„Die Autoren haben eng mit dem Microsoft ASP.NET-Team zusammengearbeitet, um zu garantieren, dass dieses Buch maßgeblich, präzise und informativ ist. Jeder, der ASP.NET einsetzt, findet eine Fülle an wertvollen Informationen zu ASP.NET 2.0."
—**Scott Guthrie,** General Manager, .NET Developer Platform, Microsoft Corporation

ADDISON-WESLEY

PEARSON Education

ASP.NET 2.0 Illustrated

Deutsche Ausgabe

Microsoft® .net™ Development Series

Alex Homer
Dave Sussman

ADDISON-WESLEY

3

Datenquellen-Steuerelemente und Datenanzeige-Steuerelemente

Im vorherigen Kapitel haben Sie erfahren, wie einfach es sein kann, datengestützte Webseiten schnell zu erstellen und sie in einer Site zu integrieren. Die erstellten Seiten enthalten viele Funktionalitäten, aber aufgrund ihrer Beschaffenheit, ihres Designs, haben sie einen festen Satz an Funktionen. Möglicherweise besitzen sie Funktionalitäten, die nicht ganz Ihren Anforderungen entsprechen. Die Grids bieten beispielsweise das Sortieren der Spalten an, aber was ist, wenn Sie die Funktion Sortieren gar nicht für alle Spalten brauchen? Oder was passiert, wenn Sie bestimmte Spalten auslassen, die Formatierung von anderen verändern oder die Methode ihrer Bearbeitung wechseln möchten? Allein die genannten Beispiele lassen erahnen, dass die Benutzerdefinierung der Daten-Steuerelemente umfangreich ist – und dieses Kapitel versucht, sie abzudecken und Ihnen zu zeigen, wie man das Beste aus den Datenquellen- und Grid-Steuerelementen herausholt.

Insbesondere erörtern wir, ...

- ... wie Sie mit der Datenbank Verbindung aufnehmen und wo Sie Verbindungsdetails speichern;
- ... wie man all die Datenquellen-Steuerelemente einsetzt, mit detaillierter Überprüfung der Elemente für die Datenbank und für XML-Daten (*Kapitel 4* behandelt `ObjectDataSource` ausführlich);
- ... wie Sie die Datenanzeige-Steuerelemente sowohl für die Darstellung als auch für die Bearbeitung von Daten nutzen.

Einige Datenanzeige-Steuerelemente werden wir ausführlicher behandeln als andere, da sie in ASP.NET 2.0 neu sind und mehr Funktionalität besitzen als die Steuerelemente, die es in Vorgängerversionen von ASP.NET gab. Lassen Sie uns mit der Untersuchung der Datenbanken und Datenverbindungen beginnen.

3.1 Datenbanken und Verbindungen

Die meisten Websites haben irgendeine Form von Datenbank im Hintergrund, sei es zum Speichern eines Produktkatalogs oder nur für die Verwaltung von Benutzer-Logins. Dies erfordert häufig eine vollständige Installation des SQL Servers auf Ihrem Entwicklungsrechner, jedoch gibt es mit ASP.NET 2.0 noch eine andere Option: SQL Server 2005 Express Edition (SSE). SSE ist eine eingeschränkte Version des SQL Server 2005, besitzt aber interessante Charakteristiken, die vor allem für die Entwicklungsumgebung nützlich sind. Zwei dieser Charakteristiken sind der automatische Datenbankanhang und die Nutzerinstanz.

3.1.1 Datenbanken automatisch anhängen

Ein Problem bei Entwicklungsapplikationen, die Datenbanken benutzen, ist die Notwendigkeit, einen SQL Server zu installieren. Sie müssen eine Datenbank anhängen, ein Login zum SQL Server erstellen und dann das Login der Datenbank hinzufügen. Das sollte auf jedem Entwicklungsrechner gemacht werden, es sei denn, der Nutzer ist als Administrator eingeloggt, was zwar häufig der Fall, aber nicht empfehlenswert ist.

SQL Server 2005 (sowohl die vollständige als auch die Express Edition) unterstützen automatisches Anhängen von Datenbanken, indem sie Folgendes im Verbindungsstring hinzufügen:

```
AttachDbFilename=db.mdf
```

Wenn die Anwendung startet, wird sich der SQL Server 2005 mit diesem Teil im Verbindungsstring automatisch an die Datenbank anhängen. Sie können explizit fest programmierte Pfade zur Datenbank vermeiden, indem Sie das neue *App_Data*-Verzeichnis und eine spezielle Funktion verwenden, um darauf zu verweisen. Betrachten Sie beispielsweise diesen Verbindungsstring:

```
Data Source=.\SQLEXPRESS; AttachDbFilename=|DataDirectory|db.mdf
```

ASP.NET platziert den speziellen String `|DataDirectory|` mit dem Pfad im *App_Data*-Verzeichnis und vermeidet somit die Notwendigkeit eines fest programmierten Pfads.

Obwohl sich Datenbanken dynamisch anhängen lassen, existiert immer noch das Problem der Berechtigung. Wie kann eine Datenbank automatisch angehängt

werden, wenn der Nutzer keine Berechtigung hat für das Anhängen von Datenbanken? Die Nutzerinstanz ist die Lösung des Problems und sie vermeidet, dass Nutzer gleichzeitig Administrator sein müssen.

3.1.2 Nutzerinstanz

Um Nutzerinstanzen zu verstehen, müssen Sie wissen, wie SQL-Serverinstanzen funktionieren. Als Standardeinstellung installiert SQL Express einen Instanznamen von SQLEXPRESS; so wird .\SQLEXPRESS in Verbindungsstrings verwendet, da Sie explizit auf eine Instanz verbinden. Der Standardnutzer, unter dem die Instanz bei Prozessstart ausgeführt wird und unter der SQL Express läuft, ist „NT AUTHORITY\NETWORK SERVICE“. So, wie es jetzt aussieht, würde immer noch ein explizites Login und Nutzererstellung gefordert werden.

Die Nutzerinstanz löst dieses Problem, indem eine Instanz auf Anforderung erstellt wird. Anstatt den Standardnutzer zu benutzen, verwendet eine Nutzerinstanz die Benutzerkennung als Servicekennung. Somit wird das Verfahren unter Beglaubigung des Nutzers ausgeführt und es sind keine expliziten Logins erforderlich. Auch benötigt man für das automatische Anhängen keine zusätzlichen Genehmigungen – der Besitzer des Verfahrens hat automatisch die vollständigen administrativen Rechte bezüglich der Datenbank, auch wenn der Nutzer nicht der Administrator auf dem Rechner ist. All das geschieht auf Anforderung, wenn die Anwendung startet. Demnach ist keine Konfiguration erforderlich. Das SQL-Server-Verfahren, das die Nutzerinstanz ausführt, bleibt 60 Minuten nach der letzten Verbindung aktiv, was sich jedoch mit der Option SP_CONFIGURE „Timeout“ konfigurieren lässt.

Sie können Nutzerinstanzen erlauben, indem Sie Folgendes in Ihrem Verbindungsstring ergänzen:

```
User Instance=True
```

Die Nutzerinstanz funktioniert nur mit integrierter Sicherheit, somit sieht der vollständige Verbindungsstring so aus:

```
Data Source=.\SQLEXPRESS;
AttachDbFilename=|DataDirectory|db.mdf;
User Instance=True;
Integrated Security=True;
```

Die Nutzerinstanz wurde für das Entwicklungsszenario entworfen, um eine Konfiguration zu vermeiden. Dem sind jedoch Grenzen gesetzt:

- Es sind nur lokale Verbindungen erlaubt.
- Replikation funktioniert nicht mit anderen Nutzerinstanzen.
- Verteilte Anfragen an Datenbanken im Netzwerk haben keinen Erfolg.
- Die Nutzerinstanz funktioniert nur in der SQL Server Express Edition.

Die folgenden üblichen Probleme erscheinen infolge von Nutzerinstanzen:

- Wenn die Entwicklungsumgebungen von Visual Studio 2005 oder Visual Web Developer zum Einsatz kommen, um zu SQL Express zu verbinden, werden die SQL-Express-Instanzen unter der Kennung „NT AUTHORITY\NETWORK SERVICE“ ausgeführt. Diese Kennung erfordert allerdings Schreibberechtigungen auf den Datenbankdateien *MDF* und *LDF*.
- ASP.NET-Seiten werden unter der ASPNET-Nutzerkennung ausgeführt. Das heißt, dass eine ASP.NET-Seite, die mit Nutzerinstanz zu einer Datenbank verbindet, in genau dem SQL-Expressverfahren endet, das der ASP.NET-Nutzer besitzt. Folglich benötigt er auch Schreibberechtigungen für die Datenbankdateien.
- Die Nutzerinstanz öffnet Datenbanken mit exklusivem Zugriff, um die Gefahr von Datenkorruption aufgrund verschiedener Nutzerinstanzen mit demselben Datenbanknamen zu reduzieren. Demnach ist es nicht möglich, die Datenbank in der Entwicklungsumgebung zu öffnen, während sie noch von der Applikation in Gebrauch ist. Das wird durch die Entwicklungsumgebung bis zu einem gewissen Grad erleichtert, da eine auszuführende Applikation (über F5) die Instanz automatisch schließt und sie von dem Tool trennt, so dass ASP.NET sie öffnen kann. Wenn Sie die Applikation außerhalb der Entwicklungsumgebung ausführen, wie beispielsweise von einem virtuellen Verzeichnis in IIS, dann ist das nicht der Fall und Sie müssen die Verbindung eventuell manuell schließen (verwenden Sie VERBINDUNG TRENNEN im Kontextmenü).
- Wenn Sie zu einer Nutzerinstanz-Datenbank verbinden, tun Sie dies als Administrator. Sobald Applikationen angewendet werden, sollten Sie mit einem niedriger privilegierten Nutzer zugreifen und müssen dann die Berechtigungen auf der Datenbank konfigurieren.

3.1.3 Platzierung des Verbindungsstrings

Verbindungsstrings werden typischerweise in der *Web.Config* gespeichert und in Version 1.x war dafür üblicherweise der Bereich `appSettings` zuständig. In ASP.NET 2.0 gibt es einen neuen `connectionStrings`-Bereich, der ein ähnliches Schlüssel/Werte-Paar bereitstellt, wie zum Beispiel:

```
<connectionStrings>
  <add name="AW"
    connectionString="Data Source=.\SQLEXPRESS; . . ."
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Obwohl das Attribut `providerName` nicht obligatorisch ist, erscheinen keine Verbindungsstrings in den Dialogfenstern DATENQUELLE KONFIGURIEREN, wenn der Provider-Name nicht festgelegt ist.

Innerhalb der Anwendungen können Sie auf Verbindungsstrings mit zwei Methoden zugreifen. Im Code benutzen Sie die `ConnectionStrings`-Eigenschaft des `ConfigurationManager`-Objekts:

```
SqlConnection conn = new SqlConnection();
conn.ConnectionString =
     ConfigurationManager.ConnectionStrings["AW"].ConnectionString;
```

Die `ConnectionStrings`-Eigenschaft enthält eine Sammlung an Verbindungsstrings aus dem Bereich im *Web.Config* und Sie verwenden die `Name`-Eigenschaft als Index für die Sammlung. Die `ConnectionString`-Eigenschaft liefert dann den eigentlichen Verbindungsstring zurück.

Als zweite Methode gibt es innerhalb der Beschreibungssprache von ASP.NET-2.0-Seiten neue hilfreiche Ausdrücke. Diese Ausdrücke ermöglichen es Ihnen, deklarativ auf Funktionen wie Verbindungsstrings, Applikationseinstellungen und Quellen zuzugreifen. Betrachten Sie beispielsweise den folgenden Code:

```
<asp:SqlDataSource id="SqlDataSource1" runat="server"
  ConnectionString="<%$ ConnectionStrings:AW %> "
```

Der Ausdruck verwendet ein serverseitiges <% %>-Tag. Sollte der erste Charakter innerhalb des Tags ein $-Zeichen sein, ist das ein Anzeichen dafür, dass ein Ausdruck verwendet werden muss. Jeder Ausdruckstyp hat ein bekanntes Präfix, für den Verbindungsstring ist es `ConnectionStrings`. Auf ähnliche Weise wie beim Code benutzen Sie das `name`-Attribut aus dem *Web.Config*-Bereich, um den erforderlichen Verbindungsstring zu identifizieren. Dabei verwenden Sie einen Doppelpunkt (:), um das Präfix vom Namen zu trennen.

Der Vorteil dieser beiden Methoden ist, dass Sie sowohl vom Code als auch aus der Beschreibungssprache die zentral gespeicherten Verbindungsstrings benutzen können.

3.1.4 Visual Web Developer und Visual Studio 2005 Server-Explorer

Sowohl Visual Studio 2005 als auch Visual Web Developer besitzen Datenbankmanagement-Funktionen und obwohl beide von unterschiedlichen Stellen aus zu erreichen sind, verhalten sie sich gleich. In Visual Studio 2005 verwenden Sie den Server-Explorer, der sich je nach Konfiguration auf der linken oder rechten Seite des Bildschirms befindet, wie in Abbildung 3.1 zu sehen ist. In Visual Web Developer kommt der Database Explorer zum Einsatz, der sich standardmäßig rechts auf dem Bildschirms befindet, wie aus Abbildung 3.2 hervorgeht. Wie Sie erkennen können, ist ihr Inhalt der gleiche und er ermöglicht Ihnen den Zugriff auf die Inhalte der Datenbank innerhalb des Entwicklungstools.

Welches Tool Sie auch immer benutzen, das Arbeiten mit Datenbanken ist einfach. Sie können Tabellen, gespeicherte Prozeduren, Ansichten und Funktionen erstellen bzw. modifizieren sowie Ad-hoc-Anfragen ausführen. In dem Buch beziehen wir uns auf den Database Explorer, weil er expliziter ist; falls Sie Visual Studio 2005 verwenden, dann benutzen Sie den Server Explorer, um auf die gleichen Funktionalitäten zuzugreifen.

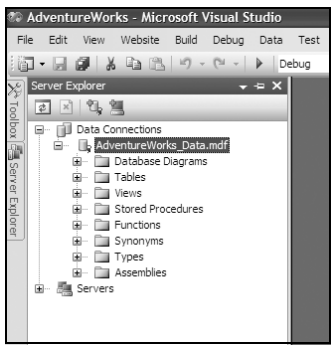


Abbildung 3.1: Der Server Explorer von Visual Studio 2005

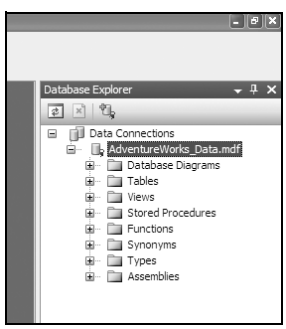


Abbildung 3.2: Der Database Explorer von Visual Web Developer

3.2 Datenquellen-Steuerelemente

Kapitel 2 hat gezeigt, wie einfach Datenquellen-Steuerelemente zu benutzen sind, aber es gehört mehr dazu als nur ihre einfache Verwendung. Datenquellen-Steuerelemente bieten eine deklarative Methode, um nicht nur die Verbindung zu einer Datenbank zu definieren, sondern auch die Kommandos, die für das Auslesen und Aktualisieren von Daten verwendet werden. Einige Datenquellen-Steuerelemente interagieren nicht direkt mit ihrer Datenquelle, sondern über einen Provider. Ein Provider abstrahiert die Funktionalität, die im Umgang mit der Datenquelle eingesetzt wird, und bietet eine konsistente API für verwendbare Applikationen und

Steuerelemente. Provider sind modular erweiterbar und erlauben es Ihnen, die bereitgestellten Provider durch Ihre eigenen zu ersetzen. Die folgenden vier Datenquellen-Steuerelemente werden als Standard mit ASP.NET 2.0 ausgeliefert:

- `ObjectDataSource` bietet die Schnittstelle zu eigenen Klassen.
- `SitMapDataSource` ist die Schnittstelle zu Sitemap-Daten für Site-Navigationen.
- `SqlDataSource` ist die Schnittstelle zu Microsoft SQL-Datenbanken.
- `XmlDataSource` bietet die Schnittstelle zu XML-Dateien.

Wir wollen nicht alle Steuerelemente in diesem Kapitel behandeln, da `ObjectDataSource` in *Kapitel 4* und `SiteMapDataSource` in *Kapitel 10* beschrieben werden. Jedoch sind viele Techniken der `SqlDataSource`- und `ObjectDataSource`-Steuerelemente ähnlich.

3.2.1 Das SqlDataSource-Steuerelement

Das `SqlDataSource`-Steuerelement bietet ein Zwei-Schichten-Modell für die Interaktion mit relationalen Datenbanken. Es entlastet den Entwickler davon, explizit Verbindungen und Kommandos zu erstellen, und gibt ihm Zeit, sich auf die Datenstatements zu konzentrieren. Wenn man die Funktionalität per Drag&Drop benutzt, wie im vorherigen Kapitel gezeigt, verwandeln sich die Statements in explizites SQL. Sie können aber auch gespeicherte Prozeduren verwenden. Es gibt keine Möglichkeit, gespeicherte Prozeduren automatisch zu erstellen und in der `SqlDataSource` zu verwenden, also müssen Sie das selbst machen. Eine schnelle Lösung besteht darin, Drag&Drop zu benutzen und dann die SQL-Statements als SQL für Ihre gespeicherten Prozeduren einzusetzen.

Um ein `SqlDataSource`-Steuerelement zu konfigurieren, können Sie die Eigenschaften manuell im Code bearbeiten oder die Aufgabenoption DATENQUELLE KONFIGURIEREN benutzen. Letzteres begleitet Sie einfach durch die Auswahl der Verbindung (entweder eine Verbindung vom *Web.Config* verwenden oder selbst eine erstellen). Dabei werden Tabellen und Spalten gesucht, aus denen Daten abgerufen werden sollen. Optional lassen sich Filter und vollständige Parallelität an die Kommandos anhängen. Wir werden diese Themen im Abschnitt „Daten filtern“ und „Daten aktualisieren“ betrachten.

Das `SqlDataSource`-Steuerelement besitzt eine Anzahl von Eigenschaften, die sein Verhalten steuern. Die erste Eigenschaft ist der `DataSourceMode`, der entweder auf `DataSet` oder `DataReader` gesetzt sein sollte, um zu erkennen, wie die Daten aus der zugrunde liegenden Datenquelle beschafft werden. Der Standard ist `DataSet` und er unterstützt zweiseitige Verbindung, während `DataReader` mehr Performance bietet, aber nur das Lesen Daten erlaubt.

Um zu kontrollieren, wie oft die Daten abgerufen werden, gibt es eine Anzahl an Eigenschaften, spezifisch für das Cachen. Sie ermöglichen es dem Steuerelement, Daten zu cachen und sie nur wieder abzurufen, wenn sie sich seit dem letzten Zugriff verändert haben. Mehr dazu lesen Sie in *Kapitel 6*.

Die Eigenschaften, die Sie häufiger benutzen, beziehen sich auf die verwendeten SQL-Kommandos, um Daten abzurufen und zu verändern. Für jeden Typ von Datenmanipulation gibt es zwei Eigenschaften: ein Kommando für die Ausführung und ein Kommandotyp, um zu identifizieren, ob das Kommando direkt in SQL steht oder eine gespeicherte Prozedur aufruft.

3.2.2 Daten mit einem GridView-Steuerelement anzeigen

In seiner einfachsten Form erfordert das `SqlDataSource`-Steuerelement nur zwei gesetzte Eigenschaften, um Daten aus der Datenbank abzurufen: den `ConnectionString` und das `SelectCommand`. Ein Beispiel:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
  ConnectionString="<%$ ConnectionStrings:NorthwindConnectString %>"
  SelectCommand="SELECT * FROM Products" />
```

Der Code ruft alle Zeilen der `Products`-Tabelle auf. Um zu den gespeicherten Prozeduren zu wechseln, ersetzen Sie das integrierte SQL-Statement mit dem Namen der gespeicherten Prozedur und fügen das `SelectCommandType`-Attribut hinzu und setzen seinen Wert auf `StoredProcedure`.

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
  ConnectionString="<%$ ConnectionStrings:NorthwindConnectString %>"
  SelectCommand="usp_Products"
  SelectCommandType="StoredProcedure" />
```

Sie können dieses Datenquellen-Steuerelement verwenden, indem Sie ein anderes Steuerelement, wie beispielsweise ein `GridView`, damit verbinden:

```
<asp:GridView ID="GridView1" runat="server"
  DataSourceID="SqlDataSource1" />
```

Hier wird die `DataSourceID`-Eigenschaft verwendet, um die `ID`-Eigenschaft des Datenquellen-Steuerelements zu identifizieren, das die Daten liefert. Dabei handelt es sich um eine für alle Datenquellen-Steuerelemente verfügbare Funktionalität. Als Standard zeigt das `GridView` alle Spalten der zugrunde liegenden Datenquelle an. Dies lässt sich aber ausschalten, indem die `AutoGenerateColumns`-Eigenschaft auf `False` gesetzt wird und die Spalten explizit definiert werden. Darüber diskutieren wir ausführlich im Abschnitt über eigene Anpassungen des `GridView`-Steuerelements (*Abschnitt 3.3.2* in diesem Kapitel).

3.2.3 Daten filtern und auswählen

Eine der praktischsten Funktionen der neuen Daten-Steuerelemente ist die Unterstützung für parametrisierte Anfragen. Ihnen steht die Möglichkeit offen, Daten in Abhängigkeit von externen Kriterien, wie beispielsweise anderer Steuerelemente oder Anfragestring, zu filtern. Sie können Daten in der Datenbank filtern, indem Sie `SelectParameters` benutzen oder auf der Webseite, nachdem die Daten abgerufen wurden, `FilterParameters` einsetzen. Der Unterschied ist, dass `SelectParameters` und WHERE-Klauseln eingesetzt werden, bevor die Daten abgerufen werden, während `FilterParameters` die Daten abruft und sie dann filtert. Es gibt keine sich gegenseitig ausschließenden Vorgänge und beide können gleichzeitig verwendet werden, falls nötig. Selektieren ist hilfreich, wenn die gesuchten Datenmengen groß sind, weil nur eine Untermenge der Daten von der Datenbank zur Seite zurückgesendet wird. Filtern ist sinnvoll, wenn die Daten gecacht sind, weil es die Datenbankzugriffe reduziert.

Parameter filtern und auswählen

Es gibt sieben Datenquellen bzw. Parameter, die zum Selektieren und Filtern verwendet werden können:

- `ControlParameter`, der seine Daten aus einem ASP.NET Server-Steuerelement übernimmt,
- `CookieParameter`, der seine Daten aus einem Cookie bezieht,
- `FormParameter`, der seine Daten aus einem HTML-Steuerelement ableitet,
- `Parameter`, der die Basisklasse für die anderen Parameter ist und keine Standardquellen an Daten besitzt,
- `ProfileParameter`, der seine Daten aus einer Profil-Eigenschaft nimmt,
- `QueryStringParameter`, der seine Daten aus einem Anfragestringwert ableitet,
- `SessionParameter`, der seine Daten aus einem Sessionwert bezieht.

Diese Parameter bieten sehr viel Flexibilität bezüglich des deklarativen Filterns von Daten. Sämtliche Parameter unterstützen die in Tabelle 3.1 aufgelisten Eigenschaften.

Zusätzlich zu den Eigenschaften besitzt jeder Parametertyp seine eigenen spezifischen Eigenschaften, wie Sie in Tabelle 3.2 erkennen können.

Eigenschaft	Beschreibung
`ConvertEmptyStringToNull`	Zeigt an, ob ein leerer Wert auf `0` gesetzt werden soll oder nicht, bevor er an den SQL Server weitergereicht wird. Der Standardwert ist `True`.
`DefaultValue`	Zeigt den Standardwert des Parameters an, wenn keiner von einer externen Quelle gestellt wird.
`Direction`	Zeigt die Richtung des Datenflusses an und kann einer der `ParameterDirection`-Werte sein: • `Input` für Daten, die zum SQL Server übertragen werden • `InputOutput` für Daten, die zum und vom SQL Server übertragen werden • `Output` für Daten, die vom SQL Server übertragen werden • `ReturnValue` für Daten, die vom SQL Server als Rückgabewert einer Anfrage übertragen werden
`Name`	Der Name des Parameters
`Size`	Die Größe des Parameters. Nicht erforderlich für feste Größentypen.
`Type`	Der Datentyp des Parameters. Kann einer der `TypeCode`-Werte sein: `Boolean`, `Byte`, `Char`, `DateTime`, `DBNull`, `Decimal`, `Double`, `Empty`, `Int16`, `Int32`, `Int64`, `Object`, `SByte`, `Single`, `String`, `UInt16`, `UInt32`, `UInt64`

Tabelle 3.1: Gewöhnliche Eigenschaften aller Parametertypen

Sind Parameter im Einsatz, funktionieren sie alle auf dieselbe Art und Weise. Die Daten, die beim Selektieren oder Filtern Verwendung finden, werden der Quelle entnommen, die der Parameter spezifiziert hat. Für einen `ControlParameter` identifiziert die ID also das Steuerelement und `PropertyName` die Eigenschaft, die die Daten speichert. Wenn zum Beispiel eine `TextBox` als Quelle der Parameterdaten benutzt wird, kommt die `Text`-Eigenschaft als `PropertyName` in Frage. Einfache Steuerelemente, wie eine `TextBox`, sind nicht die einzige Quelle für Parameterdaten; ein `GridView` könnte das `SelectedValue` als `PropertyName` einsetzen, welches das Schlüsselfeld (aus der `DataKeyNames`-Eigenschaft) für den Parameterwert nehmen würde.

Parameter	Eigenschaft	Beschreibung
`ControlParameter`	`ControlID`	Die ID des Steuerelements liefert die Parameterdaten.
	`PropertyName`	Der Name der Eigenschaft auf dem Steuerelement, zum Beispiel die `Text`-Eigenschaft
`CookieParameter`	`CookieName`	Der Name des Cookies liefert die Parameterdaten.
`FormField`	`FormParameter`	Der Name des Formfelds als Parameterdaten
`ProfileProperty`	`PropertyName`	Der Name der `Profil`-Eigenschaft liefert die Parameterdaten.
`QueryStringParameter`	`QueryStringField`	Der Name des Anfragestringfelds bestimmt die Parameterdaten.
`SessionParameter`	`SessionField`	Der Name des Sessionwerts liefert die Parameterdaten.

Tabelle 3.2: Spezielle Parametertyp-Eigenschaften

Parameter lassen sich entweder mit dem Assistenten konfigurieren oder deklarativ in der Quellenansicht (die der Assistent generiert) oder im Code.

Daten mit der Konfiguration über den Assistenten auswählen

Die Konfiguration über den Assistenten ist sehr einfach zu benutzen, aber es lohnt sich, die wichtigsten Bereiche näher zu betrachten, die die Parametrisierung betreffen. Der erste Bereich (siehe Abbildung 3.3) zeigt die WHERE-Schaltfläche, die bei Klick ein Fenster öffnet, in dem die Konfiguration der Parameter möglich ist.

In Abbildung 3.4 können Sie auf der linken Seite des Fensters erkennen, dass drei Schlüsselinformationen erforderlich sind:

- die Spalte, welche die Spalte in den Daten ist, die Sie filtern (das ist der Spaltenname, der in der WHERE-Klausel dem SQL hinzugefügt ist);
- der Operator, der unterschiedliche Vergleichsoptionen zur Verfügung stellt, wie „ist gleich“, „größer als“ und „LIKE“;
- die Quelle, welche die Quelle des Werts ist und mit den Parametertypen übereinstimmt.

Abbildung 3.5 zeigt eine Beispielkonfiguration für das Filtern: In diesem Fall können Sie erkennen, dass die TEXTBOX1 gewählt ist, und Sie sehen den SQL-Ausdruck, der dem Code hinzugefügt wird. Das Klicken auf HINZUFÜGEN erstellt die WHERE-Klausel.

Der Assistent bietet nur eine Methode, um `SelectParameters` auf dem Steuerelement visuell festzusetzen; bei Bedarf können Sie das auch manuell.

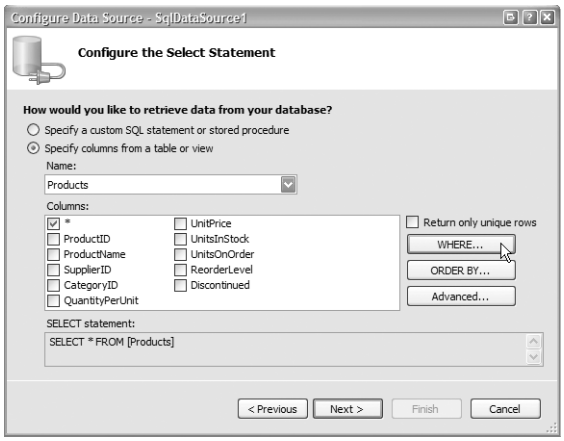


Abbildung 3.3: Eine WHERE-Klausel dem SqlDataSource-Steuerelement hinzufügen

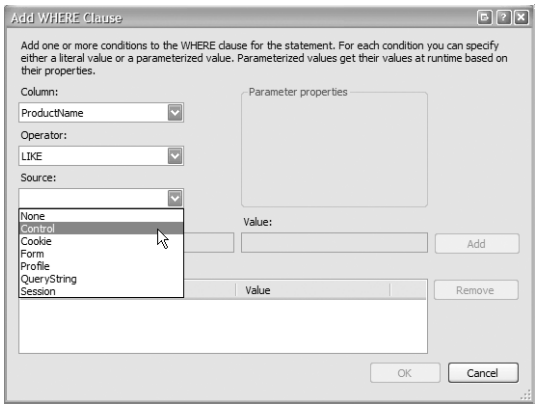


Abbildung 3.4: Den Parametertyp auswählen

Daten deklarativ auswählen

Um Daten deklarativ auszuwählen, benutzen Sie `SelectParameters` aus der Datenquelle, wie in Listing 3.1 demonstriert:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
  ConnectionString="<%$ ConnectionStrings:NorthwindConnectString %>"
  SelectCommand="SELECT * FROM [Products] WHERE
                 ([ProductName] LIKE '%' + @ProductName + '%')">
  <SelectParameters>
    <asp:ControlParameter Name="ProductName" Type="String"
      ControlID="TextBox1" PropertyName="Text" />
  </SelectParameters>
  </asp:SqlDataSource>
```

Listing 3.1: SelectParameters verwenden

Listing 3.1 zeigt das Ergebnis der Konfiguration eines `SqlDataSource`-Steuerelements mit dem Assistenten. Er hat hier die WHERE-Klausel hinzugefügt.

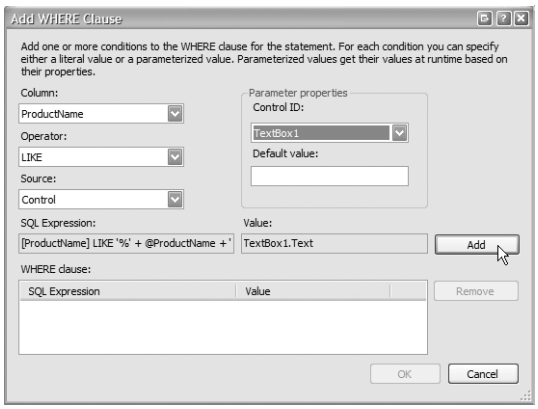


Abbildung 3.5: Die Quelle der Parameterdaten auswählen

Beachten Sie, dass der Parameter in der WHERE-Klausel genauso behandelt wird wie ein Standard-SQL-Server-Parameter, nämlich mit einem vorangestellten @-Zeichen. Achten Sie auch darauf, dass die `Name`-Eigenschaft des `ControlParameter` dem Parameternamen in der Anfrage entspricht. Sie können einen beliebigen Wert für den Namen der Parameter verwenden, aber er muss dem aktuellen Parameter entsprechen; ihn gleich zu benennen wie den Spaltennamen, vereinfacht das Lesen des Codes.

Weitere Parameter können nach Bedarf hinzugefügt werden, wie aus Listing 3.2 hervorgeht.

Hier gibt es drei Parameter: Der erste bleibt wie zuvor gezeigt unverändert, während der zweite nur den Zeilen entspricht, in denen der `UnitPrice` unter einem bestimmten Wert liegt; der dritte Parameter entspricht der Spalte `Discontinued`. `SelectCommand` folgt immer noch dem Standard-SQL-Format, aber jeder Parameter hat Zugriff auf die `SelectParameters`. Die ersten beiden entnehmen ihre Werte aus den Steuerelementen, der erste aus der `Text`-Eigenschaft einer `TextBox`, der zweite aus dem `SelectedValue` einer Liste. Dies demonstriert, dass Sie einen Wert aus einer beliebigen Eigenschaft eines Steuerelements wählen können.

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
  ConnectionString="<%$ ConnectionStrings:NorthwindConnectString %>"
  SelectCommand="SELECT * FROM [Products] WHERE
                 ([ProductName] LIKE '%' + @ProductName + '%')
                 AND UnitPrice < @UnitPrice
                 AND Discontinued = @Discontinued">
  <SelectParameters>
    <asp:ControlParameter Name="ProductName" Type="String"
      ControlID="TextBox1" PropertyName="Text" />
    <asp:ControlParameter Name="UnitPrice" Type="Decimal"
      ControlID="PriceList" PropertyName="SelectedValue" />
```

```
    <asp:QueryStringParameter Name="Discontinued" Type="Boolean"
      QueryStringField="Discontinued" />
  </SelectParameters>
  </asp:SqlDataSource>
```

Listing 3.2: Verschiedene SelectParameters

Der dritte Parameter holt seinen Wert aus dem `QueryString` und zeigt somit, dass Sie unterschiedliche Typen von Parametern in der gleichen Anfrage mischen dürfen.

Das Filtern funktioniert ein wenig anders: `SelectCommand` wird nicht modifiziert. Stattdessen wird eine `FilterExpression`-Eigenschaft hinzugefügt und `FilterParameters` anstelle von `SelectParameters` verwendet (vergleiche Listing 3.3):

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
  ConnectionString="<%$ ConnectionStrings:NorthwindConnectString %>"
  SelectCommand="SELECT * FROM [Products]"
  FilterExpression="ProductName LIKE '{0}%'">
  <FilterParameters>
    <asp:ControlParameter Name="ProductName" Type="String"
      ControlID="TextBox1" PropertyName="Text" />
  </FilterParameters>
</asp:SqlDataSource>
```

Listing 3.3: FilterExpression und FilterParameters verwenden

Beachten Sie, dass Parameter in `FilterExpression` nicht explizit benannt sind – sie sind positionierbar. Also bezieht sich {0} auf den ersten Parameter in `FilterParameters`, weitere Parameter wären dann {1}, {2}usw.

Daten per Code auswählen

Filtern und Auswahl erfordern keinen Code, wenn sie deklarativ verwendet werden. Sie bieten eine einfache Methode, um datengebundene Steuerelemente mit anderen Steuerelementen oder anderen Quellen von Filterdaten zu verbinden. Leider können Sie nicht direkt innerhalb des Codes filtern oder auswählen, obwohl es Ereignisse gibt, die es Ihnen gestatten, Parameterwerte zu modifizieren, bevor eine Aktion stattfindet. Das `SqlDataSource`-Steuerelement hat eine `Select`-Methode, aber die führt nur Sortieren und Paging aus.

3.2.4 Daten aktualisieren

Für Datenaktualisierungen bietet das `SqlDataSource`-Steuerelement ein ähnliches Modell wie das bei der Datenauswahl, bei dem es ein `SelectCommand`, einen `SelectCommandType` und `SelectParameters` gibt. Bei der Modifizierung von Daten sind die Eigenschaften aus Tabelle 3.3 behilflich.

Wie die Auswahlfunktionen können sie manuell oder über den Assistent konfiguriert werden. Für Letztere wählen Sie die ERWEITERT-Option, wenn Sie Ihre Tabelle oder Anfrage auswählen. Das erscheinende Fenster (siehe Abbildung 3.6) ermöglicht es Ihnen, die Datenmodifizierungs-Statements automatisch der Daten-

quelle hinzuzufügen. Sie haben zudem die Möglichkeit, mithilfe der vollständigen Parallelität bei jeder modifizierenden Spalte eine Spaltenprüfung anzubringen. Dies gewährleistet, dass alle Daten nur aktualisiert werden, wenn alle Spalten die gleichen sind, und es verhindert, dass Sie Daten überschreiben, die jemand anderes aktualisiert hat. *Kapitel 5* befasst sich eingehender mit diesem Thema.

Ist die Konfiguration einmal abgeschlossen, sind die Kommandos und Parameter festgesetzt. Sehen Sie sich beispielsweise Listing 3.4 an. Die Datenquelle wurde mit nur drei Spalten konfiguriert, um die Übersichtlichkeit zu wahren.

Jedes der Kommandos ist ein normales SQL-Statement und Sie können erkennen, wie die Parameter entsprechend zusammenpassen. Für das `DeleteCommand` wird nur ein einziger Parameter `ProductID` benötigt, weil er die Zeile eindeutig identifiziert. Die Information wird automatisch aus den Schlüsselwertfeldern der Datenbank entnommen. Wenn Sie also eine Tabelle haben, deren eindeutiger Schlüssel mehrere Spalten sind, so sind auch mehrere `DeleteParameters` erforderlich.

Aktion	Eigenschaft	Beschreibung
Neue Daten hinzufügen	`InsertCommand`	Das Kommando führt das Einfügen einer einzelnen Zeile in der Datenbank aus.
	`InsertCommandType`	Der Kommandotyp, der innerhalb der `InsertCommand`-Eigenschaft enthalten ist. Kann einer der `SqlDataSourceCommandType`-Werte sein: `StoredProcedure` oder `Text`.
	`InsertParameters`	Die Parametersammlung, die Daten zum Einfügen enthält
Bestehende Daten aktualisieren	`UpdateCommand`	Das Kommando führt die Aktualisierung einer einzelnen Zeile in der Datenbank aus.
	`UpdateCommandType`	Der Kommandotyp, der innerhalb der `UpdateCommand`-Eigenschaft enthalten ist. Kann eine der `SqlDataSourceCommandType`-Aufzählungen sein: `StoredProcedure` oder `Text`.
	`UpdateParameters`	Die Parametersammlung, die Daten zum Aktualisieren enthält
Daten löschen	`DeleteCommand`	Das Kommando führt das Löschen einer einzelnen Zeile in der Datenbank aus.
	`DeleteCommandType`	Der Kommandotyp, der innerhalb der `DeleteCommand`-Eigenschaft enthalten ist. Kann eine der `SqlDataSourceCommandType`-Aufzählungen sein: `StoredProcedure` oder `Text`.
	`DeleteParameters`	Die Kommandosammlung, die die Daten zum Löschen enthält

Tabelle 3.3: Eigenschaften für Datenmodifikation

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
  ConnectionString="<%$ ConnectionStrings:NorthwindConnectString %>"
  SelectCommand="SELECT [ProductID], [ProductName], [UnitPrice]
                 FROM [Products]
                 WHERE ([ProductName] LIKE '%' + @ProductName + '%')"
  DeleteCommand="DELETE FROM [Products]
                 WHERE [ProductID] = @ProductID"
  InsertCommand="INSERT INTO [Products] ([ProductName], [UnitPrice])
                 VALUES (@ProductName, @UnitPrice)"
  UpdateCommand="UPDATE [Products]
                 SET [ProductName] = @ProductName,
                     [UnitPrice] = @UnitPrice
                 WHERE [ProductID] = @ProductID">
  <SelectParameters>
    <asp:ControlParameter ControlID="TextBox1"
      Name="ProductName" PropertyName="Text"
      Type="String" />
  </SelectParameters>
  <DeleteParameters>
    <asp:Parameter Name="ProductID" Type="Int32" />
  </DeleteParameters>
  <UpdateParameters>
    <asp:Parameter Name="ProductName" Type="String" />
    <asp:Parameter Name="UnitPrice" Type="Decimal" />
    <asp:Parameter Name="ProductID" Type="Int32" />
  </UpdateParameters>
  <InsertParameters>
    <asp:Parameter Name="ProductName" Type="String" />
    <asp:Parameter Name="UnitPrice" Type="Decimal" />
  </InsertParameters>
</asp:SqlDataSource>
```

Listing 3.4: Ein komplett konfiguriertes SqlDataSource-Steuerelement

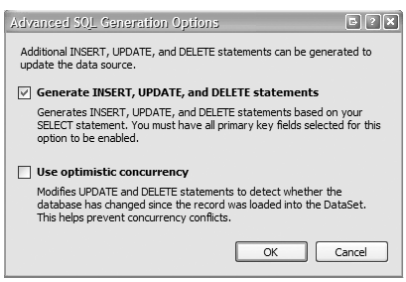


Abbildung 3.6: Datenmodifikationen zum SqlDataSource-Steuerelement hinzufügen

Für das `UpdateCommand` gibt es drei Attribute: `ProductName` und `UnitPrice`, die aktualisiert sind, sowie `ProductID` mit der Aufgabe, die zu aktualisierende Zeile zu identifizieren. Beachten Sie, dass die Parameter, obwohl sie unterschiedliche Vorgänge im Kommando vornehmen, innerhalb des `UpdateParameter` auf gleiche Weise definiert sind.

Für das `InsertCommand` und den dazugehörigen Parameter sind nur zwei Attribute erforderlich – beide definieren die Daten, die einzufügen sind.

Abbildung 3.7 zeigt, wie die <asp: Parameter/>-Objekte mit den Kommandoparametern zusammenhängen.

Man sollte beachten, dass für Datenaktualisierungen die über den Assistenten erstellten Parameter die Basisparametertypen sind, da die Daten aus der zugrunde liegenden Datenquelle kommen. Wenn das Steuerelement, das an die Datenquelle gebunden ist, ein Kommando aussendet, werden die geeigneten Werte aus dem Steuerelement in den Parameter geschoben und das Kommando wird ausgeführt.

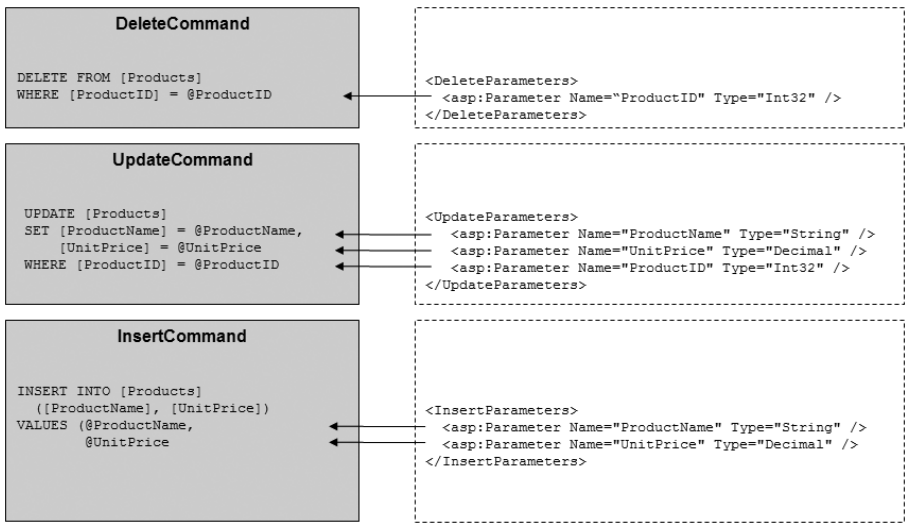


Abbildung 3.7: Die Parameter und das SQL-Kommando

3.3 Datenanzeige- und Bearbeiten-Steuerelemente

Im vorherigen Kapitel haben Sie gesehen, wie einfach es ist, ein Grid zu konfigurieren, um ein Datenquellen-Steuerelement nicht nur für die Ansicht der Daten zu verwenden, sondern auch um eine Bearbeitungsmöglichkeit zu bieten. An dieser Stelle wollen wir ausführlicher darauf eingehen, indem wir Ihnen anhand der folgenden Szenarien zeigen, ...:

- ... wie Sie Steuerelemente mit Datenquellen verbinden,
- ... welche Steuerelemente für die Datenanzeige verwendet werden,
- ... wie man Benutzerbearbeitung der Daten erlaubt,
- ... wie Sie Anzeige-Steuerelemente benutzerdefiniert anpassen.

Die Methoden bei der Abwicklung dieser Aufgaben sind für viele Steuerelemente ähnlich, somit werden Sie es einfach finden, zwischen ihnen zu wechseln.

3.3.1 Datenverbindung zu Datenquellen-Steuerelementen

Zusammen mit dem deklarativen Stil der Einstellung des `SqlDataSource`-Steuerelements bieten die neuen Grid- und Bearbeitungs-Steuerelemente eine `DataSourceID`-Eigenschaft, die auf die `ID` der Datenquelle gesetzt werden kann. Sie verbindet die beiden Steuerelemente automatisch, so dass beim Laden der Seite die Daten abgerufen und ohne Aufforderung angezeigt werden. Ein Beispiel:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
  ...
</asp:SqlDataSource>
<asp:GridView ID="GridView1" runat="server"
  DataKeyNames="ProductID"
  DataSourceID="SqlDataSource1"></asp:GridView>
```

Die `DataKeyNames`-Eigenschaft erkennt den eindeutigen Schlüssel der zugrunde liegenden Daten; er ist nicht erforderlich für das Anzeigen von Daten, sondern wenn Sie das Grid für die Aktualisierung von Daten verwenden. Das `GridView` zeigt automatisch alle übergebenen Spalten an, stellt als Standard aber nur eine lesbare Ansicht zur Verfügung. Um Bearbeitung und Veränderung zu aktivieren, können Sie die Eigenschaften aus der `GridView`-Aufgabenleiste benutzen, wie in Abbildung 3.8 gezeigt.

Die Aktivierung von BEARBEITEN und LÖSCHEN fügt eine angepasste Spalte zu dem Grid hinzu:

```
<asp:GridView ID="GridView1" runat="server"
  DataKeyNames="ProductID"
  DataSourceID="SqlDataSource1">
  <Columns>
    <asp:CommandField ShowDeleteButton="True"
                      ShowEditButton="True" />
  </Columns>
</asp:GridView>
```

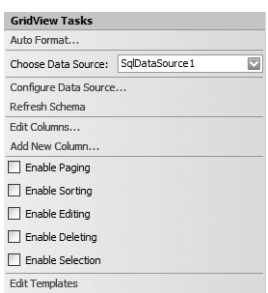


Abbildung 3.8: GridView-Aufgabenleiste

Die Aufgaben enthalten keine Option, um neue Reihen zu aktivieren. Obwohl es eine `ShowInsertButton`-Eigenschaft gibt, unterstützt das Grid selbst nicht die Anzeige einer neuen Zeile; Sie bekommen keine leere Zeile, in die Sie Ihre neuen Daten hinzufügen können. Es gibt Möglichkeiten, dies bei Bedarf zu umgehen,

aber eine weitaus bessere Lösung ist die Verwendung eines `DetailsView`- oder `FormView`-Steuerelements, da hier der `ShowInsertButton` in erster Linie eingesetzt wird. Die Steuerelemente `DetailsView` und `FormView` wollen wir später in diesem Kapitel behandeln.

Die Einstellung `ShowDeleteButton` und `ShowEditButton` auf `True` liefert automatisch die Funktionalitäten zum Bearbeiten und Löschen von Daten, wie Abbildung 3.9 zeigt. Wenn der Link BEARBEITEN ausgewählt ist, wechselt die Zeile automatisch in den Bearbeitungsmodus mit Textboxen für die Bearbeitung der Spalten. Das Klicken auf AKTUALISIEREN (UPDATE) sendet ohne Aufforderung die Veränderungen zurück an die Datenquelle, diese wiederum schickt sie zur SQL-Tabelle.

Abbildung 3.9: Ein GridView im Bearbeiten-Modus

Es ist kein Code erforderlich, um das Bearbeiten und Löschen innerhalb eines Grid einzuschalten. Wenn Sie auf den AKTUALISIEREN-Link klicken, werden die Spaltenwerte in die Parameter der `SqlDataSource` weitergereicht und das `UpdateCommand` ausgeführt. Klicken Sie auf den ABBRECHEN-Link (CANCEL), werden die gegenwärtigen Werte ignoriert und das `UpdateCommand` wird nicht ausgeführt.

3.3.2 Das GridView-Steuerelement anpassen

Das Aussehen in Abbildung 3.9 ist das Standardaussehen und -verhalten für ein `GridView`-Steuerelement. Völlige Flexibilität erreichen Sie durch eigene Einstellungen.

Eigenes Paging

Immobilien abzubilden, ist immer ein Problem, insbesondere wenn man Datentabellen anzeigt. Paging der Daten ist die Lösung des Problems, denn so wird eine begrenzte Anzahl an Zeilen auf einmal gezeigt. Paging kann einfach erreicht werden, indem Sie die `AllowPaging`-Eigenschaft auf `True` setzen und optional die `PageSize`-Eigenschaft auf die Anzahl der anzuzeigenden Zeilen (standardmäßig 10). Das Hinzufügen von Paging platziert als Standard eine zusätzliche Zeile mit der Seitenanzahl unterhalb des Grid, wie Abbildung 3.10 zeigt.

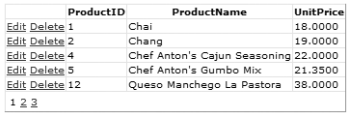


Abbildung 3.10: Das Standard-Paging

Die Seite lässt sich auf zwei Methoden benutzerdefiniert anpassen. Die erste Methode besteht in der Verwendung des `PagerSettings`-Elements, wie die allgemeine Deklaration in Listing 3.5 veranschaulicht:

```
<PagerSettings
  FirstPageImageUrl="String"
  FirstPageText="String"
  LastPageImageUrl="String"
  LastPageText="String"
  Mode="NextPrevious|NextPreviousFirstLast|Numeric|NumericFirstLast"
  NextPageImageUrl="String"
  NextPageText="String"
  PageButtonCount="Integer"
  Position="Bottom|Top|TopAndBottom"
  PreviousPageImageUrl="String"
  PreviousPageText="String" />
```

Listing 3.5: Das PagerSettings-Element

Die `PageButtonCount`-Eigenschaft definiert, wie viele Seiten gezeigt werden. Falls der Wert niedriger angegeben ist als die tatsächliche Anzahl an Seiten, weisen Ellipsen darauf hin, dass mehr Seiten existieren.

Den Pager dürfen Sie auch setzen, um Text anstelle von Seitenzahlen anzuzeigen, indem die `Mode`-Eigenschaft benutzt wird. Wenn dies der Fall ist, definieren die `PageText`-Eigenschaften den zu zeigenden Text. Diese stehen standardmäßig auf <<, <, > und >>, aber Sie können sie auf jeden gewünschten Text setzen, wie Listing 3.6 zeigt:

```
<PageSettings
  Mode="NextPreviousFirstLast"
  FirstPageText="First"
  LastPageText="Last"
  NextPageText="Next"
  PreviousPageText="Previous" />
```

Listing 3.6: Die Paging-Textlinks setzen

Als Alternative ist es möglich, auch die `ImageUrl`-Eigenschaften zu setzen, die Bildlinks anstelle der Textlinks darstellen.

Sie können weitere benutzerdefinierte Anpassungen vornehmen, indem Sie das Paging mittels `PagerTemplate`-Subelement mit Templates versehen. Innerhalb des `PagerTemplate` dürfen Sie beliebigen Inhalt hinzufügen, einschließlich Server-Steuerelemente. Anstelle der Seitenanzahl der NEXT/PREVIOUS-Schaltflächen könnten Sie beispielsweise eine `DropDownList` mit den enthaltenen Seiten platzieren.

Spalten definieren

Falls Sie die automatische Generierung von Spalten nicht akzeptieren möchten, sollten Sie die `AutoGenerateColumns`-Eigenschaft auf `False` setzen und das `Columns`-Subelement verwenden, um Ihre eigenen Spalten zu definieren. Der Vor-

teil liegt darin, dass Sie bestimmen, wie die Spalten im Gegensatz zu den Standardtypen, die vom Grid verwendet werden, in den beiden Modi für Anzeige und Bearbeiten erscheinen. Im Standardmodus kommen Labels zum Einsatz, um Daten darzustellen, eine `TextBox` wird im Bearbeiten-Modus für Strings benutzt und eine `CheckBox` für Bit-Spalten. Sie dürfen anstatt dieser Standards auch die folgenden Steuerelemente einsetzen:

- `BoundField`, das wie eine Standardspalte funktioniert, zeigt Text im Anzeige-Modus an und eine `TextBox` im Bearbeiten-Modus.
- `ButtonField` zeigt eine Schaltfläche in jeder Zeile an. Die Schaltfläche kann ein Standard-`Button`, ein `LinkButton` oder ein `ImageButton` sein.
- `CheckBoxField` stellt Bitdaten als `CheckBox` dar.
- `CommandField` kann einen Textlink, Schaltflächen oder Symbolschaltflächen anzeigen, die für die Ausgabe von Kommandos auf dem Grid benutzt werden. Die Kommandos können Standardkommandos sein, um zwischen Anzeige- und Bearbeiten-Modus zu wechseln. In diesem Fall werden die Standardtexte der Kommandos angezeigt (für Auswählen, Bearbeiten, Löschen, Einfügen und Abbrechen). Oder benutzerdefinierte Kommandos kommen zum Einsatz, um benutzerdefinierte Aktionen abzubilden.
- `HyperLinkField` liefert ein `HyperLink`-Steuerelement.
- `ImageField` zeigt ein `Image`-Steuerelement an.
- `TemplateField` stellt einen benutzerdefinierten Bereich für Inhalt zur Verfügung.

Es gibt unterschiedliche Template-Typen für unterschiedliche Zustände. Sie werden detaillierter im Abschnitt „Template-Spalten“ in diesem Kapitel aufgeführt.

3.3.3 Das BoundField-Steuerelement

Das `BoundField`-Steuerelement zeigt Text im Anzeige-Modus an und eine `TextBox` im Bearbeiten-Modus. Das Feld zur Anzeige wird mit der `DataField`-Eigenschaft gesetzt, während Sie die `DataFormatString`-Eigenschaft verwenden können, um den Wert zu formatieren. So lässt sich beispielsweise ein Währungswert anzeigen, indem Sie `DataFormatString` auf `{0;C}` einstellen. Listing 3.7 zeigt einen Überblick über die Syntax für das `BoundField`-Steuerelement:

```
<asp:BoundField
  DataField="String"
  DataFormatString="String"
  ApplyFormatInEditMode="[True|False]"
  ConvertEmptyStringToNull="[True|False]"
  NullDisplayText="String"
  HtmlEncode="[True|False]"
```

```
ReadOnly="[True|False]"
Visible="[True|False]"
InsertVisible="[True|False]"
SortExpression="String"
ShowHeader="[True|False]"
HeaderText="String"
AccessibleHeaderText="String"
ControlStyle-[PropertyName]="[value]"
HeaderStyle-[PropertyName]="[value]"
HeaderImageUrl="String"
ItemStyle-[PropertyName]="[value]"
FooterText="String"
FooterStyle-[PropertyName]="[value]" />
```

Listing 3.7: Syntax des BoundField-Steuerelements

Eine Beschreibung dieser Eigenschaften ist in Tabelle 3.4 zu finden.

Eigenschaft/Attribut	Beschreibung
`Data Field`	Setzt oder sendet einen `String` zurück, der der Name der Datenspalte ist und der dafür sorgt, dass die Daten in der entsprechenden Spalte des Grid angezeigt werden.
`DataFormatString`	Setzt oder sendet einen `String` zurück, der die formatierten Details für den Wert, der in dieser Spalte angezeigt wird, enthält.
`ApplyFormatInEditMode`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob die Formatierung von der `DataFormatString`-Eigenschaft aus dem „normalen" Modus auch im Bearbeiten-Modus ausgeführt wurde. Ein Beispiel: Sie zeigen ein Währungssymbol an und fügen Nullen in der Textbox im Bearbeiten-Modus hinzu, statt einen einfachen numerischen Wert zu verwenden.
`ConvertEmptyStringToNull`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob ein leerer `String` in dieser Spalte als Null angesehen werden sollte. Das ist bei der Datenbearbeitung nützlich, falls die Datenquelle erwartet, dass Nullwerte benutzt werden, wenn kein Wert vorhanden ist.
`NullDisplayText`	Setzt oder sendet einen `String` zurück, der der anzuzeigende Text für die Zeilen im Grid ist, die einen Nullwert in dieser Spalte haben.
`HtmlEncode`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob die Werte in dieser Spalte HTML-kodiert werden, bevor sie in die vom `GridView`-Steuerelement generierte Ausgabe eingefügt werden.
`ReadOnly`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob die Werte in dieser Spalte bearbeitet werden können. Bei `True` zeigt die Spalte keine Textbox im Bearbeiten-Modus.

Eigenschaft/Attribut	Beschreibung
`Visible`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob die Spalte innerhalb der vom Grid-Steuerelement generierten Ausgabe sichtbar ist.
`InsertVisible`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob diese Spalte sichtbar ist, wenn das Grid sich im Einfügen-Modus befindet.
`SortExpression`	Setzt oder sendet einen `String` zurück, der den Sortierausdruck für diese Spalte als komma-getrennte Liste von Spaltennamen definiert.
`ShowHeader`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob die Kopfzelle für diese Spalte angezeigt wird.
`HeaderText`	Setzt oder sendet einen `String`, der der anzuzeigende Text in der Kopfzelle für diese Spalte ist.
`AccessibleHeaderText`	Setzt oder sendet einen `String` zurück, der den Wert des HTML-`abbr`-Attributs des `<th>`-Elements bestimmt, der die Spaltenüberschriften anzeigt. Nicht visuelle Screenreader und spezialisierte Browser verwenden das `abbr`-Attribut, um dem Nutzer beim Erkennen des Layouts einer Tabelle zu assistieren.
`ControlStyle`	Sendet eine Referenz an eine `Style`-Instanz zurück, die den Stil und die Formatierung eines beliebigen Server-Steuerelements in der Zeile anzeigt.
`HeaderStyle`	Sendet eine Referenz an eine `TabelItemStyle`-Instanz zurück, die den Stil und die Formatierung des Headers der Spalte beschreibt.
`HeaderImageUrl`	Setzt oder sendet einen `String` zurück, der die relative oder absolute URL eines Bilds ist, das in der Header-Reihe der Spalte angezeigt wird.
`ItemStyle`	Sendet eine Referenz an eine `TableItemStyle`-Instanz zurück, die den Stil und die Formatierung der Werte in den Datenverbindungszeilen der Spalte beschreibt.
`FooterText`	Setzt oder sendet einen `String` zurück, der den anzuzeigenden Text in der Footer-Zeile der Spalte darstellt.
`FooterStyle`	Sendet eine Referenz zu einer `TableItemStyle`-Instanz zurück, die den Stil und die Formatierung des Footer der Spalte beschreibt.

Tabelle 3.4: Eigenschaften des BoundField-Steuerelements

3.3.4 Das ButtonField-Steuerelement

Das `ButtonField`-Steuerelement ist dann geeignet, wenn Sie eine Schaltfläche oder einen Link in der Spalte benötigen, um ein Seiten-PostBack zu verursachen. Möglicherweise beabsichtigen Sie, dass mehr Details für die Zeile gezeigt werden sollen, oder Sie möchten eine Aktion wie z.B. das Bearbeiten der Zeile durchführen.

Ein `ButtonField`-Steuerelement kann eine Standardschaltfläche, eine Link-Schaltfläche oder eine Symbolschaltfläche sein. Listing 3.8 gibt einen Überblick über die Syntax des `ButtonField`-Steuerelements:

```
<asp:ButtonField
  ButtonType="[Button|Image|Link]"
  CommandName="String"
  DataTextField="String"
  DataTextFormatString="String"
  CausesValidation="[True|False]"
  ValidationGroup="String"
  Text="String"
  ImageUrl="String"
  ApplyFormatInEditMode="[True|False]"
  HtmlEncode="[True|False]"
  Visible="[True|False]"
  InsertVisible="[True|False]"
  SortExpression="String"
  ShowHeader="[True|False]"
  HeaderText="String"
  AccessibleHeaderText="String"
  ControlStyle-[PropertyName]="[value]"
  HeaderStyle-[PropertyName]="[value]"
  HeaderImageUrl="String"
  ItemStyle-[PropertyName]="[value]"
  FooterText="String"
  FooterStyle-[PropertyName]="[value]" />
```

Listing 3.8: Syntax des ButtonField-Steuerelements

Die fett geschriebenen Eigenschaften sind für das `ButtonField`-Steuerelement spezifisch und werden in Tabelle 3.5 beschrieben, während die anderen Eigenschaften die gleichen sind wie für das `BoundField`-Steuerelement.

Eigenschaft/Attribut	Beschreibung
`ButtonType`	Setzt oder sendet einen Wert von der `ButtonType`-Aufzählung (`Button`, `Image` oder `Link`) zurück, der den zu erstellenden Steuerelementtyp in jeder Zeile der Spalte bestimmt. Der Standard ist `Link`.
`CommandName`	Setzt oder sendet einen `String`-Wert zurück, der die `CommandName`-Eigenschaft der Schaltfläche in jeder Zeile der Ausgabe ist.
`DataTextField`	Setzt oder sendet einen `String` zurück, der den Spaltennamen innerhalb der Datenquelle anzeigt und der als Wert für die `Text`-Eigenschaft des Steuerelements dient (den Titel einer Schaltfläche oder den Text eines Links).

Eigenschaft/Attribut	Beschreibung
`DataTextFormatString`	Setzt oder sendet einen `String` zurück, der die formatierende Information für den Wert in der Zeile enthält. Verwendet die gleiche Syntax wie die `DataFormatString`-Eigenschaft, die für das `BoundField`-Steuerelement beschrieben wurde, und benutzt eine {o} als Platzhalter.
`CausesValidation`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob die Schaltfläche Validierungs-Steuerelemente auf der Seite aufrufen, um Werte zu prüfen und irgendwelche Fehler zu melden. Der Standard ist `True`.
`ValidationGroup`	Setzt oder sendet einen `String` zurück, der der Name der Gruppe der Validierungs-Steuerelemente ist, von dem diese Schaltfläche ein Mitglied sein wird. Siehe für weitere Informationen zu Validierungsgruppen *Kapitel 9*.
`Text`	Setzt oder sendet einen `String` zurück, der anstelle des `DataTextField` benutzt wird, das ist der statische Wert für den Titel oder Text des Links, der derselbe für jede Reihe ist.
`ImageUrl`	Setzt oder sendet einen `String` zurück, der die relative oder absolute URL des zu zeigenden Bilds ist, wenn die `ButtonType`-Eigenschaft auf `Image` gesetzt ist.

Tabelle 3.5: Eigenschaften des ButtonField-Steuerelements

3.3.5 Das CheckBoxField-Steuerelement

Das `CheckBoxField`-Steuerelement wird verwendet, um ein Kontrollkästchen anzuzeigen, das einen `Boolean`-Wert aus den zugrunde liegenden Daten reflektiert. Dieser stammt beispielsweise aus Bitspalten eines SQL Server. Listing 3.9 veranschaulicht die Syntax für das `CheckBoxField`-Steuerelement:

```
<asp:CheckBoxField
  DataField="String"
  DataFormatString="String"
  NullDisplayText="String"
  ReadOnly="[True|False]"
  Text="String"
  ApplyFormatInEditMode="[True|False]"
  HtmlEncode="[True|False]"
  InsertVisible="[True|False]"
  Visible="[True|False]"
  SortExpression="String"
  ShowHeader="[True|False]"
  AccessibleHeaderText="String"
  HeaderText="String"
  ControlStyle-[PropertyName]="[value]"
  HeaderStyle-[PropertyName]="[value]"
  HeaderImageUrl="String"
  ItemStyle-[PropertyName]="[value]"
  FooterText="String"
  FooterStyle-[PropertyName]="[value]" />
```

Listing 3.9: Syntax des CheckBoxField-Steuerelements

Auch hier ist die fett gedruckte Eigenschaft spezifisch für das `CheckBoxField`-Steuerelement und wird in Tabelle 3.6 beschrieben. Die anderen Eigenschaften sind die gleichen wie diejenigen für das `BoundField`-Steuerelement.

Eigenschaft/Attribut	Beschreibung
`Text`	Setzt oder sendet einen `String` zurück, der als `Text`-Eigenschaft des `CheckBox`-Steuerelements verwendet wird.

Tabelle 3.6: Eigenschaften des CheckBoxField-Steuerelements

3.3.6 Das HyperLinkField-Steuerelement

Das `HyperLink`-Steuerelement verwenden Sie, um einen anklickbaren Link in jeder Zeile anzuzeigen. Dies realisiert ASP.NET 2.0 mithilfe eines Standard-HTML `<a>`-Elements. Die Text- und Link-Referenz kann als statischer Text gesetzt werden oder in den Spalten aus zugrunde liegenden Daten eingebunden sein. Listing 3.10 zeigt einen Überblick über die Syntax für das `HyperLinkField`-Steuerelement:

```
<asp:HyperLinkField
  DataTextField="String"
  DataTextFormatString="String"
  Text="String"
  DataNavigateUrlFields="String[,String]"
  DataNavigateUrlFormatString="String"
  NavigateUrl="String"
  Target="String"
  ApplyFormatInEditMode="[True|False]"
  HtmlEncode="[True|False]"
  Visible="[True|False]"
  SortExpression="String"
  ShowHeader="[True|False]"
  HeaderText="String"
  AccessibleHeaderText="String"
  ControlStyle-[PropertyName]="[value]"
  HeaderStyle-[PropertyName]="[value]"
  HeaderImageUrl="String"
  ItemStyle-[PropertyName]="[value]"
  FooterText="String"
  FooterStyle-[PropertyName]="[value]" />
```

Listing 3.10: Syntax des HyperLinkField-Steuerelements

Die fett geschriebenen Eigenschaften sind für das `HyperLinkField`-Steuerelement spezifisch und sind in Tabelle 3.5 beschrieben, während die anderen Eigenschaften die gleichen sind wie für das `BoundField`-Steuerelement.

Eigenschaft/Attribut	Beschreibung
`DataTextField`	Setzt oder sendet einen `String` zurück, der den Spaltennamen innerhalb der Datenquelle anzeigt und den Wert für die `Text`-Eigenschaft des Steuerelements liefert (der sichtbare Text oder der Link).
`DataTextFormatString`	Setzt oder sendet einen `String` zurück, der die Formatinformationen für den gebundenen Wert enthält, der auf die `Text`-Eigenschaft des Links anwendet wird.
`Text`	Setzt oder sendet einen `String` zurück, der anstelle eines `DataTextField` verwendet wird. Das Attribut gibt also mit anderen Worten den statischen Wert für den Text oder den Link an, der für jede Zeile der gleiche ist.
`DataNavigateUrlFields`	Setzt oder sendet ein `String`-Array, das die Spaltennamen innerhalb der Datenquelle spezifiziert, die Werte für die `NavigateUrl`-Eigenschaft des Steuerelements liefern werden (das `href`-Attribut des resultierenden `<a>`-Elements). Verwenden Sie eine komma-getrennte Liste der Spaltennamen bei Deklaration des Steuerelements. Das bedeutet, dass Sie die Werte aus unterschiedlichen Spalten für Hyperlinks verwenden können. Mehr dazu im nächsten Abschnitt zu `DataNavigateUrlFields`.
`DataNavigateUrlFormatString`	Setzt oder sendet einen `String` zurück, der die formatierenden Informationen für die gebundenen Werte der `NavigateUrl`-Eigenschaft enthält.
`NavigateUrl`	Setzt oder sendet einen `String` zurück, der anstelle der `DataNavigateUrlFields` verwendet wird. Dies ist also der statische Wert für das `href`-Attribut der Links, sprich, er ist für jede Zeile derselbe.
`Target`	Setzt oder sendet einen `String` mit dem Namen des Zielfensters für den Link zurück und wird als Attribut der resultierenden `<a>`-Elemente benutzt.

Tabelle 3.7: Eigenschaften des HyperLinkField-Steuerelements

3.3.7 Die DataNavigateUrlFields-Eigenschaft

Sie können mehr als eine Spalte für die `DataNavigateUrlFields`-Eigenschaft oder für das Attribut eines Grid-Steuerelements benennen.

Wenn das `GridView` ein `HyperlinkField` mit seiner Datenquelle verbindet, dürfen alle für die `DataNavigateUrlFields`-Eigenschaften deklarierten Spalten innerhalb der `DataTextFormatString`-Eigenschaft verwendet werden, um mehr als einen `href`-Wert für die Hyperlinks zur Verfügung zu stellen. Sie können zum Beispiel ein `HyperlinkField` auf diese Weise ausdrücken:

```
<asp:HyperLinkField DataTextField="ProductName"
  DataNavigateUrlFields="ProductID,ProductName"
  DataNavigateUrlFormatString=
    "http://www.mysite.com/products?product={0}" />
```

In diesem Fall erscheint der `href`-Wert für eine Zeile, die das Produkt namens `Chang` mit dem `ProductID` Wert `2` beinhaltet als:

```
http://www.mysite.com/products?product=2
```

Allerdings können Sie den `DataNavigateUrlFormatString` auch ausdrücken als:

```
  "http://www.mysite.com/products?product={1}" />
```

Hier sehen Sie den `href`-Wert für die gleiche Zeile:

```
http://www.mysite.com/products?product=Chang
```

Derselbe Effekt kann selbstverständlich durch Veränderung des Werts von `DataNavigateUrlFormatString` während der Laufzeit im `ItemDataBound`-Ereignis erreicht werden – genauso wie Sie es mit einem der ASP.NET 1.x-Grids oder Listen-Steuerelemente machen würden.

Allerdings besteht die nützlichste und am meisten geforderte Funktion darin, dass `DataNavigateUrlFormatString` jetzt mehr als einen Platzhalter enthalten kann:

```
DataNavigateUrlFormatString=
  "http://www.mysite.com/products?product={0}&amp;name={1}" />
```

Der `href`-Wert für die gleiche Zeile erscheint jetzt so:

```
http://www.mysite.com/products?product=2&name=Chang
```

Das gibt Ihnen die Möglichkeit, einfach `href`-Werte zu erstellen, die mehrere Anfragestring-Parameter enthalten. Denken Sie bitte daran, das HTML-encodierte kaufmännische Und (`&`) zu verwenden, um die Parameter miteinander zu verketten.

3.3.8 Das ImageField-Steuerelement

Das `ImageField`-Steuerelement verwenden Sie für die Darstellung von Bildern. Dabei werden die zugrunde liegenden Daten benutzt, um die URL für das Bild anzugeben. Listing 3.11 zeigt Ihnen die Syntax für das `ImageField`-Steuerelement.

Auch hier gilt: Die fett gedruckten Eigenschaften sind für das `ImageField`-Steuerelement spezifisch (Beschreibung in Tabelle 3.8), während die anderen Eigenschaften die gleichen sind wie diejenigen für das `BoundField`-Steuerelement.

```
<asp:ImageField
  DataImageUrlField="[String]"
  DataImageUrlFormatString="[String]"
  DataImageTextField="[String]"
  DataImageTextFormatString="[String]"
  AlternateText="[String]"
  NullImageUrl="[String]"
```

```
ApplyFormatInEditMode="[True|False]"
HtmlEncode="[True|False]"
Visible="[True|False]"
SortExpression="String"
ShowHeader="[True|False]"
HeaderText="String"
AccessibleHeaderText="String"
HeaderStyle-[PropertyName]="[value]"
HeaderImageUrl="String"
ItemStyle-[PropertyName]="[value]"
FooterText="String"
FooterStyle-[PropertyName]="[value]" />
```

Listing 3.11: Syntax des ImageField-Steuerelements

Eigenschaft/Attribut	Beschreibung
`DataImageUrlField`	Setzt oder sendet einen `String` zurück, der den Spaltennamen innerhalb der Datenquelle angibt, der den Wert für die `ImageUrl`-Eigenschaft des Bild-Steuerelements liefert.
`DataImageUrlFormatString`	Setzt oder sendet einen `String` zurück, der die Formatinformation für den gebundenen Wert enthält, der die `Url`-Eigenschaft des Bilds enthält.
`DataAlternateTextField`	Setzt oder sendet einen `String` zurück, der den Spaltennamen innerhalb der Datenquelle angibt, der den Wert für die `AlternateText`-Eigenschaft des Bild-Steuerelements liefert.
`DataAlternateTextFormatString`	Setzt oder sendet einen `String` zurück, der die formatierende Information für den gebundenen Wert enthält, der zur `AlternateText`-Eigenschaft des Bilds gehört.
`AlternateText`	Setzt oder sendet einen `String` zurück, der den angezeigten Alternativtext des Bilds enthält.
`NullImageUrl`	Setzt oder sendet einen `String` zurück, der die URL des anzuzeigenden Bilds enthält, wenn die Daten im `DataImageUrlField` einen Nullwert beinhalten.

Tabelle 3.8: Eigenschaften des ImageField-Steuerelements

Command-Spalten

Sie haben zu Beginn des Kapitels gesehen, dass man eine Spalte hinzufügen darf, die Befehle zum Grid sendet. Die Kommandos können die Zeilenauswahl, die Zeilenlöschung und den Wechsel zwischen dem Anzeige- und Bearbeiten-Modus übernehmen oder aus einem benutzerdefinierten Befehl bestehen. Um die Befehle einem Grid hinzuzufügen, verwenden Sie ein `CommandField`-Steuerelement (vergleiche Listing 3.12).

```
<asp:CommandField
  ButtonType="[Button|Image|Link]"
  UpdateText="String"
  UpdateImageUrl="String"
  ShowCancelButton="[True|False]"
  CancelText="String"
  CancelImageUrl="String"
  ShowSelectButton="[True|False]"
  SelectText="String"
  SelectImageUrl="String"
  ShowEditButton="[True|False]"
  EditText="String"
  EditImageUrl="String"
  ShowInsertButton="[True|False]"
  InsertText="String"
  InsertImageUrl="String"
  NewText="String"
  NewImageUrl="String"
  ShowDeleteButton="[True|False]"
  DeleteText="String"
  DeleteImageUrl="String"
  CausesValidation="[True|False]"
  ValidationGroup="String"
  Visible="[True|False]"
  SortExpression="String"
  ShowHeader="[True|False]"
  HeaderText="String"
  AccessibleHeaderText="String"
  ControlStyle-[PropertyName]="[value]"
  HeaderStyle-[PropertyName]="[value]"
  HeaderImageUrl="String"
  ItemStyle-[PropertyName]="[value]"
  FooterText="String"
  FooterStyle-[PropertyName]="[value]" />
```

Listing 3.12: Syntax des CommandField-Steuerelements

Die fett geschriebenen Eigenschaften sind für das `CommandField`-Steuerelement spezifisch und werden in Tabelle 3.9 definiert, während die anderen Eigenschaften Ihnen vom `BoundField`-Steuerelement bekannt sind.

Eigenschaft/Attribut	Beschreibung
`ButtonType`	Setzt oder sendet einen Wert aus `ButtonType` (`Button`, `Image` oder `Link`) zurück, der den Steuerelementtyp in jeder Zeile der Spalte bestimmt. Der Standard ist `Link`.
`UpdateType`	Setzt oder sendet einen `String`-Wert zurück, welcher der Titel für die Schaltfläche ist, die bewirkt, dass ein Aktualisierungsprozess aufgerufen wird. Der Standard ist `Update`.
`UpdateImageUrl`	Setzt oder sendet einen `String` zurück, der die relative oder absolute URL des anzuzeigenden Bilds anstelle eines Text-UPDATE-Links ist.

Eigenschaft/Attribut	Beschreibung
`ShowCancelButton`	Setzt oder sendet einen `Boolean`-Wert, der anzeigt, ob eine ABBRECHEN-Schaltfläche in der Spalte dargestellt wird, wenn die Reihe im Bearbeitungsmodus ist.
`CancelText`	Setzt oder sendet einen `String`-Wert mit Titel der Schaltfläche zurück, die einen Aktualisierungsprozess abbricht. Der Standard ist ABBRECHEN.
`CancelImageUrl`	Setzt oder sendet einen `String` zurück, der die relative oder absolute URL des anzuzeigenden Bilds anstelle des Text-ABBRECHEN-Links ist.
`ShowSelectButton`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob eine AUSWÄHLEN-Schaltfläche in der Spalte dargestellt wird.
`SelectText`	Setzt oder sendet einen `String`-Wert zurück, der der Titel für die Schaltfläche ist, die dafür sorgt, dass die Zeile ausgewählt werden. Der Standard ist AUSWÄHLEN.
`SelectImageUrl`	Setzt oder sendet einen `String` zurück, der die relative oder absolute URL des anzuzeigenden Bilds anstelle eines Text-AUSWÄHLEN-Links ist.
`ShowEditButton`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob eine BEARBEITEN-Schaltfläche in der Spalte dargestellt wird.
`EditText`	Setzt oder sendet einen `String`-Wert zurück, der der Titel der Schaltfläche ist, die dafür sorgt, dass die Zeile im Bearbeiten-Modus angezeigt wird. Der Standard ist BEARBEITEN.
`EditImageUrl`	Setzt oder sendet einen `String` zurück, der die relative oder absolute URL des anzuzeigenden Bilds anstelle eines Text-BEARBEITEN-Links ist.
`ShowInsertButton`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob eine EINFÜGEN-Schaltfläche in der Spalte dargestellt wird. Einfügen hat nur in den Steuerelementen `DetailsView` und `FormView` einen Effekt und wird im `GridView`-Steuerelement nicht unterstützt.
`InsertText`	Setzt oder sendet einen `String`-Wert zurück, der der Titel für die Schaltfläche ist, die dafür sorgt, dass die Zeile im Einfügen-Modus angezeigt wird. Der Standard ist EINFÜGEN.
`InsertImageUrl`	Setzt oder sendet einen `String` zurück, der die relative oder absolute URL des anzuzeigenden Bilds anstelle eines Text-EINFÜGEN-Links ist.
`NewText`	Setzt oder sendet einen `String`-Wert zurück, der der Titel für die Schaltfläche ist, die dafür sorgt, dass die Reihe im Modus für neuen Inhalt dargestellt wird. Der Standard ist NEU.
`NewImageUrl`	Setzt oder sendet einen `String` zurück, der die relative oder absolute URL des anzuzeigenden Bilds anstelle eines Text-NEU-Links ist.
`ShowDeleteButton`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob eine LÖSCHEN-Schaltfläche in der Spalte dargestellt wird.

Eigenschaft/Attribut	Beschreibung
`DeleteText`	Setzt oder sendet einen `String`-Wert zurück, der der Titel für die Schaltfläche ist, die eine Zeile löscht. Der Standard ist LÖSCHEN.
`DeleteImageUrl`	Setzt oder sendet einen `String` zurück, der die relative oder absolute URL des anzuzeigenden Bilds anstelle eines Text-LÖSCHEN-Links ist.
`CausesValidation`	Setzt oder sendet einen `Boolean`-Wert zurück, der anzeigt, ob die Schaltfläche Validierungs-Steuerelemente auf der Seite verursacht, um deren Werte zu bestätigen und jeden Fehler zu melden. Der Standard ist `True`.
`ValidationGroup`	Setzt oder sendet einen `String` zurück, der der Gruppenname der Validierungs-Steuerelemente ist, zu der diese Schaltfläche dazugehören wird. Für weitere Informationen zu Validierungsgruppen siehe *Kapitel 9*.

Tabelle 3.9: Eigenschaften des CommandField-Steuerelements

Sie können ein `CommandField`-Steuerelement für automatisiertes Management der Datenbearbeitung verwenden. Das Setzen der `ShowEditButton`-Eigenschaft auf `True` stellt beispielsweise einen BEARBEITEN-Link in der Spalte dar. Durch das Klicken auf diesen Link wechselt die Zeile in den Bearbeiten-Modus, der automatisch die Links AKTUALISIEREN und ABBRECHEN anbietet. Der erste Link aktualisiert die zugrunde liegenden Daten mit den Veränderungen des Nutzers, während der zweite die vom Nutzer vorgenommenen Veränderungen abbricht und löscht. Beide wechseln in den Anzeige-Modus zurück. Die `ShowDeleteButton`-Eigenschaft zeigt einen LÖSCHEN-Link, der die betroffene Zeile ohne Nutzerbestätigung löscht. Bestätigung und Ereignisse werden im Abschnitt „Ereignisse verwenden“ später im Kapitel diskutiert.

Template-Spalten

Falls die hier dargestellten Spaltentypen Ihnen nicht genügend Kontrollmöglichkeiten über das Layout Ihrer Daten geben, können Sie auf Template-Spalten zurückgreifen. Sie funktionieren sehr ähnlich wie die Templates in den Steuerelementen der `DataList` oder des `DataGrid`. Bestimmen Sie den Output in einem oder mehreren Templates und das Steuerelement wählt das geeignete Template in Abhängigkeit vom Status des Grid aus. Listing 3.13 veranschaulicht die Syntax für ein `TemplateField`-Steuerelement:

```
<asp:TemplateField
  ApplyFormatInEditMode="[True|False]"
  HtmlEncode="[True|False]"
  Visible="[True|False]"
  SortExpression="String"
  ShowHeader="[True|False]"
```

```
  HeaderText="String"
  AccessibleHeaderText="String"
  HeaderStyle-[PropertyName]="[value]"
  HeaderImageUrl="String"
  ItemStyle-[PropertyName]="[value]"
  FooterText="String"
  FooterStyle-[PropertyName]="[value]" >
    <HeaderTemplate>...</HeaderTemplate>
    <ItemTemplate>...</ItemTemplate>
    <AlternatingItemTemplate>...</ AlternatingItemTemplate>
    <EditItemTemplate>...</EditItemTemplate>
    <FooterTemplate>...</FooterTemplate>
 <asp:TemplateField>
```

Listing 3.13: Syntax des TemplateField-Steuerelements

Alle Attribute des Steuerelements sind die gleichen wie diejenigen, die für das `BoundField`-Steuerelement in Tabelle 3.4 aufgelistet sind. Die fünf Template-Arten, die Sie wie in Listing 3.13 hervorgehoben bestimmen können, sind in Tabelle 3.10 dokumentiert.

Wenn Sie Template-Spalten benutzen, müssen Sie die Bindung an Daten direkt festsetzen. Dafür verwenden Sie die Methoden `Eval` oder `Bind`. Der Unterschied zwischen den beiden Methoden besteht darin, dass `Eval` eine nur lesbare Verbindung zur Verfügung stellt, während `Bind` eine les- und schreibbare Bindung unterstützt. Prinzipiell setzen Sie die Methoden in unterschiedlichen Templates ein. Betrachten Sie beispielsweise Listing 3.14. Dort wird ein `ItemTemplate` benutzt, um den Preis eines Produkts anzuzeigen. Dagegen zeigt das `EditItemTemplate` eine `TextBox`, hat aber auch einen `RequiredFieldValidator`, um zu gewährleisten, dass ein Wert für den Preis eingegeben ist.

Template	Beschreibung
`HeaderTemplate`	Tags, Text, Steuerelemente und anderer Inhalt, der erforderlich ist, um den ganzen Inhalt für den Header der Spalte des Grid zu erstellen.
`ItemTemplate`	Tags, Text, Steuerelemente und anderer Inhalt, der erforderlich ist, um den ganzen Inhalt für die Spalte in dateneingebundenen Zellen innerhalb des Grid zu erstellen.
`Alternating ItemTemplate`	Tags, Text, Steuerelemente und anderer Inhalt, der erforderlich ist, um den ganzen Inhalt der Spalte in alternierenden datengebundenen Zeilen innerhalb des Grid zu erstellen.
`EditItemTemplate`	Tags, Text, Steuerelemente und anderer Inhalt, der erforderlich ist, um den ganzen Inhalt der Spalte in der Zeile innerhalb des Grid zu erstellen, der im Bearbeiten-Modus ist.
`FooterTemplate`	Tags, Text, Steuerelemente und anderer Inhalt, der erforderlich ist, um den ganzen Inhalt für den Footer der Spalte des Grid zu erstellen.

Tabelle 3.10: Templates des TemplateField-Steuerelements

```
<asp:TemplateField HeaderText="Price"  SortExpression="UnitPrice"
  ItemStyle-Font-Bold="True">
  <ItemTemplate>
    <asp:Label runat="server"
      Text='<%# Eval("UnitPrice", "${0:F2}") %>' />
  </ItemTemplate>
  <EditItemTemplate>
    <asp:TextBox ID="UnitPrice" runat="server"
      Text='<%#Bind("UnitPrice")%>' />
    <asp:RequiredFieldValidator ID="rfv1" runat="server"
      ControlToValidate="UnitPrice"
      Text="You must enter the price" />
  </EditItemTemplate>
</asp:TemplateField>
```

Listing 3.14: Zweiseitige Verbindung in einem Template

Diese Technik ist sehr nützlich, wenn Sie die Datenaktualisierungsfunktionen für die Grid- und Datenquellen-Steuerelemente verwenden möchten. Außerdem benötigen Sie eigenen Inhalt wie Validierungslisten oder Auswahlmenülisten, wenn der Nutzer Daten bearbeitet.

> Die Verwendung von `Eval` ist eine Abkürzung der alten Syntax von DataBinder.Eval. `Bind` ist in ASP.NET Version 2.0 neu.

Spalten gestalten

Sie können Spalten auf verschiedene Weisen gestalten: entweder durch Verwendung von Eigenschaften auf den Spalten selbst oder durch das Setzen von allgemeinen Spalteneigenschaften im Grid. Letzteres lässt sich auf zwei Wegen durchführen. Die erste Möglichkeit besteht darin, die Eigenschaften auf der `GridView`-Deklaration wie folgt zu setzen:

```
<asp:GridView ID="grid1" runat="server" DataSourceID="ds1"
  RowStyle-BackColor="Aqua" HeaderStyle-Font-Bold="True"
```

Alternativ können Sie das Design auch als separates Element festlegen:

```
<asp:GridView ID="grid1" runat="server" DataSourceID="ds1">
  <HeaderStyle Font-Bold="true" Font-Names="Verdana" />
  <RowStyle Font-Names="Verdana" />
</asp:GridView>
```

Es gibt praktisch keinen Unterschied zwischen den beiden Methoden, obwohl der letztere Stil einfacher zu lesen ist und das Design getrennt hält vom Rest der Grid-Deklaration. Stile lassen sich auch auf Spalten anwenden:

```
<asp:HyperLinkField DataTextField="ProductName"
  ItemStyle-Font-Bold="True" ItemStyle-BackColor="Yellow" />
```

Genauso gut wie die Verwendung individueller Stilelemente ist der Einsatz eines CSS für das Design von Elementen. In diesem Fall machen Sie von der `CssClass`-Eigenschaft Gebrauch:

```
<asp:HyperLinkField DataTextField="ProductName"
  CssClass="GridLink" />
```

Die folgenden acht Stilelemente stehen zur Verfügung:

- `AlternatingRowStyle` für alternierende Zeilen (`RowStyle` wickelt die anderen ab)
- `EditRowStyle` für Zeilen im Bearbeiten-Modus
- `EmptyDataRowStyle` für die dargestellte Zeile, wenn es keine Daten für die Anzeige gibt
- `FooterStyle` für die Footer-Zeile
- `HeaderStyle` für die Header-Zeile
- `PagerStyle` für die Paging-Zeile mit den Seitenangaben
- `RowStyle` für alle Zeilen, außer wenn `AlternatingRowStyle` gesetzt ist
- `SelectedRowStyle` für Zeilen, die von einem Auswahl-Befehl ausgewählt sind

Stile, die auf einzelne Spalten verwendet werden, überschreiben alle Stile, die allgemein für das Grid gesetzt sind.

GridView-Befehle anpassen

Sie sind nicht darauf beschränkt, ein `CommandField` für das Aussenden von Kommandos zu verwenden, weil Sie jede Steuerelement-Schaltfläche (alles, was ein PostBack hervorruft) so setzen können, dass sie ein Standardkommando emuliert. Jede Versendeschaltfläche hat zwei Eigenschaften, `CommandName` und `CommandArgument`, die sich auf ein bestimmtes Kommando übertragen lassen. Wenn Sie auf eine entsprechende Schaltfläche innerhalb des Kontexts einer Grid-Zeile klicken, wird das Kommando ausgeführt. Betrachten Sie beispielsweise folgenden Code:

```
<asp:LinkButton id="LinkButton1" runat="server"
  CommandName="Edit" Text="Edit" />
```

Falls dieser Code in einer Spalte platziert ist, wird das Klicken auf den Link das Bearbeitungskommando ausführen und die Zeilen wechseln in den Bearbeitungsmodus, genau wie bei einer Standard-COMMAND-Schaltfläche zum Bearbeiten. Sie können die `CommandName`-Eigenschaft auf EINFÜGEN, ABBRECHEN, LÖSCHEN, NEU oder AUSWÄHLEN setzen.

Alles zusammen – ein eigenes GridView

Es gibt keine Einschränkung, wie Sie Spalten kombinieren. Sie können sogar die automatische Generierung der Spalten aktiviert lassen und dennoch Spalten im `<Columns>`-Bereichs definieren. In diesem Fall wird jede Spalte, die Sie explizit definieren, an den Anfang des Grid platziert, während automatische Spalten danach folgen. Prinzipiell ist es besser, die Spalten ausdrücklich zu benennen, weil Sie dann die Kontrolle darüber haben, wie sie erscheinen. Abbildung 3.11 zeigt ein Grid mit einer Vielfalt an Spaltentypen und Formaten.

Die Definition dieses Grid entnehmen Sie Listing 3.15. Die automatische Generierung der Spalten ist deaktiviert, während Sortieren und Paging mit einem Stil, der für den Header und jede Zeile gesetzt ist, aktiviert sind.

Bis jetzt hat das Grid nicht das beste Aussehen, aber Sie können erkennen, dass die Spaltentypen und das Spaltendesign Ihnen eine große Auswahl an Flexibilität in Aussehen und Handhabung der Grid bieten. Obwohl eine Bearbeitung möglich ist, sehen Sie, dass das `GridView` Beschränkungen haben könnte, wenn Sie mehr als ein oder zwei Spalten bearbeiten. Beispielsweise ist das Hinzufügen von Validierung erlaubt, aber Validierungsfehler passen möglicherweise nicht innerhalb der Grid-Zellen. Eine weitere Einschränkung ergibt sich, wenn Sie eine Spalte haben, die eine große Menge an Text akzeptiert, zum Beispiel eine Produktbeschreibung. In diesem Fall wäre eine Zelle innerhalb eines Grid nur mit Einschränkungen sowohl für die Anzeige als auch für die Bearbeitung der Daten sinnvoll. Eine bessere Bearbeitungslösung bzw. Anzeige einzelner Datenreihen liefert entweder ein `DetailsView`-Steuerelement oder ein `FormView`-Steuerelement.

```
<asp:GridView ID="grid1" runat="server" DataSourceID="ds1"
  DataKeyNames="ProductID"
  AutoGenerateColumns="False"
  AllowSorting="True" AllowPaging="True" PageSize="5">
  <HeaderStyle Font-Bold="True" Font-Names="Verdana" />
  <RowStyle Font-Names="Verdana" />
  <Columns>
    <asp:ButtonField ButtonType="Button" DataTextField="ProductID"
      SortExpression="ProductID" HeaderText="ID" />
    <asp:HyperLinkField DataTextField="ProductName"
      DataNavigateUrlFields="ProductID,ProductName"
      DataNavigateUrlFormatString="http://www.site-that-shows-more-info.com/
products?product={0}&amp;name={1}"
      SortExpression="ProductName" HeaderText="Product"
      ItemStyle-Font-Bold="True" ItemStyle-BackColor="Yellow" />
    <asp:BoundField DataField="QuantityPerUnit"
      HeaderText="Packaging"  />
    <asp:CheckBoxField DataField="Discontinued" HeaderText="N/A" >
    <asp:TemplateField HeaderText="Price"
      SortExpression="UnitPrice" ItemStyle-Font-Bold="True">
      <ItemTemplate>
        <asp:Label runat="server"
          Text='<%# Eval("UnitPrice", "${0:F2}") %>62;' />
      </ItemTemplate>
```

```
      <AlternatingItemTemplate>
        <asp:Label runat="server" ForeColor="DarkGray"
          Text='<%# Eval("UnitPrice", "${0:F2}") %>' />;
      </AlternatingItemTemplate>
      <EditItemTemplate>
        <asp:TextBox ID="UnitPrice" runat="server"
          Text='<%#Bind("UnitPrice")%>' />
        <asp:RequiredFieldValidator ID="rfv1" runat="server"
          ControlToValidate="UnitPrice"
          Text="You must enter the price" />
      </EditItemTemplate>
    </asp:TemplateField>
    <asp:CommandField ButtonType="Image" ShowCancelButton="True"
      ShowEditButton="True" ShowDeleteButton="True"
      CancelImageUrl="s.gif" EditImageUrl="q.gif"
      UpdateImageUrl="i.gif" DeleteImageUrl="x.gif"
      CancelText="Cancel this update" EditText="Edit this row"
      UpdateText="Apply these changes" DeleteText="Delete this row"/>
  </Columns>
</asp:GridView>
```

Listing 3.15: Die GridView-Definition

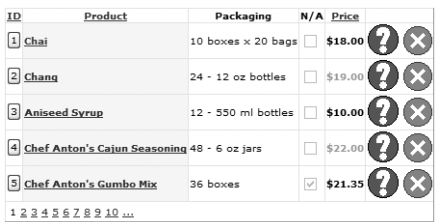


Abbildung 3.11: Ein GridView mit gemischten Spaltentypen

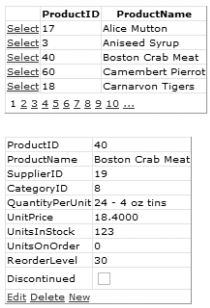
3.3.9 Das DetailsView-Steuerelement

Das `DetailsView`-Steuerelement stellt einzelne Datenzeile eher in einem Formular-Stil als in einem Grid-Stil zur Verfügung. Das `DetailsView` kann allein oder in Kombination mit einem `GridView` benutzt werden, um individuelle Zeilen anzuzeigen und zu bearbeiten. Das funktioniert gut, wenn man die Auswahl von einem Grid mit einem `SelectParameters` auf einem `SqlDataSource`-Steuerelement kombiniert (siehe Listing 3.16).

```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
  ConnectionString="<%$ ConnectionStrings:NorthwindConnectString %>"
  DeleteCommand="..." InsertCommand="..." UpdateCommand="..."
  SelectCommand="SELECT * FROM Products
                 WHERE ProductID = @ProductID)">
  ...
  <SelectParameters>
    <asp:ControlParameter ControlID="GridView1"
      Name="ProductID" PropertyName="SelectedValue"
      Type="Int32" />
  </SelectParameters>
</asp:SqlDataSource>
```

```
<asp:DetailsView ID="DetailsView1" runat="server"
  DataKeyNames="ProductID" DataSourceID="SqlDataSource2">
</asp:DetailsView>
```

Listing 3.16: Ein DetailsView-Steuerelement verwenden



	ProductID	ProductName
Select	17	Alice Mutton
Select	3	Aniseed Syrup
Select	40	Boston Crab Meat
Select	60	Camembert Pierrot
Select	18	Carnarvon Tigers

1 2 3 4 5 6 7 8 9 10 ...

ProductID	40
ProductName	Boston Crab Meat
SupplierID	19
CategoryID	8
QuantityPerUnit	24 - 4 oz tins
UnitPrice	18.4000
UnitsInStock	123
UnitsOnOrder	0
ReorderLevel	30
Discontinued	☐

Edit Delete New

Abbildung 3.12: Einzelne Zeile mit einem DetailsView-Steuerelement zeigen

Abbildung 3.12 zeigt das `DetailsView` mit Darstellung einer Datenzeile, bei der die Zeile in einem `GridView` ausgewählt wurde.

Als Standard liefert das `DetailsView` keine Bearbeitungskommandos, diese lassen sich jedoch durch ein oder zwei Methoden hinzufügen. Zum einen durch Eigenschaften für das Steuerelement selbst:

```
<asp:DetailsView ID="DetailsView1" runat="server"
  DataKeyNames="ProductID" DataSourceID="SqlDataSource2"
  AutoGenerateInsertButton="True"
  AutoGenerateDeleteButton="True"
  AutoGenerateEditButton="True">
```

Auf diese Art würden Sie die Befehle unten an das Steuerelement anfügen, wie in Abbildung 3.12 dargestellt. Alternativ kann ein `CommandField` explizit so definiert sein:

```
<asp:DetailsView ID="DetailsView1" runat="server"
  DataKeyNames="ProductID" DataSourceID="SqlDataSource2">
  <Fields>
    <asp:CommandField ShowDeleteButton="True" ShowEditButton="True"
      ShowInsertButton="True" />
  </Fields>
</asp:DetailsView>
```

Wie das `GridView` ist das `DetailsView` standardmäßig sowohl für die Anzeige als auch für die Bearbeitung von Daten geeignet. Doch es hat die gleichen Einschränkungen, beispielsweise in Hinblick darauf, wie unterschiedliche Steuerelemente angezeigt werden, um Daten zu bearbeiten.

Felder definieren

Die `DetailsView` generiert automatisch Felder, die auf den zugrunde liegenden Daten basieren. Wie das `GridView` auch, kann die automatische Generierung von

Feldern ausgeschaltet werden und die Felder können direkt bestimmt werden. Dazu verwenden Sie die `AutoGenerateRows`-Eigenschaft und das `Fields`-Subelement (vergleiche Listing 3.17):

```
<Fields>
  <asp:BoundField DataField="ProductID"
    HeaderText="ProductID" InsertVisible="False"
    ReadOnly="True" SortExpression="ProductID" />
  <asp:BoundField DataField="ProductName"
    HeaderText="ProductName" SortExpression="ProductName" />
  <asp:BoundField DataField="SupplierID"
    HeaderText="SupplierID" SortExpression="SupplierID" />
  <asp:BoundField DataField="CategoryID"
    HeaderText="CategoryID" SortExpression="CategoryID" />
  <asp:BoundField DataField="QuantityPerUnit"
    HeaderText="QuantityPerUnit" SortExpression="QuantityPerUnit" />
  <asp:BoundField DataField="UnitPrice"
    HeaderText="UnitPrice" SortExpression="UnitPrice" />
  <asp:BoundField DataField="UnitsInStock"
    HeaderText="UnitsInStock" SortExpression="UnitsInStock" />
  <asp:BoundField DataField="UnitsOnOrder"
    HeaderText="UnitsOnOrder" SortExpression="UnitsOnOrder" />
  <asp:BoundField DataField="ReorderLevel"
    HeaderText="ReorderLevel" SortExpression="ReorderLevel" />
  <asp:CheckBoxField DataField="Discontinued"
    HeaderText="Discontinued" SortExpression="Discontinued" />
  <asp:CommandField ShowDeleteButton="True"
    ShowEditButton="True" ShowInsertButton="True" />
</Fields>
```

Listing 3.17: Felder auf einem DetailsView definieren

Wie Sie erkennen, handelt es sich um die gleichen Felder wie im `GridView`. Somit können Sie `TemplateField`-Steuerelemente verwenden, um Benutzerdefiniertes, wie beispielsweise Auswahlmenülisten für Lieferanten, sowie Kategorien zur Verfügung zu stellen. Listing 3.18 zeigt ein Beispiel mit Lieferanten:

```
<asp:TemplateField HeaderText="Supplier">
  <ItemTemplate>
    <%#Eval("CompanyName") %>
  </ItemTemplate>
  <EditItemTemplate>
    <asp:SqlDataSource ID="Sds1" runat="server"
     ConnectionString="<%$ConnectionStrings:NorthwindConnectString%>"
      SelectCommand="SELECT SupplierID, CompanyName
                     FROM Suppliers ORDER BY CompanyName" />
    <asp:DropDownList ID="SupplierID" runat="server"
      DataSourceId="Sds1"
      DataValueField="SupplierID" DataTextField="CompanyName"
      SelectedValue='<%#Bind("SupplierID")%>' />
  </EditItemTemplate>
</asp:TemplateField>
```

Listing 3.18: TemplateField-Steuerelemente verwenden

Sie sehen, dass das `ItemTemplate` einfach den `CompanyName` anzeigt, während das `EditItemTemplate` eine `SqlDataSource` und eine `DropDownList` enthält. Die Datenquelle holt die ID sowie den Namen aller Lieferanten und die Liste verbindet zu diesen Daten, wie aus Abbildung 3.13 hervorgeht. Das `DataValueField` nimmt die ID auf, während das `DataTextField` die Datenanzeige enthält. Schlüsselobjekt ist hier das `SelectValue`, das auf einen Ausdruck zur Datenbindung gesetzt ist: Hier wird `Bind` verwendet, um eine zweiseitige Verbindung zur `SupplierID`-Spalte zu ermöglichen. Da `SelectedValue` zum Einsatz kommt, wird die `SupplierID` der zugrunde liegenden Zeile ausgewählt. Und weil `Bind` verwendet wird, sorgt ein Klicken auf den AKTUALISIEREN-Link dafür, dass die neue `SupplierID` durch das Datenquellen-Steuerelement zurück in die Datenbank geschoben wird.

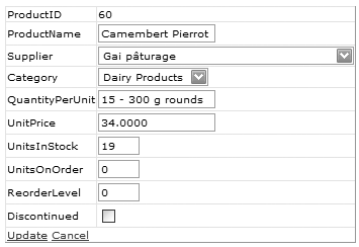


Abbildung 3.13: TemplateColumns mit DropDownLists

Das `DetailsView` kann weiterhin ähnlich wie beim `GridView` mit Designs benutzerdefiniert angepasst werden. Das `DetailsView` unterstützt die folgenden Stilelemente:

- `AlternatingRowStyle` für alternierende Zeilen (`RowStyle` handhabt die anderen)
- `CommandRowStyle` für die Zeile, die die Kommandos enthält (Bearbeiten, Neu usw.)
- `EditRowStyle` für Zeilen im Bearbeiten-Modus
- `EmptyDataRowStyle` für die Zeile, die angezeigt wird, wenn es keine Daten anzuzeigen gibt
- `FieldHeaderStyle` für die Zeilen-Header
- `FooterStyle` für die Footer-Zeile
- `HeaderStyle` für die Header-Zeile
- `InsertRowStyle` für die Zeilen, die im Einfügen-Modus angezeigt werden
- `PageStyle` für die Seitenzeile
- `RowStyle` für alle Zeilen, außer wenn `AlternatingRowStyle` eingestellt ist

Es gibt noch andere Eigenschaften, wie beispielsweise `GridLines`, das die begrenzenden Linien festlegt, so dass Sie das Grid-ähnliche Aussehen, das dieses Steuerelement hat, entfernen können.

Ein DetailsView alleine

Das `DetailsView` muss nicht in Verbindung mit einem anderen Steuerelement verwendet werden, sondern lässt sich eigenständig nutzen. Während es jeweils nur einen Datensatz anzeigt, kann der Nutzer alle Datensätze durchschalten. Abbildung 3.14 veranschaulicht, wie drei Datensätze durchgeschaltet wurden. Die Navigationsschaltflächen sind intelligent, daher sind sie nur bei Bedarf eingeblendet. So sind für die erste Anzeige nur die Schaltflächen NEXT und LAST sichtbar, für die letzte Anzeige die Schaltflächen FIRST und PREVIOUS.

Abbildung 3.14: Das DetailsView im alleinstehenden Modus

Eigene DetailsView-Befehle

Wie beim `GridView` kann jede Schaltfläche das Standard-Schaltflächenkommando durch Verwendung der Eigenschaften `CommandName` und `CommandArgument` emulieren. Für die Standardkommandos setzt man den `CommandName` auf `Insert`, `Cancel`, `Delete`, `New` oder `Select`. Um die Paging-Schaltflächen nachzumachen, setzen Sie `CommandName` auf PAGE und `CommandArgument` auf `First`, `Last`, `Prev` oder `Next`. Ein Beispiel:

```
<asp:LinkButton id="LinkButton1" runat="server"
  CommandName="Page" CommandArgument="First"
  Text="Erster" />
```

3.3.10 Das FormView-Steuerelement

Falls Sie mehr Flexibilität wünschen, als das `DetailsView` zur Verfügung stellt, können Sie auf ein `FormView` zurückgreifen. Es ist in seiner Funktionsweise dem `DetailsView` sehr ähnlich: Es zeigt jeweils eine einzige Zeile an, bietet aber keine Standard-Benutzerschnittstelle; Sie müssen die komplette Schnittstelle über Templates definieren. Die meisten Eigenschaften des `FormView` sind die gleichen wie für das `DetailsView`, doch unterscheidet es sich konzeptionell in einem bedeutenden Punkt: Wenn Sie die benutzerdefinierten Spalten im `DetailsView` verwenden, definieren Sie die Felder und anschließend die Templates innerhalb der Felder. Da es jedoch beim `FormView` keine Standardschnittstelle gibt, definieren Sie die Templates und fügen dort den ganzen Inhalt ein, der angezeigt werden soll, sobald dieses Template sichtbar ist. Betrachten Sie beispielsweise Abbildung 3.15, die drei Zustände einer `FormView` zeigt.

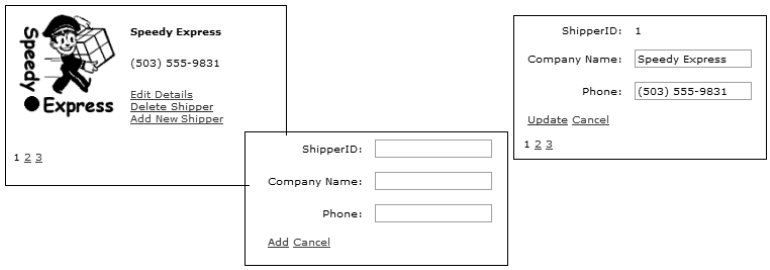


Abbildung 3.15: Das FormView-Steuerelement in den Modi Normal, Einfügen und Bearbeiten

Die `FormView`-Deklaration ist in Listing 3.19 zu sehen:

```
<asp:FormView id="FormView1" DataSourceID="dvs1" runat="server"
  DataKeyNames="ShipperID" AllowPaging="True"
  PagerSettings-Mode="Numeric">
```

Listing 3.19: Die FormView-Deklaration

Listing 3.20 zeigt das `ItemTemplate`. Sie können erkennen, dass die komplette Schnittstelle detailliert einschließlich einer Layout-Tabelle angegeben ist. Ebenfalls enthalten sind Standardserver-Steuerelemente, die nur eine lesbare Verbindung benutzen, sowie `LinkButton`-Steuerelemente für die Kommandos, die das Bearbeiten, Löschen und Hinzufügen von Zeilen ermöglichen.

```
<ItemTemplate>
  <table border="0" cellpadding="5">
  <tr>
    <td>
      <asp:Image ID="Image1" runat="server"
        Width="100" Height="123"
        ImageUrl='<%# Eval("ShipperID", "{0}.gif") %>'
        AlternateText='<%# Eval("CompanyName", "{0} Logo") %>' />
    </td>
    <td>
      <b><%# Eval("CompanyName") %></b><p />
      <%# Eval("Phone") %><p />
      <asp:LinkButton id="btnEdit" runat="server"
        CommandName="Edit" Text="Edit Details" /><br />
      <asp:LinkButton id="btnDelete" runat="server"
        CommandName="Delete" Text="Delete Shipper" /><br />
      <asp:LinkButton id="btnInsert" runat="server"
        CommandName="New" Text="Add New Shipper" />
    </td>
  </tr>
  </table>
</ItemTemplate>
```

Listing 3.20: Das FormView ItemTemplate

In Listing 3.21 sehen Sie das `EditItemTemplate` sowohl mit der nur lesbaren als auch mit der lesenden und schreibenden Verbindung. Die nur lesbare Verbindung wird für die `ShipperID` benutzt, das Primärschlüssel-Feld der Tabelle, die nicht bearbeitet werden kann. Die Spalten `CompanyName` und `Phone` verwenden `Bind`, um es zu ermöglichen, dass der vom Nutzer eingegebene Wert zurück in die Datenbank geschickt wird. Da dieses Template bei der Bearbeitung von Daten angezeigt wird, haben die Schaltflächen ihre Kommandos auf UPDATE und CANCEL gesetzt – die zwei Vorgänge lassen sich auf einer Zeile im Bearbeitungsmodus ausführen.

```
<EditItemTemplate>
  <table border="0" cellpadding="5">
    <tr>
      <td align="right">ShipperID:</td>
      <td><%# Eval("ShipperID") %></td>
    </tr>
    <tr>
      <td align="right">Company Name:</td>
      <td>
        <asp:TextBox id="txtEditName" runat="server"
             Text='<%# Bind("CompanyName") %>' />
      </td>
    </tr>
    <tr>
      <td align="right">Phone:</td>
      <td>
        <asp:TextBox id="txtEditPhone" runat="server"
          Text='<%# Bind("Phone") %>' />
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <asp:LinkButton id="btnUpdate" CommandName="Update"
          Text="Update" runat="server" />
        <asp:LinkButton id="btnCancel" CommandName="Cancel"
          Text="Cancel" runat="server" />
      </td>
    </tr>
  </table>
</EditItemTemplate>
```

Listing 3.21: Das FormView EditItemTemplate

Listing 3.22 liefert das `InsertItemTemplate`, das `Bind` für die les- und schreibbare Verbindung der `ShipperID` und des `Phone`-Felds benutzt. Obwohl es sich um eine neue Zeile handelt, wird `Bind` noch verwendet. Da die Zeile neu ist, sind die Kommandos auf ADD und CANCEL gesetzt; ADD fügt die neue Zeile hinzu, indem es die `InsertMethode` aufruft, während CANCEL die Eintragung abbricht und die Zeile zum Anzeigemodus zurückkehrt.

```
<InsertItemTemplate>
  <table border="0" cellpadding="5">
    <tr>
      <td align="right">ShipperID:</td>
      <td>
        <asp:TextBox id="txtInsertID" runat="server"
          Text='<%# Bind("ShipperID") %>' />
      </td>
    </tr>
    <tr>
      <td align="right">Company Name:</td>
      <td>
        <asp:TextBox id="txtInsertName" runat="server"
          Text='<%# Bind("CompanyName") %>' />
      </td>
    </tr>
    <tr>
      <td align="right">Phone:</td>
      <td>
        <asp:TextBox id="txtInsertPhone" runat="server"
          Text='<%# Bind("Phone") %>' />
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <asp:LinkButton id="btnAdd" CommandName="Insert"
          Text="Add" runat="server" />
        <asp:LinkButton id="btnAbandon" CommandName="Cancel"
          Text="Cancel" runat="server" />
      </td>
    </tr>
  </table>
</InsertItemTemplate>
```

Listing 3.22: Das FormView InsertItemTemplate

Obwohl mehr Quellen existieren, konnten Sie sehen, dass es immer noch keinen Bedarf für Code gibt, um Datenaktualisierungen durchzuführen. Die Techniken für das `DetailsView` und `FormView` sind sehr ähnlich und die Verwendung von Kommandos ist für all diese Daten-Steuerelemente üblich. Die deklarative Beschaffenheit der Steuerelemente erleichtert ihre Anwendung, aber es gibt Situationen, in denen Code erforderlich ist. Eine solche Situation tritt ein, wenn Sie das Zwei-Schichten-System der `SqlDataSource` vermeiden und beispielsweise eine getrennte Datensicht verwenden möchten. Das wird ausführlich in *Kapitel 4* behandelt. Die zweite Situation ist gegeben, wenn Sie erweiterte Funktionen für die Datenquelle- und Grid- Steuerelementen benötigen. Dieses Thema wird in *Kapitel 5* abgedeckt.

3.4 Zusammenfassung

In diesem Kapitel haben Sie die Grundfunktionen der `SqlDataSource` und der Datenbearbeitungs-Steuerelemente, `GridView`, `DetailsView` und `FormView` kennengelernt. Alle zusammen stellen eine einfache deklarative Methode dar, um Daten anzuzeigen und zu bearbeiten. Mit ihrer umfangreichen Einstellmöglichkeiten bieten sie sehr viel Kontrolle darüber, wie die Daten bearbeitet werden.

Sie haben erfahren, dass das `SqlDataSource`-Steuerelement Kommandos für die Beschaffung von Daten aus der Datenbank sowie Kommandos für das Senden von Datenänderungen zurück zur Datenbank zur Verfügung stellt. Der Einsatz von Parametern ermöglicht eine Filterung der angezeigten Daten entsprechend externer Kriterien. Als Beispiele seien der vom Nutzer auswählbare Wert sowie die gespeicherten Werte, die in der Session eine Rolle spielen, genannt.

Das `GridView` macht sich die Datenfunktionen der `SqlDataSource`-Steuerelemente zunutze, um eine tabellarische Ansicht mit Bearbeitungsfähigkeiten zur Verfügung zu stellen. Dagegen bieten das `DetailsView` und das `FormView` einzelne Zeilenansicht und Zeilenbearbeitung an. Im nächsten Kapitel lernen Sie das `ObjectDataSource`-Steuerelement kennen, das es Ihnen ermöglicht, auf Daten in eigenen Klassen zuzugreifen. In *Kapitel 6* werden Ihnen noch mehr erweiterte Funktionen der Steuerelemente Datenquelle und Bearbeitung begegnen.