

Windows Forms 2.0

Gut in Forms

Am 7. November 2005 wurde Visual Studio 2005 gemeinsam mit dem .NET Framework 2.0 released. Auch für die Windows-Forms-Programmierung gibt es viele neue Features und Verbesserungen. dotnetpro zeigt, welche neuen Steuerelemente es gibt und wie der Windows-Forms-Designer von Visual Studio 2005 Entwicklern die Arbeit erleichtert.

Das neue .NET Framework 2.0 und Visual Studio 2005 bestechen nicht nur durch Verbesserungen in der Webentwicklung, sondern auch durch eine Vielzahl von neuen Features in der Windows-Programmierung mit Windows Forms 2.0.

Wer sich bereits unter dem .NET Framework 1.x näher mit der Windows-Forms-Programmierung beschäftigt hat, wird festgestellt haben, dass es hier an manchen Stellen Probleme gegeben hat, die mit der Zeit lästig wurden. Oder haben Sie sich auch schon mal gewünscht, mit Windows Forms eine Oberfläche zu zaubern, die sehr stark an Office 2003 erinnert? Oder wer hat schon einmal versucht das Webbrowser-Steuerelement in eine Windows-Forms-Anwendung zu integrieren? Wie haben Sie sich mit dem Databinding-Modell gefühlt? Mit welchen Hürden war ein asynchroner Datenzugriff möglich? All das sind Dinge, mit denen es unter dem .NET Framework 1.x Probleme gegeben hat.

Windows Forms 2.0, das im .NET Framework 2.0 enthalten ist, versucht all diese Probleme zu lösen und auch neue Features zur Verfügung zu stellen, von denen Sie unter Windows Forms 1.x nur träumen konnten. In diesem Artikel unternehme ich daher einen Streifzug durch die Windows-Forms-Programmierung mit dem .NET Framework 2.0 und zeige dabei, welche neuen Features diese Bibliothek anbietet. Den Anfang sollen die allgemeinen Neuerungen bilden, mit denen Sie immer wieder bei der Windows-Forms-Programmierung konfrontiert werden.

Neue UI-Features

Mit dem Windows Forms Designer von Visual Studio 2003 konnten Sie leistungsfähige Oberflächen entwickeln, da Sie die komplette Oberfläche auf grafischem Wege zur Entwicklungszeit aufbauen konnten. Diese Vorgehensweise nennt sich RAD – Rapid Application Development – also schnelle Anwendungsentwicklung.

Aber war dieser Windows Forms Designer wirklich so ausgereift, dass Sie schnell und einfach Anwendungen entwickeln konnten? Ich würde diese Frage so beantworten: Ja, in gewissen Bereichen konnten wirklich schnell und einfach Anwendungen entwickelt werden, aber sobald Sie versucht haben, professionelle Anwendungen damit zu realisieren, sind Sie an die Grenzen des Windows Forms Designers gestoßen.

Wenn Sie zum Beispiel Anwendungsoberflächen entwickeln wollten, die an das Look-and-Feel von Office 2003 angepasst sein sollten, hatten Sie mit den standardmäßigen Bordmitteln von Windows Forms ein Problem, da es einfach nicht möglich war. Hierbei mussten Sie Fremdkomponenten verwenden, die Sie für viel Geld einkaufen mussten. Unter Windows Forms 2.0 stehen eine Reihe von

verschiedenen Steuerelementen zur Verfügung, mit denen Sie nun endlich Anwendungen auch nach dem Look-and-Feel von Office 2003 entwickeln können – mehr zu diesem Punkt aber später.

Ein anderer wichtiger Punkt war die generelle Handhabung des Windows Forms Designers. Es konnten zwar ganz schnell und einfach Steuerelemente auf einem Formular positioniert werden. Wenn Sie aber das Layout der Steuerelemente genau anpassen wollten, mussten Sie schon ein Meister in der Bedienung der Maus sein. Die andere Möglichkeit war, das Layout mithilfe des Eigenschaftfensters anzupassen beziehungsweise direkt in den Code zu wechseln.

Der Windows Forms Designer von Visual Studio 2005 bietet dagegen neue Features, mit denen Sie Steuerelemente auf einem Formular leichter ausrichten können. Dabei handelt es sich um die so genannten Führungslinien, die automatisch beim Einfügen eines neuen Steuerelementes auf dem Formular im Designer angezeigt werden. Abbildung 1 zeigt ein Beispiel.

Über diese Führungslinien haben Sie nun die Möglichkeit, Steuerelemente punktgenau auf dem Formular zu positionieren.

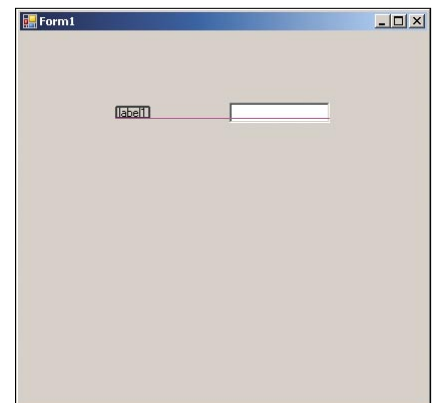


Abbildung 1 Führungslinien erleichtern die Ausrichtung.

Auf einen Blick

Autor



Klaus Aschenbrenner

(MVP) arbeitet als Software-Architect und Consultant bei der Firma ANECON in Wien. Sie erreichen ihn über www.csharp.at und weblogs.asp.net/klaus.aschenbrenner.

dotnetpro.code

A0602WinForms20

Sprachen C#

Technik Windows Forms 2.0

Voraussetzungen Visual Studio 2005

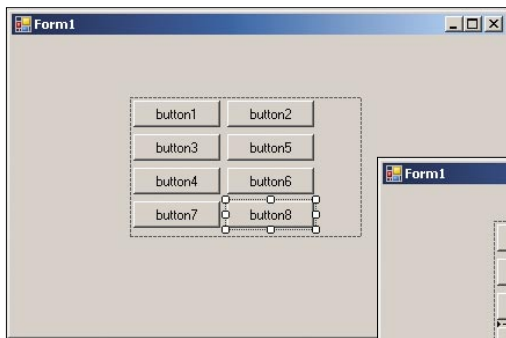
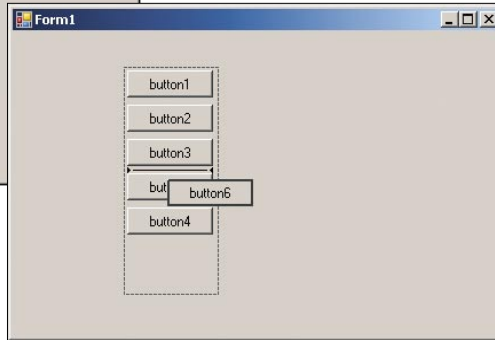


Abbildung 3 Markierungen erleichtern das Einfügen.

Abbildung 2 Layoutsteuerelemente ermöglichen einen automatischen Umbruch.



onieren und sie auch entsprechend den anderen Steuerelementen auszurichten. Daher ist es nun zum Beispiel kein Problem mehr, Textboxen mit den entsprechenden Label-Steuerelementen zu versorgen und auch an der richtigen Stelle zu positionieren. Ein weiteres Problem von Windows Forms 1.x war das Layouten von Steuerelementen. Hier gab es im Prinzip nur eine Möglichkeit, nämlich Steuerelemente über x- und y-Koordinaten auf der Oberfläche auszurichten. Bei diesem Ansatz gab es aber ziemlich oft Probleme, allein dann schon, wenn durch den Benutzer die Größe des Formulars zur Laufzeit geändert wurde.

Im Gegensatz dazu gibt es bei der Webentwicklung mit HTML hier keine Probleme, weil eine relative Positionierung der Steuerelemente innerhalb einer Seite vorgenommen werden kann. Die Verfügbarkeit dieses Features in HTML hat auch das Windows-Forms-Team dazu bewogen, entsprechende Äquivalente beim Windows-Forms-Programmiermodell anzubieten. Daher werden nun so genannte Layout-Steuerelemente unterstützt. Aktuell sind die folgenden verfügbar:

- *FlowLayoutPanel* zum Realisieren eines Layoutmechanismus, der sehr stark an HTML erinnert,
- *TableLayoutPanel* zum Realisieren eines Layoutmechanismus, der sich einer Tabelle im Hintergrund bedient.

Die beiden neuen Layoutmechanismen sollen ein wenig näher vorgestellt werden. Das *FlowLayoutPanel*-Steuerelement verwendet ein relatives Positionierungsschema für Steuerelemente. Daher bietet es Ihnen die gleichen Möglichkeiten an, die auch bei der Webentwicklung

mit HTML zur Verfügung stehen. Dieses Layout kann dann sehr von Vorteil sein, wenn das Layout von Steuerelementen automatisch angepasst werden soll, sobald sich die Größe des Containers (des Formulars) verändert hat. Wenn Sie Steuerelemente in diesem Layoutsteuerelement positionieren, werden diese nebeneinander angeordnet und entsprechende Zeilenumbrüche durchgeführt, wie es etwa Abbildung 2 zeigt.

Eine wichtige Eigenschaft des *FlowLayoutPanel*-Steuerelementes ist die Eigenschaft *FlowDirection*, über die festgelegt wird, in welcher Richtung die Steuerelemente angeordnet werden. Dabei stehen die folgenden Auswahlmöglichkeiten bereit:

- *LeftToRight*,
- *TopDown*,
- *RightToLeft*,
- *BottomUp*.

Wenn Sie die Reihenfolge von Steuerelementen innerhalb des *FlowLayoutPanel*-Steuerelements verändern, werden ebenfalls Markierungen angezeigt an denen das Steuerelement eingefügt werden kann, wie es Abbildung 3 zeigt.

Das *TableLayoutPanel*-Steuerelement kann im Gegensatz dazu verwendet werden, um das Layout von Steuerelementen in Form von einer Tabelle zu realisieren. Diese Vorgehensweise ist ebenfalls von der HTML-Programmierung her bekannt und in unterschiedlichen Bereichen einsetzbar. Sie haben die Möglichkeit, die verschiedenen Zellen zu formatieren. Dazu steht Ihnen der Column and Row Styles Editor zur Verfügung, den Abbildung 4 zeigt.

Ein weiteres wichtiges Steuerelement für das Layouten einer Oberfläche ist

SplitPanel, das das *Splitter*-Steuerelement aus dem .NET Framework 1.x ablöst. Beim *SplitPanel*-Steuerelement handelt es sich um eine Weiterentwicklung des Splitters, der nicht auf der Z-Order von Steuerelementen basiert. Dieses Steuerelement besteht insgesamt aus zwei *Panel*-Steuerelementen mit einem *Splitter* in der Mitte, der die beiden Panels voneinander trennt. Die beiden Panels können direkt über die beiden Eigenschaften *Panel1* und *Panel2* konfiguriert werden. Außerdem können Sie über die Eigenschaft *Orientation* die Richtung des Splittings festlegen. Dabei stehen die beiden Optionen *Horizontal* und *Vertical* zur Verfügung.

Ein weiteres neues Feature, das einige Windows-Forms-Steuerelemente anbieten, ist der *AutoComplete*-Mechanismus. Dieses Feature dürfte Ihnen bereits vom Internet Explorer her bekannt sein. Im Prinzip geht es darum, dass in einer Textbox Auswahlmöglichkeiten angeboten werden, sobald Sie Daten eingeben.

Dieses Feature wird zurzeit von den beiden Steuerelementen *TextBox* und *DropDownListBox* angeboten. Um dieses Feature bei den Steuerelementen zu aktivieren, müssen Sie nur die Eigenschaft *AutoCompleteMode* auf einen der folgenden Werte setzen:

- *Suggest*,
- *Append*,
- *SuggestAppend*.

Des Weiteren müssen Sie noch über die Eigenschaft *AutoCompleteSource* konfigurieren, von woher die Vorschlagswerte genommen werden sollen. Hierbei stehen die in Tabelle 1 gezeigten Optionen zur Auswahl. Wenn Sie die Option *CustomSource* verwenden, können Sie über die Eigenschaft *AutoCompleteCustomSource* ein String-Array hinterlegen, das als benutzerdefinierte Datenquelle für das *AutoComplete*-Feature verwendet wird. Wenn Sie diese Eigenschaft zur Laufzeit befüllen, stehen Ihnen hier alle erdenklichen Möglichkeiten zur Verfügung.

Neue Steuerelemente

Die zur Verfügung stehenden Steuerelemente von Windows Forms 2.0 wurden ebenfalls enorm aufgestockt. Dadurch ist es jetzt noch einfacher, leistungsfähige und professionelle Oberflächen auf der Basis von Windows Forms zu erstellen. Das *MaskedTextBox*-Steuerelement ermöglicht es, eine Maske zu hinterlegen,

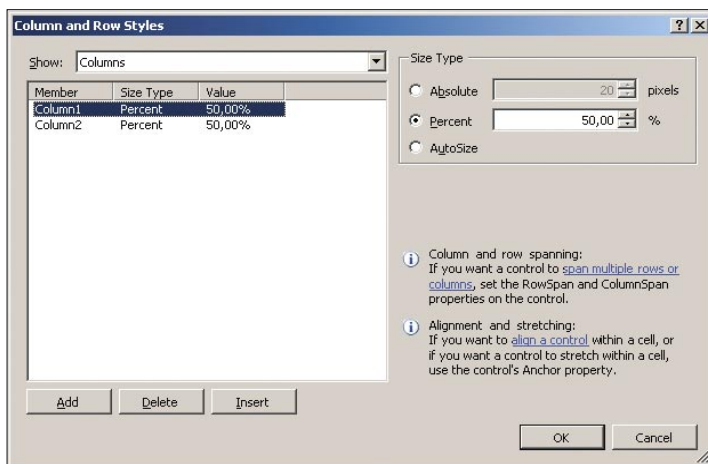


Abbildung 4
Eine Layout-
tabelle for-
matieren.

konfigurieren oder auch Tastenkombinationen ändern. Neuerungen hat es auch bei der Klasse *MenuStrip* gegeben, die ein Menü darstellt. Über das SmartTag dieser Komponente haben Sie nun die Möglichkeit, schnell und einfach Standard-Menüeinträge zum *MenuStrip* über den Befehl *Insert Standard Items* hinzuzufügen, wie es Abbildung 7 darstellt.

Außerdem können nun endlich auch andere Steuerelemente direkt in das Menü aufgenommen werden. Dazu zählen unter anderem Comboboxen und Textboxen. Dadurch können Menüs entwickelt werden, die sehr stark an das Look-and-Feel von Office 2003 erinnern.

So wie das *MenuStrip*-Steuerelement wurde auch das *ToolStrip*-Steuerelement komplett überarbeitet. Sie können über das SmartTag die Standard-Buttons in die Toolbar einfügen.

Beim *ToolStrip* haben Sie auch die Möglichkeit, verschiedene Steuerelemente zu verwenden, das heißt, Sie sind nicht nur auf *ToolStripButtons* limitiert. Unter anderem stehen Ihnen die folgenden Steuerelemente zur Verfügung:

- *Button*,
- *Label*,
- *SplitButton*,
- *DropDownButton*,
- *Separator*,
- *ComboBox*,
- *TextBox*,
- *ProgressBar*.

Wenn Sie über das SmartTag die Option *Embed in ToolStripContainer* auswählen, wird der komplette *ToolStrip* in einen *ToolStripContainer* verpackt, mit dem das Docking des *Toolstrips* innerhalb des Formulars möglich gemacht wird. Anschließend können Sie über den Designer kon-

die bestimmt, in welchem Format Daten in die Textbox eingegeben werden können. Über das *SmartTag* des Steuerelementes und den darauf folgenden Menüeintrag *Set Mask* können Sie anschließend die Maske definieren, die zur Eingabe verwendet werden soll, wie es Abbildung 5 zeigt.

Beim *LinkLabel*-Steuerelement handelt es sich ebenfalls um ein Steuerelement, das unter Windows Forms 2.0 neu dazugekommen ist. Im Prinzip ist es ein ganz normales Label, das in Form eines Links dargestellt ist. Wenn aber der Benutzer auf das *LinkLabel* klickt, wird das Ereignis *LinkClicked* ausgelöst, auf das im Code entsprechend reagiert werden kann. Möchten Sie in Ihrer Anwendung professionelle ToolTips realisieren, können Sie auf das neue *ToolTip*-Steuerelement zurückgreifen. Dieses Steuerelement hat viele neue Optionen, unter anderem kann der Tooltip in Form eines Ballons dargestellt werden, wie es Abbildung 6 zeigt.

Dazu müssen Sie die Eigenschaft *IsBalloon* auf *True* einstellen. Außerdem können über die Eigenschaften *ToolTipTitle* und *ToolTipIcon* ein Titel und ein Icon für den Tooltip festgelegt werden, die verwendet werden, sobald der Tooltip auf einem Steuerelement aktiviert wird.

Ein weiteres neues Steuerelement, auf das jeder bereits sehr lange gewartet hat, ist das *WebBrowser*-Steuerelement. Über dieses Steuerelement sind Sie in der Lage, den Internet Explorer direkt in Ihre eigene Anwendung zu integrieren. Das ist zum Beispiel dann sehr von Vorteil, wenn Sie in Ihrer Windows-Anwendung auf Teile einer ASP.NET-Anwendung zugreifen möchten.

Bei diesem Steuerelement handelt es sich um eine Wrapper-Klasse um die COM-Komponente des Internet Explo-

riers. Diese COM-Komponente konnte bereits auch unter dem .NET Framework 1.x verwendet werden. Die neue Wrapper-Klasse bietet aber eine Vielzahl von neuen Features an, die einem bei der direkten Verwendung der COM-Komponente verwehrt bleiben. Dieses Steuerelement können Sie ebenfalls über eine Vielzahl von neuen Eigenschaften und Ereignissen an Ihre eigenen Bedürfnisse und Anforderungen anpassen.

Dieses Steuerelement bietet auch Methoden wie *Navigate*, *GoForward*, *GoBack* und weitere für die Navigation an. Über die Eigenschaft *ObjectForScripting* können Sie ebenfalls eine .NET-Klasse für den Scripting-Code einer geladenen Seite zugänglich machen. Dadurch ist ein leistungsfähiger Datenaustausch zwischen dem Internet Explorer und der Windows-Anwendung möglich. Außerdem bietet Ihnen das Steuerelement viele Features an, die Sie entsprechend Ihren Vorstellungen und Wünschen anpassen können. Zum Beispiel können Sie die Standardmenüeinträge oder Kontextmenüs um-

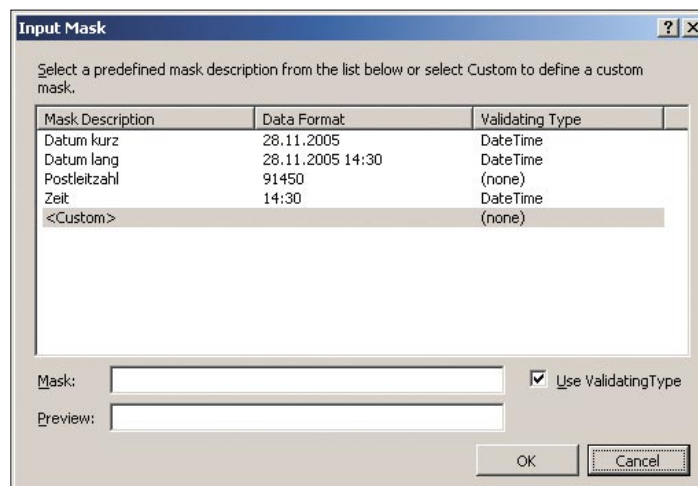


Abbildung 5
Eingabefor-
mat für eine
MaskedTextBox festlegen.



Abbildung 6 Ein ToolTip als Sprechblase.

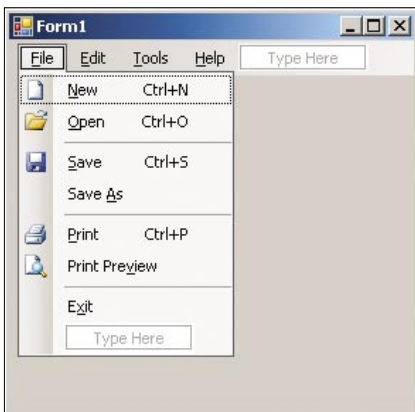


Abbildung 7 Standard-Menüeinträge einfügen.

figurieren, an welchen Stellen der *ToolStrip* innerhalb des Formulars angedockt werden kann.

Datenzugriff

Neben den neuen und erweiterten Steuerelementen hat sich in Windows Forms 2.0 auch einiges in Bezug auf den Datenzugriff getan. Unter dem .NET Framework 1.x war dieser sehr rudimentär über das *DataSet* und den *DataAdapter* implementiert worden und bot auch fast keine Erweiterungsmechanismen an.

Diese Vorgehensweise hat sich jetzt aber unter Windows Forms 2.0 grundlegend geändert. Daher möchte ich im nächsten Abschnitt zeigen, welche neuen Features Ihnen im Bereich des Datenzugriffs bereit stehen.

Als Erstes muss gleich erwähnt werden, dass der alte *DataSet Designer* komplett aus Visual Studio entfernt und durch einen neuen leistungsfähigen Designer ersetzt wurde. Der große Vorteil beim neuen *DataSet Designer* ist auch, dass hier kein XSD-Schema-Designer mehr benötigt wird, sondern das komplette Da-

taSet auf grafischem Weg erstellt werden kann. Dadurch brauchen Sie sich keine Gedanken mehr über XSD-Spezifika zu machen.

Im Hintergrund des *DataSet* arbeitet aber trotzdem immer noch ein XSD-Schema, welches die Struktur des *DataSet* darstellt. Der neue *DataSet Designer* verwendet Partial Classes, damit generierter Code bei einer Änderung neu generiert werden kann, ohne dass Änderungen von Ihnen verloren gehen. Über verschiedene Assistenten besteht für Sie ebenfalls die Möglichkeit, den generierten Code an Ihre Bedürfnisse anzupassen.

Ein weiteres Highlight des *DataSet Designers* bilden die *Typed Table Adapter*. Damit ist es möglich, dass für jedes SELECT-, INSERT-, UPDATE- und DELETE-Statement eine entsprechende Methode mit den notwendigen Parametern auf dem *TableAdapter* erzeugt wird. Dadurch ersparen Sie sich einiges an Tipparbeit und senken somit auch die Fehlerquote.

Der Server Explorer wird ebenfalls vom *DataSet Designer* unterstützt, das heißt, Sie können direkt Tabellen aus einer Datenbank auf den *DataSet Designer* ziehen, um den entsprechenden Datenzugriffscod zu generieren, der von der Anwendung benötigt wird.

Wenn Sie anschließend zum *TableAdapter* eine neue Abfragemethode hinzufügen möchten, brauchen Sie über der *DataTable* nur mit der rechten Maustaste zu klicken und aus dem Kontextmenü den Befehl *Add/Query* auswählen. Dadurch wird der in Abbildung 8 gezeigte *TableAdapter Query Configuration Wizard* aufgerufen.

Über diesen Assistenten können Sie anschließend eine SELECT-, INSERT-, UPDATE- oder DELETE-Abfrage erstellen

und dieser einen entsprechenden Namen geben, aus der anschließend eine Methode mit den entsprechenden Parametern generiert wird, die Sie direkt im Code verwenden können.

Ein weiteres wichtiges Konzept beim Datenzugriff, das in Windows Forms 2.0 komplett überarbeitet und dadurch einfacher gemacht wurde, ist das *Databinding* – also das Binden von Daten an Steuerelemente. Für das rundum erneuerte *Databinding-Feature* steht Ihnen das *Data Source Window* zur Verfügung, über das Sie die verschiedenen Datenquellen konfigurieren können, die Sie in Ihrer Anwendung verwenden möchten.

Als Datenquelle können Sie nicht nur eine Datenbank verwenden, sondern auch ein Business-Objekt oder eine *WebService-Proxyklasse*. Dadurch werden hier nun Szenarien unterstützt, die mit Visual Studio .NET 2003 noch nicht so einfach realisierbar waren.

Für jede Datenquelle, die Sie über das *Data Source Window* hinzufügen, wird ein eigenes *DataSet* erstellt, das Sie wiederum über den *DataSet Designer* bearbeiten können. Um die Leistungsfähigkeit des neuen *Databinding-Frameworks* zu demonstrieren, können Sie ganz einfach eine Datenquelle aus dem *Data Source Window* direkt auf ein Formular ziehen.

Dadurch werden alle notwendigen Steuerelemente erstellt und alle *Databinding-Einstellungen* entsprechend vorgenommen. Hierbei haben Sie die Möglichkeit, dass ein *DataGridView*-Steuerelement oder eine *Detailansicht* mit den entsprechenden Steuerelementen erstellt wird. Beide Möglichkeiten werden in Abbildung 9 dargestellt.

Die gewünschte Option können Sie durch Klick auf den *TableAdapter* im *Data Sources Window* festlegen. Sobald Sie das

Tabelle 1

Werte für AutoCompleteSource.

Option	Beschreibung
AllSystemSources	Entspricht der Kombination von <i>AllUrl</i> und <i>FileSystem</i> .
AllUrl	Entspricht der Kombination von <i>HistoryList</i> und <i>RecentlyUsedList</i> .
CustomSource	Hierbei können Sie eine benutzerdefinierte Datenquelle verwenden.
FileSystem	Als Datenquelle wird das Dateisystem herangezogen.
FileSystemDirectories	Als Datenquelle wird die Verzeichnisstruktur herangezogen.
HistoryList	Als Datenquelle wird der Verlauf des Internet Explorers herangezogen.
ListItems	Wenn Sie die Combobox verwenden, können deren Items als Datenquelle hinterlegt werden.
None	Hierbei wird keine Datenquelle verwendet. Damit ist das <i>AutoComplete-Feature</i> deaktiviert.
RecentlyUsedList	Als Datenquelle werden die zuletzt verwendeten URLs herangezogen.

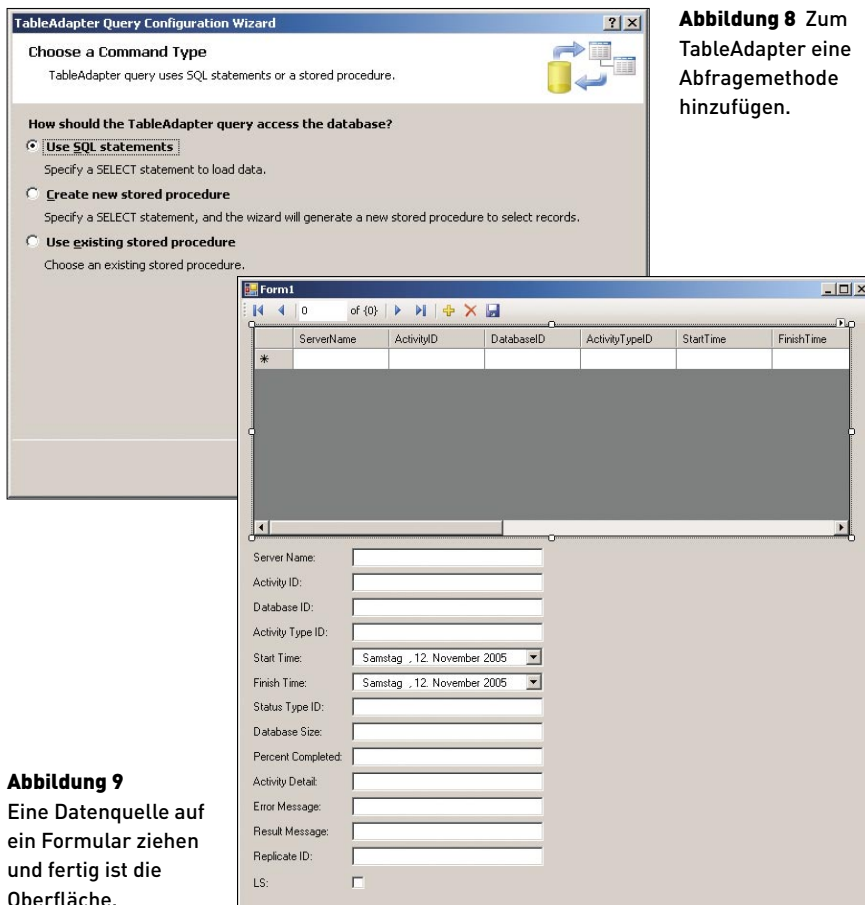


Abbildung 8 Zum TableAdapter eine Abfragemethode hinzufügen.

Abbildung 9 Eine Datenquelle auf ein Formular ziehen und fertig ist die Oberfläche.

Databinding durchgeführt haben, werden auch eine Reihe von unsichtbaren Steuerelementen angelegt, die sich in der Traybar des Formulars wiederfinden.

Das wichtigste und gleichzeitig auch das neueste Steuerelement ist *BindingSource*. Dieses Steuerelement ist für den kompletten Databinding-Mechanismus in Windows Forms 2.0 verantwortlich und kapselt auch den kompletten *CurrencyManager*, der für die Navigation zwischen den einzelnen Datensätzen verantwortlich ist.

Im ersten Schritt muss dieses Steuerelement über die Eigenschaft *DataSource* an eine Datenquelle gebunden werden. Dabei stehen die folgenden Auswahlmöglichkeiten zur Verfügung:

- DataSets,
- Business-Objekte,
- Web-Service-Proxyklassen.

Außerdem muss auch ein entsprechendes Member über die Eigenschaft *DataMember* konfiguriert werden, damit das Databinding überhaupt funktioniert. Über die *BindingSource*-Komponente wurde das alte Databinding-Framework von .NET 1.x gekapselt und in einem ein-

facheren Weg dem Benutzer bereitgestellt. Durch die Verwendung dieser Komponente verringert sich ebenfalls der Code, den ein Entwickler für das Databinding schreiben muss. Zum Beispiel werden automatisch im Ereignis *Load* des Formulars die gewünschten Daten in den *TableAdapter* geladen, damit mit ihnen direkt weitergearbeitet werden kann.

Über das *BindingSource*-Steuerelement können außerdem Features wie Sortierungen, Paging und Filterungen realisiert werden. All diese Funktionalitäten mussten unter .NET 1.x ausprogrammiert werden und stehen nun standardmäßig ohne zusätzlichen Code zur Verfügung.

Wie Sie bereits aus Abbildung 9 erkennen konnten, wird neben dem *BindingSource*-Steuerelement auch ein anderes Steuerelement dem Formular hinzugefügt, über das die Navigation zwischen den Datensätzen vorgenommen werden kann. Dabei handelt es sich um das *BindingNavigator*-Steuerelement.

Dieses Steuerelement bietet eine Reihe von Funktionen an, mit denen zwischen den verschiedenen Datensätzen navigiert werden kann. Zusätzlich können neue

Datensätze angelegt und bestehende Datensätze gelöscht werden. Dadurch können Sie datengetriebene Windows-Anwendungen nur mit ein paar Mausklicks realisieren und müssen nicht immer wieder den kompletten Infrastrukturcode neu erfinden. Das *BindingNavigator*-Steuerelement können Sie auch über eine Reihe von Eigenschaften entsprechend an Ihre Wünsche anpassen.

Ein weiteres Steuerelement, das zu Windows Forms 2.0 hinzugekommen ist, ist *DataGridView*, das eine Weiterentwicklung des *DataGrid*-Steuerelements aus .NET 1.x darstellt. Der große Unterschied zum *DataGrid*-Steuerelement ist, dass jetzt eine Vielzahl von Steuerelementen für Spalten unterstützt wird. Vorher mussten immer Drittkomponenten für diese Aufgaben zugekauft werden. Aktuell werden dabei unter anderem folgende Steuerelemente unterstützt:

- Text,
- Images,
- Buttons,
- LinkLabel,
- DropDownList.

Außerdem haben Sie nun auch endlich die Möglichkeit, jede Zelle innerhalb des *DataGridView*-Steuerelements individuell zu formatieren. Die komplette Formatierung können Sie wieder über verschiedene Assistenten direkt innerhalb von Visual Studio 2005 vornehmen.

Fazit

Wie Sie anhand dieses Artikels gesehen haben, sind in Windows Forms 2.0 eine Reihe von neuen leistungsfähigen Features dazugekommen. Das beginnt bei verschiedenen neuen Steuerelementen für die professionelle Oberflächengestaltung und setzt sich beim überarbeiteten Windows Forms Designer fort.

Große Änderungen und Erweiterungen hat es ebenfalls im Bereich des Datenzugriffs gegeben. Hierbei wurde die komplette Databinding-Architektur überdacht und in Form der neuen *BindingSource*-Komponente überarbeitet. Ich hoffe, mit diesem Artikel eine gute Einführung zu Windows Forms 2.0 gegeben zu haben. Vielleicht können Sie das eine oder andere Feature bereits in Ihrem nächsten Projekt einsetzen. |||||

[1] Weitere Infos zum Thema .NET 2.0 bietet auch der entsprechende Schwerpunkt in dotnetpro 10/2005 ab Seite 12.