Björn Persson

Webbsidor och databaser i ASP.NET

med VB.NET

Webbaserad applikationsutveckling

december 2005

Innehållsförteckning

Om denna sammanfattning

Denna sammanfattning beskriver hur man använder databaser och webbsidor i ASP.NET (1.1). Sammanfattningen visas även hur man kan använda filen `Web.config` för att lagra ConnectionString (och andra inställningar) för webbapplikationer.

Första exemplet visas hur man kan skapa en webbsida som visar alla poster i en tabell och hur man kan välja en post för att visa mer detaljerad information om posten för redigering på en annan webbsida. Andra exemplet visar hur man kan redigera poster i DataGrid på samma webbsida.

Beskrivningen bygger på att webbsidor använder *code behind*, d.v.s. separation av presentation och logik. Lämpligen används en utvecklingsmiljö som Microsofts Visual Studio.NET (VS.NET), som i denna sammanfattning, eller SharpDevelop (`www.sharpdevelop.com`).

Jag är givetvis tacksam för alla konstruktiva synpunkter på sammanfattningens utformning och innehåll.

Eskilstuna, december 2005

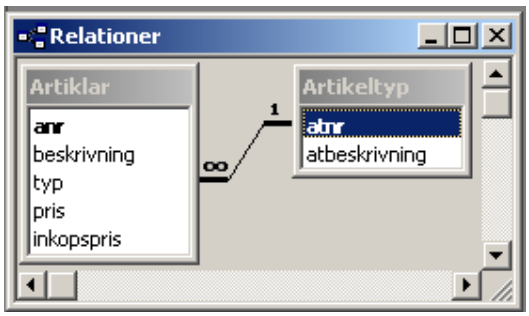
Björn Persson, e-post: `bjorn.persson@mdh.se`
Ekonomihögskolan, Mälardalens högskola
Personlig hemsida: `http://www.eki.mdh.se/personal/bpn01/`

1 Tabeller och inställningar i denna sammanfattning

Innan vi kan börja skapa webbsidorna så skapar vi tabeller och lägger till inställningar för ConnectString.

1.1 Tabeller

1.1.1 Exempel med artiklar



Exempel i denna sammanfattning använder tabeller tänkta att kunna användas i en webbaserad matvarubutik. Datatyper för attribut är i de flesta fall irrelevanta om de inte förklaras nedan.

Tabellerna är Artiklar som beskriver artiklar som kan beställas och Artikeltyp som innehåller beskrivning av de olika artiklarnas typer. Artikeltyp är en s.k. lookup-tabell, d.v.s. en tabell som vi använder för att skapa t.ex. listrutor i grafiska gränssnitt. Attributen Artiklar.typ och Atrikeltyp.atnr är av typen Tal/Långt heltal.

1.1.2 Exempel med studenter

Exempel med studenter bygger på vilka kurser studenterna läser (och vilka kurser som läses av vilka studenter, d.v.s. en många-till-många relation). Alla datatyper i exempel är av typen String.

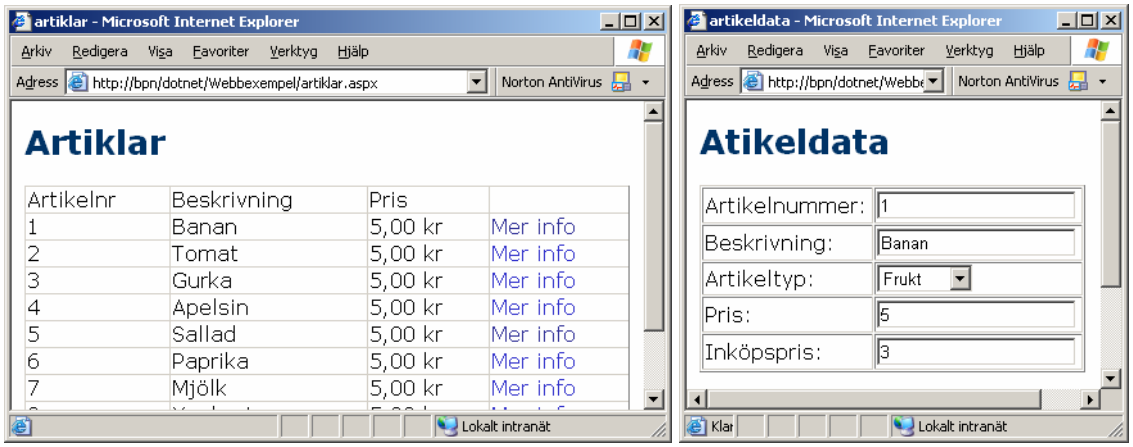
1.2 Lägga till inställningar för webbapplikation

För att bara behöva lägga till ConnectionString en gång per projekt så öppnar vi filen `Web.config`. Lägg till följande sist i filen innan avslutande `</configuration>`-tagg – ändra sökvägen till databasfilen efter behov. `<add>`-taggen ska skrivas på en rad (↵-tecknet avser att visa att raden fortsätter på nästa rad ☺).

```
  <appSettings>
    <add key="ordersystem" value="Provider=Microsoft.Jet.OLEDB.4.0; ↵
          Data Source=C:\Databaser\Ordersystem.mdb;Persist Security Info=False" />
  </appSettings>
```

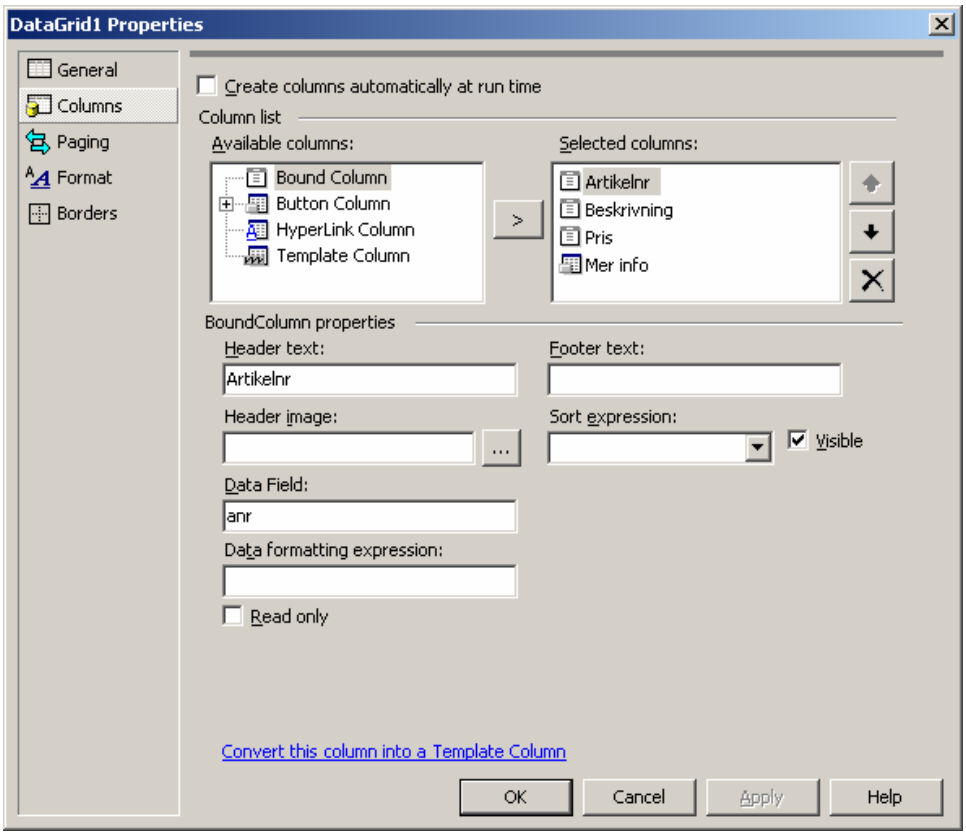
2 Visa poster med länk till mer information (artiklar)

I detta exempel skapas två webbsidor – en webbsida med lista med artiklar och länkar till mer information om en viss artikel (bild till vänster nedan) samt en webbsida med information om vald artikel för redigering (bild till höger nedan).

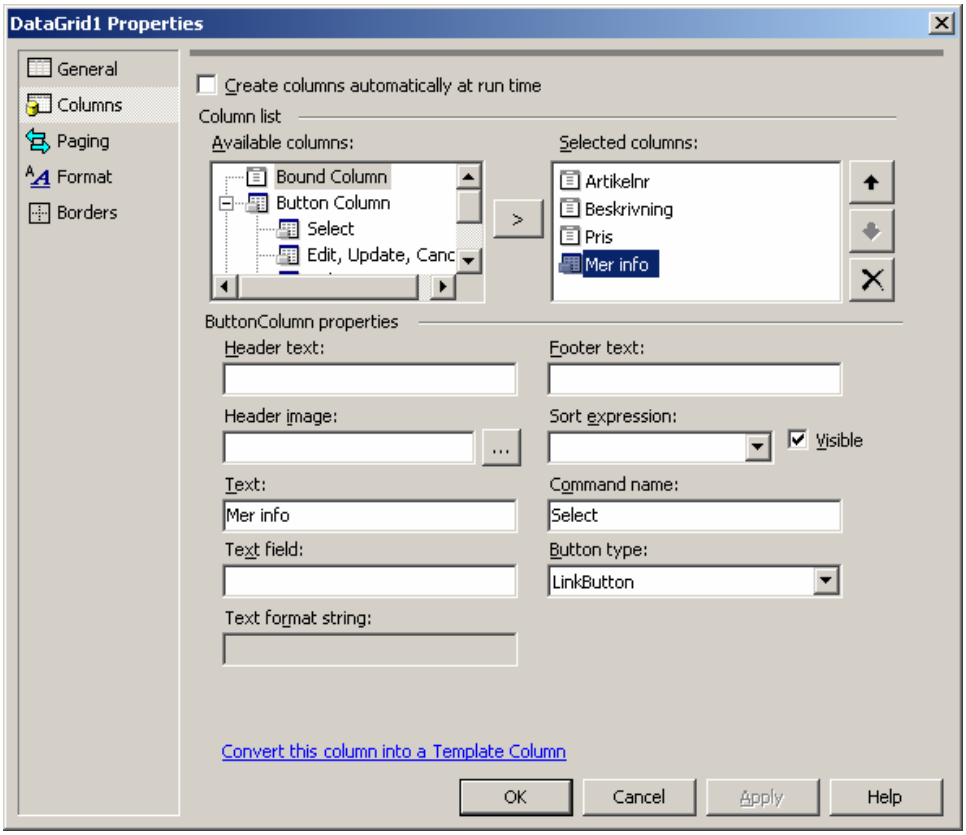


2.1 Webbsida med lista

1. Lägg till ett nytt objekt av typen Web Form (lämpligen till ett existerande webbprojekt, d.v.s. skapa ett om inte redan gjort ☺). Döp filen till `artiklar.aspx`.
2. Placera en DataGrid på webbsidan (den får automatiskt namnet `DataGrid1` om det är den första på webbsidan – byter du namn måste du ändra i kommande kod också).
3. Högerklicka på DataGrid och välj Property Builder… från menyn som visas för att visa dialogrutan DataGrid1 Properties (se bild nedan). Klicka på fliken Columns (till vänster i dialogruta) för att visa egenskaper för kolumner i DataGrid.



4. Markera Bound Column i listrutan Available columns och klicka på pilknappen för att lägga till tre alternativ i listrutan Selected columns.
5. Markera de tre alternativen i listrutan Selected columns (en åt gången) samt fyll i följande i textrutorna Header text och Data Field:
   * Artikelnr resp. anr (se bild ovan)
   * Beskrivning resp. beskrivning
   * Pris resp. pris
6. Expandera grenen Button Column i Available columns, markera Select och klicka på pilknappen för att lägga till ett alternativ i listrutan Selected columns.
7. Markera det nya alternativet i Selected columns och ändra texten i textrutan Text till Mer info (se bild nedan).



8. Klicka på OK för att stänga dialogrutan DataGrid1 Properties. Du bör nu ha en webbsida som påminner om bild till höger. (OK, så jag har lagt till en rubrik. ☺)

Artiklar

Artikelnr	Beskrivning	Pris	
Databound	Databound	Databound	Mer info
Databound	Databound	Databound	Mer info
Databound	Databound	Databound	Mer info
Databound	Databound	Databound	Mer info
Databound	Databound	Databound	Mer info

9. Dubbelklicka på webbsidans bakgrund (inte DataGrid) för att visa koden för metoden `Page_Load()`. Lägg till nedanstående kod i metoden.

```
  Dim adoConn As OleDbConnection
  Dim adoCmd As OleDbCommand
  Dim adoDA As OleDbDataAdapter
  Dim adoDS As DataSet
  Dim strConn, strSql As String

  'Hämta ConnectionString från filen Web.config
  strConn = ConfigurationSettings.AppSettings("ordersystem")

  'Om det är första gången webbsida visas...
  If Not IsPostBack Then
    adoConn = New OleDbConnection(strConn)
    strSql = "SELECT anr, beskrivning, pris FROM artiklar"
    adoCmd = New OleDbCommand(strSql, adoConn)
    adoDA = New OleDbDataAdapter(adoCmd)
    adoDS = New DataSet
    adoDA.Fill(adoDS)                'Fyll DataSet med data från DataAdapter

    DataGrid1.DataSource = adoDS     'Ange källa för DataGrid
    DataGrid1.DataBind()             'Bind data till DataGrid
  End If
```

Som alltid (☺) så börjar vi med att deklarera våra variabler – eftersom vi ska använda en DataGrid så hämtar vi lämpligen data i ett DataSet. Nästa steg är att hämta ConnectionString – den vi la till i filen `Web.config` i början på sammanfattningen. För att läsa dessa applikationsinställningar används klassen ConfigurationSettings och dess statiska (Shared i VB.NET) metod `AppSettings()`.

I nästa del av koden skapar vi objekt för förbindelse, kommando, DataAdapter och DataSet samt fyller DataSet med data. Sist kopplar vi vårt DataSet till vår DataGrid och binder data för att generera tabellen.

Du kan nu kompilera projektet och testa webbsidan om du vill.

10. Återgå till designeditorn för webbsidan (gå till ”aspx-filen”).
11. Dubbelklicka i DataGrid för att generera skalkod för metoden `DataGrid1_SelectedIndexChanged()` och lägg till nedanstående kod i metoden.

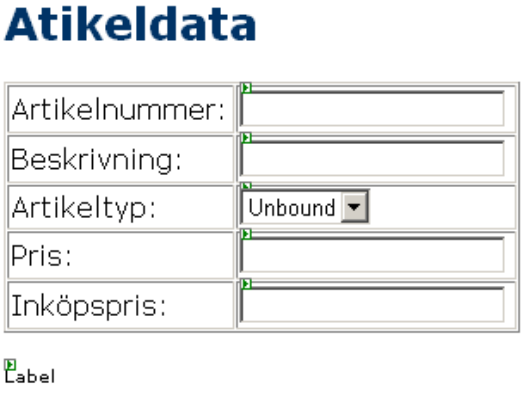
```
Dim strAnr As String

'Hämta innehåll i första kolumnen för vald post
strAnr = DataGrid1.SelectedItem.Cells(0).Text

'Skicka besökare till ny webbsida - skicka med artikelnummer i URL
Response.Redirect("artikeldata.aspx?anr=" & strAnr)
```

Här hämtar vi innehållet i första cellen (vektorn Cells är 0-baserad!), som bör innehålla artikelnumret för vald artikel. Sen använder vi metoden `Redirect()` för att skicka besökarens webbläsare till en annan webbsida – webbsidan vi ska skapa härnäst.

Innan vi kan gå vidare och testa ovanstående metod så bör vi göra nästa webbsida – `artikeldata.aspx`.

2.2 Webbsida med detaljer om vald post

1. Lägg till ytterligare en webbsida av typen Web Form. Döp webbsidan till `artikeldata.aspx`.
2. Placera fyra textrutor och en listruta (*DropDownList*) med namnen `txtAnr`, `txtBeskrivning`, `txtPris`, `txtInkopspris` resp. `ddlTyp`. För att visa felmeddelande kan vi även placera en etikett (*label*) under DataGrid. Till höger visas en bild med exempel hur webbsida skulle kunna se ut (i vilken jag använt en tabell).
3. Dubbelklicka på webbsidans bakgrund för att visa koden för metoden `Page_Load()` och lägg till nedanstående kod i metoden.




```
Dim adoConn As OleDbConnection
Dim strConn, strAnr As String
Dim intAtnr As Integer

'Hämta ConnectionString från filen Web.config
strConn = ConfigurationSettings.AppSettings("ordersystem")
```

```
'Skapa Connection-objekt och öppna förbindelse
adoConn = New OleDbConnection(strConn)
adoConn.Open()

'Fyll textrutor med data - nummer på artikelnummer returneras
intAtnr = FyllTextrutor(adoConn)

'Fyll listruta med värden - skicka artikeltyp som ska vara vald
FyllArtikeltypLista(adoConn, intAtnr)
```

Precis som i förra webbsidan så hämtas för ConnectionString från filen `Web.config`. Därefter öppnas förbindelse till databasen – en förbindelse som vi kommer använda för att hämta data till textrutor liksom för att fylla listruta med data (därav öppnas den i `Page_Load()`). Därefter anropas två metoder, `FyllTextrutor()` och `FyllArtikeltypLista()`, för att fylla textrutor och listruta – metoder som visas nedan. Till bägge metoder skickas förbindelse (Connection-objekt) som parameter till metodanropen.

Den första metoden, `FyllTextrutor()`, hämtar artikelnummer från URL som webbsida begärdes med för att hämta övrig data om artikeln från databasen. Och den andra metoden, `FyllArtikeltypLista()`, används för att fylla listruta med artikeltyper med data från databas. Denna senare metod använder nummer på artikeltyp (som första metoden returnerade) för att avgöra om något alternativ i listrutan ska vara förvalt.

```
Public Function FyllTextrutor(ByVal adoConn As OleDbConnection) As Integer
  Dim adoCmd As OleDbCommand
  Dim adoReader As OleDbDataReader
  Dim strAnr, strSql, strBeskrivning As String
  Dim decPris, decInkopspris As Decimal
  Dim intAtnr As Integer

  intAtnr = -1                    'Negativt markerar ett ogiltigt värde

  strAnr = Request("anr")         'Hämta artikelnummer från URL (som sträng)

  'Om artikelnummer skickats – hämta data från DB och fyll textrutor
  If (Not strAnr Is Nothing) And (strAnr <> "") Then
    'Skapa SQL-sats (har gjorts i Access :-)
    strSql = "SELECT Artiklar.anr, Artiklar.beskrivning, " _
      & "Artiklar.pris, Artiklar.inkopspris, Artikeltyp.atnr " _
      & "FROM Artikeltyp INNER JOIN " _
      & "Artiklar ON Artikeltyp.atnr = Artiklar.typ WHERE anr=" _
      & strAnr

    adoCmd = adoCOnn.CreateCommand()
    adoCmd.CommandText = strSql

    Try
      adoReader = adoCmd.ExecuteReader(CommandBehavior.CloseConnection)

      'Om det finns en post – läs attribut från post
      If adoReader.Read() Then
        strBeskrivning = adoReader.GetString(adoReader.GetOrdinal("beskrivning"))
        decPris = adoReader.GetDecimal(adoReader.GetOrdinal("pris"))
        decInkopspris = adoReader.GetDecimal(adoReader.GetOrdinal("inkopspris"))
        intAtnr = adoReader.GetInt32(adoReader.GetOrdinal("atnr"))
      End If

      adoReader.Close()
    Catch ex As OleDbException     'Om fel - skriv ut i etikett
      Label1.Text = ex.Message
    End Try

    'Fyll textrutor med ev. data från databas
    txtAnr.Text = strAnr
    txtBeskrivning.Text = strBeskrivning
    txtPris.Text = decPris.ToString
    txtInkopspris.Text = decInkopspris.ToString
  End If
```

```
    Return intAtnr                    'Returnera nummer på artikeltyp
  End Function
```

Först i metod börjar vi med att tilldela ett negativt värde på variabel med nummer på artikeltyp – ett värde som ska visa att det är ett ogiltigt värde. Så om vi inte kan hitta post med artikelnummer, vilket vi läser härnäst, så kommer detta negativa värde att returneras för att visa att vi inte hittat posten. Vi läser artikelnummer som vi vill visa post för från URL som webbsida begärdes med. Existensen av detta värde kontrolleras för om vi ska hämta post från databas eller inte.

I ovanstående metod använder vi en DataReader för att läsa data från posten då vi endast behöver läsa data (mer effektivt än ett DataSet). Metoden `GetOrdinal()` i DataReader returnerar index för kolumn som vi skickar som parameter till metod. Detta värde kan vi sen använda när vi anropar metoder som `GetString()` och `GetDecimal()` för att slippa behöva komma ihåg ordningen på kolumner i tabell (vi måste dock komma ihåg namnen på kolumnerna ☺).

```
  Public Sub FyllArtikeltypLista(ByVal adoConn As OleDbConnection, _
      Optional ByVal intAtnr As Integer = -1)
    Dim adoCmd As OleDbCommand
    Dim adoDA As OleDbDataAdapter
    Dim adoDS As DataSet
    Dim strSql As String

    'Hämta data och fyll listruta med data
    strSql = "SELECT * FROM artikeltyp ORDER BY atbeskrivning"
    adoCmd = adoConn.CreateCommand()
    adoCmd.CommandText = strSql
    adoDA = New OleDbDataAdapter(adoCmd)
    adoDS = New DataSet
    adoDA.Fill(adoDS)   'Fyll DataSet med data från DataAdapter

    ddlTyp.DataSource = adoDS   'Ange källa till listruta
    'Ange kolumn som ska visas (text) och vilket som ska skickas (value)
    ddlTyp.DataTextField = "atbeskrivning"
    ddlTyp.DataValueField = "atnr"

    'Om nummer på artikeltyp inte är negativt - sätt som valt alternativ
    If intAtnr > 0 Then
      ddlTyp.SelectedValue = intAtnr  'Sätt valt alternativ
    End If

    ddlTyp.DataBind()   'Bind data till DataSet
  End Sub
```

I metoden ovan hämtar vi data i ett DataSet som vi sen tilldelar till listrutans egenskap DataSource. För att ange vilket värde som ska visas tilldelar vi namnet på kolumn i tabell till listrutans egenskapen DataTextField och för vilket värde som ska skickas (när formulär skickas) tilldelar vi namn på kolumn till egenskapen DataValueField.

Genom att använda egenskapen SelectedValue i vår listruta (ddlTyp) så kan vi ange vilket alternativ i listrutan som ska vara valt. Vi måste dock först kontrollera så att värdet inte är negativt – ett värde som används för att ange att det är ogiltigt.

2.3 Vidare utveckling

Halva meningen med hemsidor är att de kan utvecklas med tiden. ☺ I detta fall så kan utveckling av dessa hemsidor göras för att spara uppdateringar i webbsidan med detaljer.

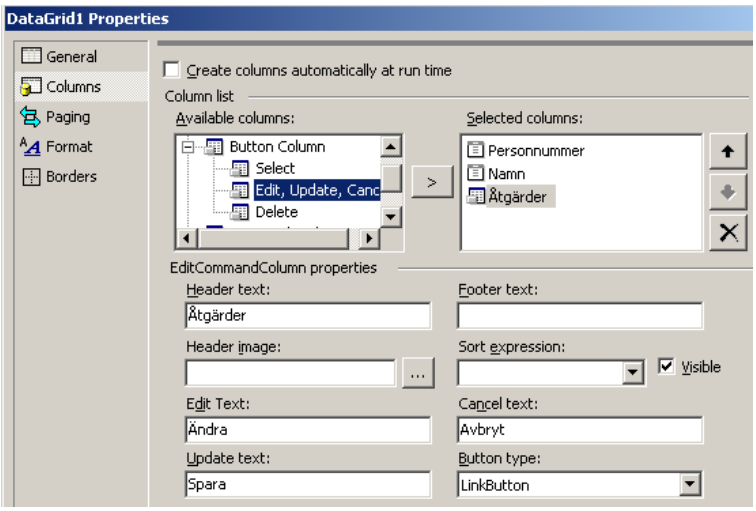
3 Redigera poster i DataGrid

I detta exempel visas hur man kan redigera poster i en DataGrid (istället för i en separat webbsida som i exempel ovan).

3.1 Konfigurera DataGrid

1. Lägg till en ny webbsida, med t.ex. namnet `studenter.aspx`, till projekt.
2. Lägg till en DataGrid till webbsida.
3. Använd DataGrid:s *Property Builder* för att lägga till 2 bundna kolumner (Personnummer/pnr och Namn/namn) samt en knappkolumn av typen *Edit, Update, Cancel* till DataGrid (se bilder till höger). Ändra rubriken (Header) för knappkolumn till Åtgärder samt rubriker för länkar till Ändra, Avbryt resp. Spara (se bild till höger).

Personnummer	Namn	Åtgärder
Databound	Databound	Ändra
Databound	Databound	Ändra
Databound	Databound	Ändra
Databound	Databound	Ändra
Databound	Databound	Ändra



4. Dubbelklicka på webbsidans bakgrund för att öppna koden för metoden `Page_Load()` och lägg till koden nedan i metoden.

```
If Not IsPostBack Then
  GridBind()
End If
```

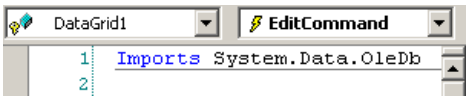
I `Page_Load()` anropas en metod `GridBind()` som hämtar data från databas och fyller DataGrid. Koden för databasåtkomst har flyttats till en separat metod då koden anropas i flera metoder (se metoder nedan) och nedan visas koden för `GridBind()`.

```
Private Sub GridBind()
  Dim adoConn As OleDbConnection, adoCmd As OleDbCommand
  Dim adoDA As OleDbDataAdapter, adoDS As DataSet
  Dim strSql As String

  strSql = "SELECT * FROM studenter"

  adoConn = New OleDbConnection(cstrConn)  'Skapa förbindelse till DB
  adoCmd = adoConn.CreateCommand()         'Skapa kommandoobjekt
  adoCmd.CommandText = strSql              'Ange kommando att utföra
  adoDA = New OleDbDataAdapter(adoCmd)     'Skapa DataAdapter-objekt
  adoDS = New DataSet                      'Skapa DataSet-objekt
  adoDA.Fill(adoDS)                        'Fyll DataSet från DataAdapter

  DataGrid1.DataSource = adoDS             'Ange datakälla för DataGrid
  DataGrid1.DataBind()                     'Bind data till DataGrid
```

```
End Sub
```

3.2 Metoder för redigering och uppdatering

5. I kodfönstret, välj DataGrid1 i listrutan Class Name (överst till vänster i kodfönstret) och sen metoden EditCommand i listrutan Method Name (överst till höger). Detta lägger till metoder `DataGrid1_EditCommand()` – fyll i koden nedan som saknas (metodsignaturen ska skrivas på en rad – pilen visar på en ofrivillig radbrytning).




```
Private Sub DataGrid1_EditCommand(ByVal source As Object, ByVal e As ↵
  System.Web.UI.WebControls.DataGridCommandEventArgs) Handles DataGrid1.EditCommand
   DataGrid1.EditItemIndex = e.Item.ItemIndex
   GridBind()
End Sub
```

Metoden ovan används för att hantera DataGrid:s händelse för redigering (*edit* – andra är uppdatera/spara, ta bort, avbryt och välj). För att påbörja redigering så sätter vi egenskapen `EditItemIndex` till index som valdes för redigering, vilket vi kan få reda på från argumentet e som skickats till metod. Sen anropas vår metod `GridBind()` för att ”binda om” data till DataGrid. Kolumnerna i vald post kommer visas i textrutor så att dess värden kan redigeras (se bild till höger).

Personnummer	Namn	Åtgärder
7123456789	Oscar Ohlsson	Ändra
8123456789	Nisse Hult	Spara Avbryt
9123456789	Pelle P	Ändra
1123456789	Sune S	Ändra
2123456789	Sture	Ändra

Eftersom personnummer är en primärnyckel i tabellen så kan vi gå tillbaka till DataGrid:s *Property Builder* och bocka för kryssrutan *Read only* för den bundna kolumnen Personnummer. Därmed kommer inte kolumnen för personnummer att visas i en textruta, så som jag gjort i detta exempel (se bild ovan).

Nästa steg är att lägga till metoder för att avbryta redigering samt uppdatera/spara ändringar.

6. I kodfönstret, välj DataGrid1 i listrutan Class Name (överst till vänster i kodfönstret) samt sen metoden CancelCommand och UpdateCommand i listrutan Method Name (överst till höger). Detta lägger till metoderna `DataGrid1_CancelCommand()` och `DataGrid1_UpdateCommand()` – fyll i koden nedan som saknas.

```
Private Sub DataGrid1_CancelCommand(ByVal source As Object, ByVal e As ↵
   System.Web.UI.WebControls.DataGridCommandEventArgs) Handles ↵
    DataGrid1.CancelCommand
  DataGrid1.EditItemIndex = -1   'Avbryt redigering
  GridBind()
End Sub

Private Sub DataGrid1_UpdateCommand(ByVal source As Object, ByVal e As ↵
   System.Web.UI.WebControls.DataGridCommandEventArgs) Handles ↵
   DataGrid1.UpdateCommand
  Dim adoConn As OleDbConnection, adoCmd As OleDbCommand
  Dim strSql, strPnr, strNamn As String

  strPnr = e.Item.Cells(0).Text    'Hämta personnummer från 1:a kol. i vald post
    'Hämta först kontroll i 2:a kol., konvertera till textruta och hämta innehåll
  strNamn = CType(e.Item.Cells(1).Controls(0), TextBox).Text

    'SQL-sats för att uppdatera post
  strSql = "UPDATE studenter SET namn='" & strNamn & "' WHERE pnr='" _
     & strPnr & "'"
```

```
    adoConn = New OleDbConnection(cstrConn)  'Öppna förbindelse och uppdatera post
    adoConn.Open()
    adoCmd = New OleDbCommand(strSql, adoConn)
    adoCmd.ExecuteNonQuery()
    adoConn.Close()                     'Stäng databas så att transaktion avslutas

    e.Item.Cells(1).Text = strNamn
    DataGrid1.EditItemIndex = -1        'Avbryt redigering
    GridBind()
  End Sub
```

För att avbryta redigering av post i DataGrid så sätts egenskapen `EditItemIndex` till ett ogiltigt värde, d.v.s. negativt (`-1` här). Detta görs i både metod för att avbryta redigering och efter att vi uppdaterat posten.

För att kunna uppdatera post så måste vi hämta värdena för posten. Detta görs genom att använda argumentet e som skickas till metod och som innehåller en instans av DataGridItem, d.v.s. vald post. En DataGridItem innehåller en vektor Cells med innehållet i respektive kolumn i DataGrid. Innehållet är antingen text, som personnummer i exempel, eller en kontroll, textruta i exempel. Om det är en kontroll så måste kontroll konverteras till rätt typ, vilket vi gör med funktionen `CType()`. När vi har alla värden så kan vi uppdatera posten i databasen.

3.3 Metod för radering

Dags att lägga till en kolumn i DataGrid så att vi kan radera poster.

7. Högerklicka på DataGrid och välj Property Builder… från menyn som visas.
8. Klicka på fliken Columns och lägg till en knappkolumn av typen Delete.
9. Ändra länktexten (*Text*) till Radera.
10. Klicka på OK för att stänga dialogrutan. Vi bör ha en DataGrid med ett utseende likt den i bild till höger.

Personnummer	Namn	Åtgärder	
7123456789	Oscar Ohlsson	Ändra	Radera
8123456789	Nisse Hult	Ändra	Radera
9123456789	Pelle P	Ändra	Radera
1123456789	Sune S	Ändra	Radera
2123456789	Sture	Ändra	Radera

11. Lägg till en etikett i webbformulär (för felmeddelande) och radera innehållet i dess egenskap Text.
12. Växla till kodfönstret. Välj DataGrid1 i Class Name och sen DeleteCommand i Method Name för att skapa metoden `DataGrid1_DeleteCommand()`.
13. Lägg till nedanstående kod i metoden.

```
  Dim adoConn As OleDbConnection, adoCmd As OleDbCommand
  Dim strSql, strPnr As String

  Label1.Text = "Student raderades."  'Meddela besökare om resultat
  strPnr = e.Item.Cells(0).Text       'Hämta personnr för person att ta bort
  strSql = "DELETE FROM  studenter WHERE pnr='" & strPnr & "'"

  adoConn = New OleDbConnection(cstrConn)
  adoConn.Open()
  adoCmd = New OleDbCommand(strSql, adoConn)

  Try
    adoCmd.ExecuteNonQuery()
  Catch oe As OleDbException
    Label1.Text = "<b>FEL:</b> Studenten registrerad på kurs och kan inte raderas!"
  End Try
```

```
    adoConn.Close() 'Stäng databas så att transaktion avslutas
    GridBind()      'Bind data till DataGrid
```

Vi börjar med att meddela besökare om resultatet, d.v.s. resultatet om allt går bra ☺ (se mer nedan). Sen hämtar vi personnummer för student som ska raderas genom att hämta värdet i första kolumnen för aktuell post. Därefter skapar vi SQL-sats att utföra med personnumret.

Nästa steg är databasåtkomsten, d.v.s. vi skapar objekt för förbindelse och kommando. Eftersom uppdateringar av databas, radering i detta fall, kan generera fel så placerar vi utförandet av kommandot inom Try-Catch-block.[1] Om fel uppstår här så meddelas besökare om att student är registrerad på en kurs, d.v.s. vi ersätter det positiva resultatet i etikett som sattes i början på metoden. Siste stänger vi förbindelse till databas och binder om data till DataGrid.

[1] … något som vi borde gjort även vid uppdatering ovan. ☺