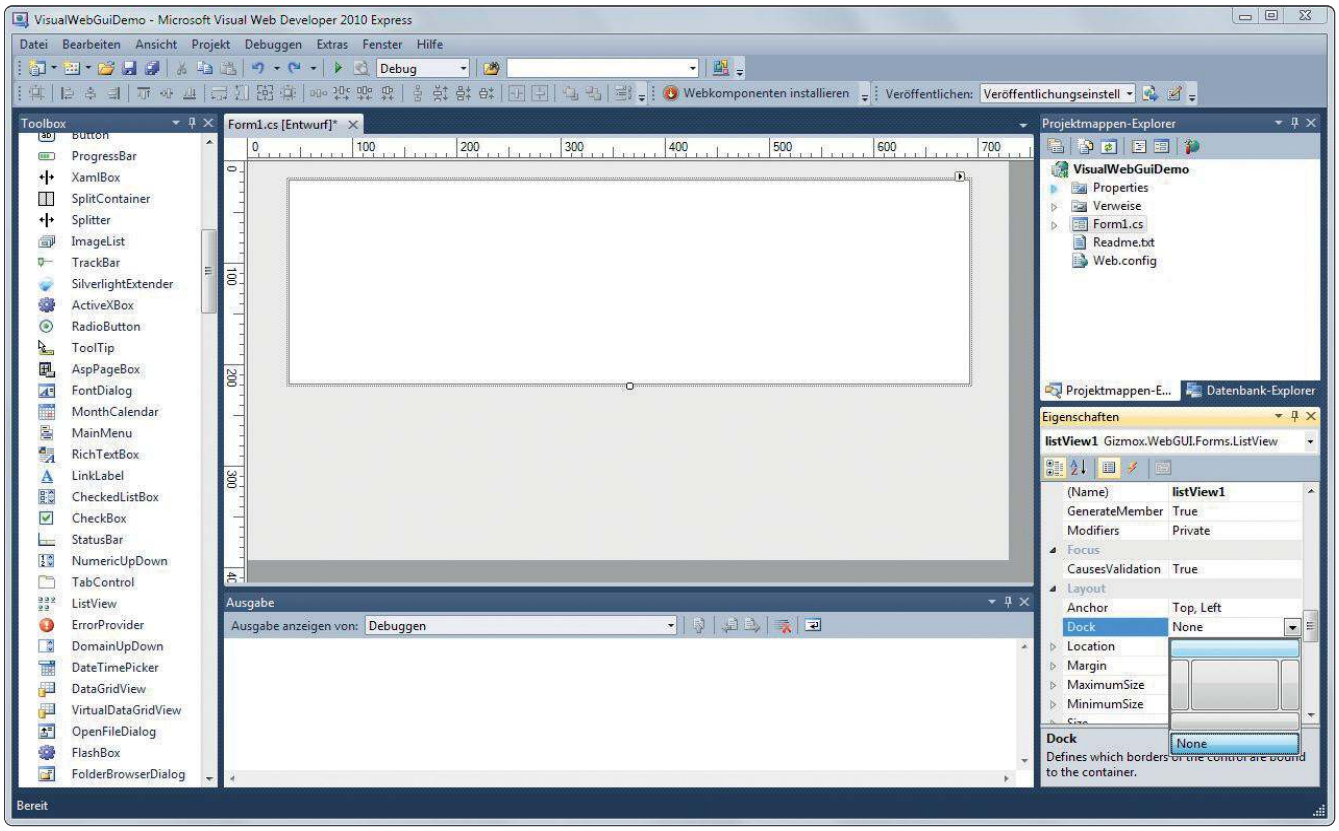
Auf los geht's los

Nach erfolgter Installation findet sich im Startmenü die Programmgruppe *Microsoft Visual Studio 2010 Express*. Dort lässt sich der Microsoft Visual Web Developer 2010 Express aufrufen, der bereits über die erforderlichen Visual-WebGui-Erweiterungen verfügt. Entsprechend können wir *Neues Projekt* auswählen, unter den Vorlagen wahlweise *Visual Basic* oder lieber *Visual C#* aktivieren und die Vorlage *Visual WebGui Application* auswählen. Dem Projekt schnell noch einen Namen gegeben – vielleicht *VisualWebGuiDemo* –, und dem Fertigstellen durch Klicken auf *Ok* steht nichts mehr im Wege **(Abbildung 3)**.

Zur Designzeit ist die Entwicklung mit Visual WebGui kaum von der gewohnten .NET-Entwicklung zu unterscheiden. Ent-



[Abb. 3] Ein neues Visual-WebGui-Projekt anlegen.



[Abb. 4] **ListView-Control von Visual WebGui.**

sprechend wird im Projektmappen-Explorer wie gewohnt das Formular *Form1.cs* gewählt und dann auf das neu erschienene zweite Symbol von rechts geklickt, wodurch sich der Ansicht-Designer öffnet. In diesem lassen sich mit Drag-and-drop die Controls platzieren, deren Auswahl in einer Toolbox auf der linken Seite erscheint – falls nicht, lässt sich diese durch Klicken auf den Menüpunkt *Ansicht | Weitere Fenster | Toolbox* oder durch gleichzeitiges Drücken von [Strg] [Alt] [X] öffnen. Auch wenn Designer und Werkzeuge aussehen wie gewohnt, handelt es sich hierbei um die speziellen Visual-WebGui-Varianten.

Bis auf ganz wenige Ausnahmen, die für die Webentwicklung nicht benötigt werden, sehen sie aus und verhalten sich auch wie die bekannten Windows-Controls. Der wichtige Unterschied besteht darin, dass sie im Hintergrund Code auf dem Webserver erzeugen. Um diesen serverseitigen Code muss sich der Entwickler allerdings nicht kümmern, er kann einfach weiterentwickeln wie bei normaler Client-GUI-Programmierung.

Machen Sie mit, und probieren Sie Visual WebGui anhand des Beispiels selbst aus: Ziehen Sie die Arbeitsfläche größer, und platzieren Sie darauf ein *ListView*-Objekt. Dieses vergrößern Sie mit der Maus ebenfalls noch etwas, dann klicken Sie mit der rechten Maustaste auf *Eigenschaften* und setzen unter *Layout* die Einstellung *Dock* auf *oben*, damit das Element oben am Fenster andockt **(Abbildung 4)**. Damit sich die Größe später verändern lässt, kommt noch ein Splitter darunter und wird ebenfalls oben angedockt.

Der Platz unterhalb des Splitters soll für Datenfelder genutzt werden. Als Grundlage dafür wird ein Panel platziert und dessen *Dock*-Eigenschaft auf *Fill* gesetzt, damit es die ganze verbleibende Fläche ausfüllt. Jetzt schnell noch eine *ComboBox* und eine *TextBox* daraufgesetzt, und das Layout ist fertig. Gegenüber normaler Windows-Programmierung war keine Besonderheit zu bemerken.

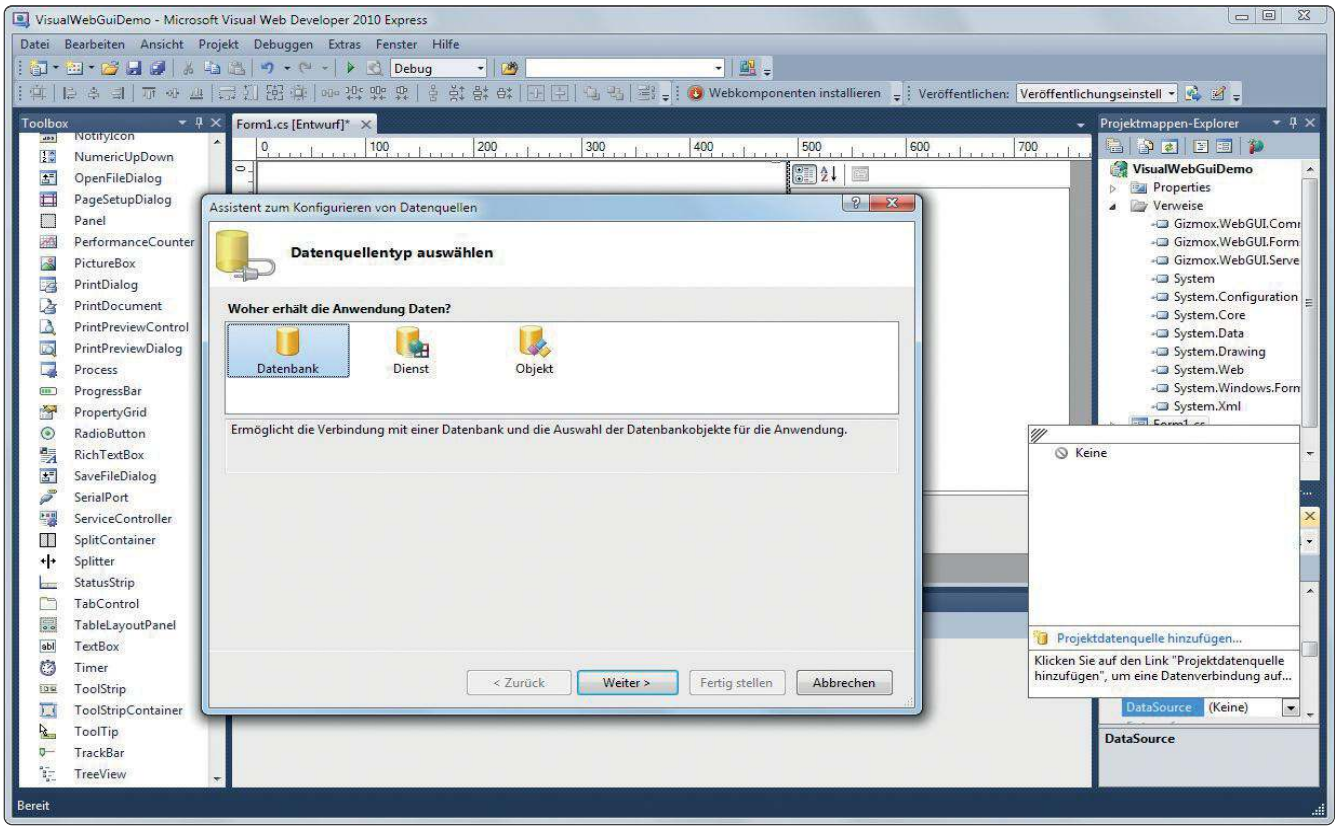
Im nächsten Schritt geht es daran, die *ListView* mit Daten zu füllen. Zu diesem Zweck verfügt diese über ein Attribut *DataSource*, das bislang leer ist. Hier lässt sich eine einfache „Datenbank“, durch die Auswahl *Objekt* aber auch eine ganze komplexe Anwendungslogik hinterlegen, die das UI-Element später füllt. Für das vorliegende Beispiel wird die SQL-Server-Beispieldatenbank Northwind [3] gewählt **(Abbildung 5)**, für deren Installation sich eine sehr gute Anleitung beispielsweise unter [4] findet.

Wählen Sie die Northwind-Datenbank als *DataSource*, können Sie den Verbindungsstring in das Projekt übernehmen und aus den enthaltenen Tabellen, Ansichten und gespeicherten Prozeduren wählen. Für das Beispiel kommt die Tabelle *Customers* zum Einsatz, deren Daten an das *ListView*-Element gebunden werden. Um dieses mit Daten zu füllen, fehlt nur noch eine einzige Zeile Code. Um diese einzugeben, machen Sie einen Rechtsklick auf das Formular und wählen im Kontextmenü *Code anzeigen*. Hier wird die *Load*-Methode für das Formular *Form1* wie folgt ergänzt:

```
private void Form1_Load(object sender,
  EventArgs e)
{
  customersTableAdapter.Fill(
    nORTHWNDDataSet.Customers);
}
```

Damit wird die *ListView* beim Laden des Formulars mit den Kundendaten aus der Northwind-Datenbank gefüllt.

Jetzt wird noch das Attribut *UseInternalPaging* des *ListView*-Elements auf *True* gesetzt, damit die anzuzeigenden Datensätze seitenweise geladen werden können und darin geblättert werden kann. Das ist zugleich ein Beispiel für die zusätzliche Funktionalität, die Visual-WebGui-Controls an



[Abb. 5] **Auswahl der Datenquelle für das ListView-Element.**

vielen Stellen gegenüber ihren gewöhnlichen Windows-Forms-Pendants zu bieten haben.

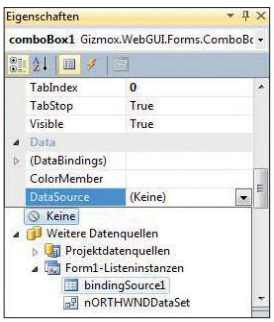
Zu guter Letzt bestimmen wir noch, welche Daten der ausgewählten Kundensätze in der Detailansicht zu sehen und auszuwählen respektive zu bearbeiten sein sollen. In der *ComboBox* dient dazu das Attribut *DataSource* **(Abbildung 6)**. Hier müssen Sie unter *Weitere Datenquellen Form1-Listeninstanzen* und dann *bindingSource1* auswählen, um jeweils die in der *ListView* ausgewählte Instanz zu erhalten. Die Eigenschaft *DisplayMember* gibt vor, welche Tabellenspalte angezeigt werden soll, wählen Sie dort *CompanyName* aus.

Ähnlich gehen wir bei der *TextBox* vor. Hier findet sich das fragliche Element unter *DataBindings* und heißt *Text* für den im Control anzuzeigenden Text. Dort wählen Sie wieder *Weitere Datenquellen*, *Form1-Listeninstanzen* und *bindingSource1*. Hier geht die Verzweigung allerdings gleich noch eine Ebene tiefer bis zu den Tabellenspalten, wo direkt der *ContactName* ausgewählt wird.

Damit ist das erste Visual-WebGui-Projekt fast schon fertig. Unter *Projekt | Visual-WebGuiDemo-Eigenschaften | Web* ist unter *Startaktionen* als *Bestimmte Seite* noch das Formular *Form1.wgx* einzutragen.

Jetzt wird das Projekt noch gespeichert und mit [F5] aufgerufen. Alternativ lässt es sich auch einfach über seinen URL (hier http://localhost:2701/Form1.wgx) aufrufen oder so auch verlinken, etwa aus einer statischen Website heraus.

Der erste Aufruf dauert einen Augenblick, weil zunächst der serverseitige Code generiert werden muss. Sobald die Anwendung läuft, ist die *ListView* bereits mit den Daten gefüllt **(Abbildung 7)**.

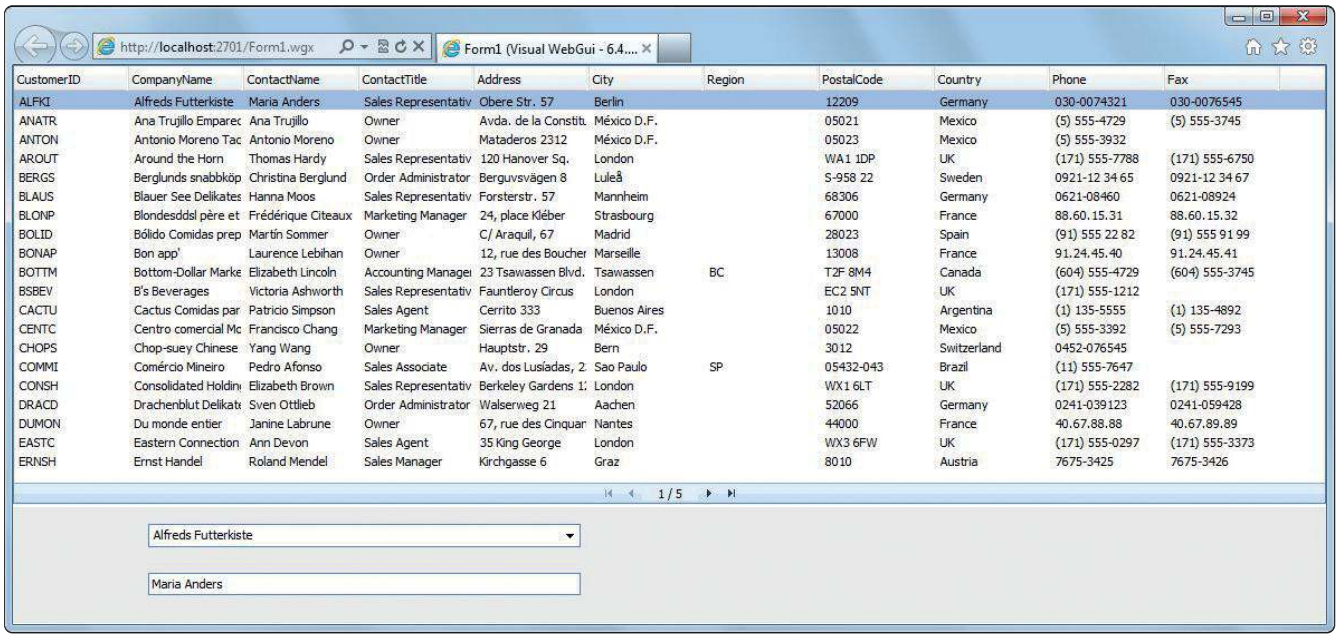


[Abb. 6] **Auswahl der DataSource der ComboBox.**

Ziel erreicht

Sieht man von den Installationszeiten für Visual WebGui und VS 2010 Express ab, so haben rund zehn Minuten ausgereicht, um die komplette Webanwendung aufzubauen – vom Design bis zur Laufzeit. Trotzdem bietet die Anwendung schon die wichtigsten Basisfunktionalitäten: Man kann die Kundenliste seitenweise durchblättern, durch Anklicken der Spaltenköpfe etwa alphabetisch oder nach Postleitzahl sortieren und natürlich einzelne Kunden auswählen, sodass in den Detailfeldern die Firma und der Kontaktnamen angezeigt werden. Ebenso kann man in der *ComboBox* einen anderen Kunden wählen oder den Kontaktnamen ändern. Die Änderung wird dann automatisch gespeichert und die Anzeige aktualisiert. Selbst wenn der Sitz des ausgewählten Unternehmens noch als Google-Map angezeigt werden soll, müssten wir nicht mehr tun, als eine Picture-Box dafür vorsehen und drei Zeilen Code für die *SelectedIndexChanged*-Methode der *ListView* schreiben, um Google mit den korrekten Parametern zur Erzeugung des Images zu versorgen und dieses dann als Quelle für die Picture-Box zu setzen.

Sämtliche damit zusammenhängenden Ereignisse – vom Anzeigen der Informationen im Browser über die Anwendungslogik im Applikationsserver bis zum Ändern in der Datenbank – in einer klassischen Webumgebung wie ASP.NET zu programmieren hätte sehr viel länger gedauert. So hingegen konnten wir, wie in .NET gewohnt, objektorientiert entwickeln und es dem Framework überlassen, sich um das Zu-



[Abb. 7] **Die Visual-WebGui-Anwendung im Browser.**

sammenspiel der verschiedenen Schichten und die Kommunikation dazwischen zu kümmern. Dabei wird auf Ereignisse reagiert und nur das absolut notwendige Maß an Informationen übertragen, wie man es aus dem Ajax-Modell kennt.

Der Browser muss dazu nur HTML und JavaScript verstehen. Ohne auf die Unterstützung durch spezielle Plug-ins oder Ähnliches angewiesen zu sein, läuft die Beispielanwendung deshalb auf allen Plattformen und in jedem Browser. Trotzdem muss sie nicht wie eine Browseranwendung aussehen: Mit Visual WebGui werden Themen vom Office-2010-Look bis zum typischen Facebook-Design fertig mitgeliefert, und natürlich lassen sich eigene gestalten. Einen Eindruck von der Oberflächengestaltung gibt die Demoanwendung unter [5], die einen Mailclient mit ausgefeilter Benutzeroberfläche implementiert und auch die Auswahl verschiedener Themen ermöglicht (unter *View | Themes*). Lesen Sie dazu auch den Kasten **„Visual WebGui"**.

Fazit

Vor allem aber verhält sich die Anwendung auf dem Client wie eine native Anwendung der jeweiligen Plattform, sodass dem Anwender etwa auf dem iPhone eben auch die typischen Besonderheiten, wie Fingerbedienung oder Kamerabenutzung, zur Verfügung stehen. Dennoch findet sich ein Entwickler mit guten Windows-Forms-Kenntnissen, der auch VB.NET oder C# beherrscht, innerhalb weniger Stunden mit dem Werkzeug zurecht.

Dabei muss die Zielrichtung nicht immer die Neuentwicklung von Applikationen sein. Ganz im Gegenteil: Eine der Stärken der Visual-WebGui-Architektur besteht gerade darin, dass vorhandene Geschäftslogik leicht in die so entwickelten Webprojekte integriert werden kann.

Damit bietet das Framework unzähligen .NET-Anwendungen im klassischen Client-Server-Modell einen sanften Migrationspfad in die Cloud. Und auch für alle anderen Altanwendungen – nach neuesten Schätzungen sind noch allein 15 Milliarden Zeilen VB6-Code produktiv im Einsatz – kann die Migration eine überlegenswerte Alternative zur ungleich aufwendigeren Neuentwicklung bieten. **[bl]**

Visual WebGui

Die RIA-Plattform der israelischen Softwareschmiede Gizmox hat eine mehrschichtige Architektur und erfordert auf der Serverseite eine Microsoft-Plattform mit dem Standard-IIS-Webserver.
Der Client ist ein sogenannter Empty Client, dort ist also keinerlei Softwareinstallation erforderlich. Vielmehr werden alle Standard-Browserplattformen unter Windows, Linux und Mac sowie von iPhone, iPad oder Android unterstützt. Das Ajax-Framework unterstützt eine reiche Benutzeroberfläche, die wie eine native Anwendung alle Elemente der Clientplattform nutzt – zum Beispiel die Fingerbedienung auf Tablet oder Mobiltelefon.

DevTools
Mit den DevTools lassen sich Visual-WebGui-Anwendungen direkt in Visual Studio erstellen. Alle Oberflächeninformationen werden automatisch in XML, XSLT und CSS-Stylesheets aufgeteilt, die zur Laufzeit an den Browser gesendet werden.

CloudMove
Als CloudMove-Projekt bietet der deutsche Gizmox-Partner fecher die toolgestützte Migration vorhandener Desktop-Anwendungen in die Cloud, ins Internet oder auf mobile Devices zum Festpreis an [6].

Enterprise Mobile
Die Enterprise-Mobile-Komponenten ermöglichen den Einsatz einer Visual-WebGui-Anwendung auf mobilen Clients unter Nutzung sämtlicher Möglichkeiten der jeweiligen Plattform.

[1] Visual Studio 2010 Express, www.dotnetpro.de/SL1202VisualWebGui1
[2] Testversion von Visual WebGui ProStudio for .NET (HTML5), www.dotnetpro.de/SL1202VisualWebGui2
[3] SQL-Server-Beispieldatenbank Northwind, www.dotnetpro.de/SL1202VisualWebGui3
[4] Installationsanleitung Northwind, www.dotnetpro.de/SL1202VisualWebGui4
[5] Demoanwendung, www.dotnetpro.de/SL1202VisualWebGui5
[6] CloudMove von fecher, www.fecher.eu/vwg