

Databundna kontroller

Upphovsrätt för detta verk

Detta verk är framtaget i anslutning till kursen ASP.NET Web Forms vid Linnéuniversitetet.

Du får använda detta verk så här:

Allt innehåll i detta verk av Mats Looch, förutom Linnéuniversitetets logotyp och symbol samt ikoner, bilder och fotografier, är licensierad under:



Creative Commons Erkännande 4.0 Internationell licens.

<http://creativecommons.org/licenses/by/4.0>

Det betyder att du i icke-kommersiella syften får:

- kopiera hela eller delar av innehållet
- sprida hela eller delar av innehållet
- visa hela eller delar av innehållet offentligt och digitalt
- konvertera innehållet till annat format
- du får även göra om innehållet

Om du förändrar innehållet så ta inte med Linnéuniversitetets logotyp och symbol samt ikoner och fotografier i din nya version!

Vid all användning måste du ange källan: "Linnéuniversitetet – ASP.NET Web Forms" och en länk till <https://coursepress.lnu.se/kurs/aspnet-web-forms> och till Creative Common-licensen här ovan.

Vad är en databunden kontroll?



- ✓ Du använder databundna kontroller till att skapa ett gränssnitt för arbete med data.
- ✓ Databundna kontroller använder du till att presentera och redigera data från databaser, data från XML-filer eller data från vilken annan typ av datakälla som helst.
- ✓ Databundna kontroller kan delas in i tre huvudgrupper efter hur de presenterar datat:
 - Listor
 - Tabeller
 - Hierarkier (tar denna kurs inte upp)

Databundna listkontroller

BulletedList

- Wilma
- Maja
- Ella
- Emma
- Julia
- Alice
- Alva
- Linnea
- Ida
- Ebba

CheckBoxList

- ☐ Wilma
- ☐ Maja
- ☐ Ella
- ☒ Emma
- ☒ Julia
- ☐ Alice
- ☐ Alva
- ☒ Linnea
- ☐ Ida
- ☐ Ebba

RadioButtonList

- ☐ Wilma
- ☐ Maja
- ☐ Ella
- ☒ Emma
- ☐ Julia
- ☐ Alice
- ☐ Alva
- ☐ Linnea
- ☐ Ida
- ☐ Ebba

DropDownList

Wilma ▼

ListBox

Wilma ▲
Maja ▢
Ella ▢
Emma ▼

- ✓ Du använder listkontroller till att presentera enkla listor.
- ✓ Alla fem ärver från samma basklass – **ListControl**.
- ✓ CheckBoxList- och RadioButtonList-kontrollerna renderas som input-element i en tabell. Vill du inte ha en tabell sätter du egenskapen **RepeatLayout** till **Flow**.
- ✓ Du kan binda data, t.ex. en array, till kontrollerna programmatiskt med hjälp av egenskapen **DataSource** och metoden **DataBind**.

```
string[] girlNames = { "Wilma", "Maja", "Ella", "Emma", "Julia",  
                        "Alice", "Alva", "Linnea", "Ida", "Ebba" };  
GirlNamesBulletedList.DataSource = girlNames;  
GirlNamesBulletedList.DataBind();
```

Databundna tabellkontroller

GridView

Name	Rank	Number
Wilma	1	922
Maja	2	907
Ella	3	860
Emma	4	835
Julia	5	8
Alice	6	8
Alva	7	7
Linnea	8	7
Ida	9	7
Ebba	10	7

DataList

Wilma [1, 922] Maja [2, 907] Ella [3, 860] Emma [4, 835]
 Julia [5, 816] Alice [6, 806] Alva [7, 781] Linnea [8, 778]
 Ida [9, 763] Ebba [10, 758]

FormView

Name: Wilma
 Rank: 1
 Number: 922
 1 2 3 4 5 6 7 8 9 10

Listview

Name: Wilma Name: Maja Name: Ella
 Rank: 1 Rank: 2 Rank: 3
 Number: 922 Number: 907 Number: 860
 First Previous Next Last

Repeater

Wilma [1, 922] Maja [2, 907] Ella [3, 860] Emma [4, 835] Julia [5, 816]
 Alice [6, 806] Alva [7, 781] Linnea [8, 778] Ida [9, 763] Ebba [10, 758]

- ✓ Databundna tabellkontroller behöver nödvändigtvis inte renderas som **table**-element.
- ✓ Med databundna tabellkontroller kan du presentera och modifiera data från databaser eller andra datakällor.
- ✓ Det finns sex databundna tabellkontroll som kan delas in i två huvudkategorier:
 - De som kan visa flera dataposter i taget.
 - GridView, DataList, ListView och Repeater.
 - De som kan visa en datapost i taget.
 - DetailsView och FormView.
- ✓ Du kan binda en datakälla till kontrollen deklarativt med hjälp av en DataSource-kontroll, t.ex. en ObjectDataSource-kontroll som du kopplar till en metod i en affärslogikklass.

Två sätt att binda data till en databunden kontroll

- ✓ Data som binds till en databunden kontroll kan t.ex. exponeras via metoder i klasser som exempelvis returnerar en array eller samling av referenser till objekt.
- ✓ Programmatiskt
 - Datakällan binds till kontrollen i "code-behind"-filen i regel i samband med en "get" av sidan.
- ✓ Deklarativt
 - Datakällan binds till kontrollen i aspx-sidan t.ex. med en ObjectDataSource-kontroll som använder en metod i en affärslagerklass.

```
public static List<BabyName> GetBabyNames() { ... }

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        GridView1.DataSource = BabyName.GetBabyNames();
        GridView1.DataBind();
    }
}
```

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server" SelectMethod="GetBabyNames"
    TypeName="BabyName" />
<asp:GridView ID="GridView1" runat="server" AllowPaging="True" DataSourceID="ObjectDataSource1"
    CssClass="nameGrid" GridLines="None" OnRowDataBound="GridView1_RowDataBound">
    <HeaderStyle CssClass="header" />
</asp:GridView>
```

Databindningsuttryck

- ✓ Databindningsuttryck är speciella uttryck som inte utvärderas då koden körs. Ett databindningsuttryck skrivs mellan <%# och %>.
- ✓ Då du binder en kontroll till data via ett DataSource-objekt, eller om du gör det i "code-behind"-filen med egenskapen DataSource och metoden DataBind, så skapas databindningshändelser vilket leder till att databindningsuttryck utvärderas.
- ✓ Använder du en t.ex. en Repeater-kontroll måste du skapa en mall ("item template") innehållande databindningsuttryck för att kunna presentera data från en datakälla.

```
<asp:Repeater ID="Repeater1" runat="server" DataSourceID="ObjectDataSource1">
  <ItemTemplate>
    <strong>
      <%# Eval("Name") %></strong> <em>[<%# Eval("Rank") %>,
      <%# Eval("Number") %>] </em>
    </ItemTemplate>
  </asp:Repeater>
```

Repeater

Wilma [1, 922] Maja [2, 907] Ella [3, 860] Emma [4, 835] Julia [5, 816]
Alice [6, 806] Alva [7, 781] Linnea [8, 778] Ida [9, 763] Ebba [10, 758]

Att binda data till en "template"

- ✓ ObjectDataSource-kontrollen bestämmer med egenskaperna/attributen TypeName och SelectMethod vilken klass och metod som ska användas som datakälla.
- ✓ Databindningsuttrycken utvärderas för var och en av posterna som binds till kontrollen.
- ✓ Metoden Eval utvärderar posten (dataobjektet) som bundits för att hitta en egenskap med ett angivet namn, den egenskap som ger värdet som ska presenteras.

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server" SelectMethod="GetPopularGirlNames"
    TypeName="BabyGirlName" />
<asp:Repeater ID="Repeater1" runat="server" DataSourceID="ObjectDataSource1">
    <ItemTemplate>
        <strong>
            <%=# Eval("Name") %></strong> <em>[<%=# Eval("Rank") %>,
            <%=# Eval("Number") %>]</em>
        </ItemTemplate>
    </asp:Repeater>
```

```
public class BabyGirlName
{
    public string Name { get; set; }
    public int Rank { get; set; }
    public int Number { get; set; }

    public BabyGirlName(string name, int rank, int number) {...}

    public static List<BabyGirlName> GetPopularGirlNames() {...}
}
```

```
public static List<BabyGirlName> GetPopularGirlNames()
{
    List<BabyGirlName> names = new List<BabyGirlName>();
    names.Add(new BabyGirlName("Wilma", 1, 922));
    names.Add(new BabyGirlName("Maja", 2, 907));
    names.Add(new BabyGirlName("Ella", 3, 860));
    names.Add(new BabyGirlName("Emma", 4, 835));
    names.Add(new BabyGirlName("Julia", 5, 816));
    names.Add(new BabyGirlName("Alice", 6, 806));
    names.Add(new BabyGirlName("Alva", 7, 781));
    names.Add(new BabyGirlName("Linnea", 8, 778));
    names.Add(new BabyGirlName("Ida", 9, 763));
    names.Add(new BabyGirlName("Ebba", 10, 758));
}
```


Formatering av databundna tabellkontroller

- ✓ Du har stora möjligheter att påverka formateringen av de databundna tabellkontrollerna. Kontrollerna, utom ListView och Repeater, har flera egenskaper som har med formatering att göra – **ANVÄND INTE DEM** – använd istället "cascading style sheets" för formatering. Du kan via egenskapen `CssClass` formatera olika delar av kontrollerna.

```
<h1>
  Formaterad GridView</h1>
<asp:GridView ID="GridView2" runat="server" DataSourceID="ObjectDataSource1" CssClass="grid"
  GridLines="None">
  <HeaderStyle CssClass="header" />
  <RowStyle CssClass="row" />
  <AlternatingRowStyle CssClass="alternating" />
</asp:GridView>
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

/// <summary>
/// Summary description for Baby
/// </summary>
public class BabyGirlName
{
    public string Name { get; set; }
    public int Rank { get; set; }
    public int Number { get; set; }

    public BabyGirlName(string name, int rank, int number)
    {
        Name = name;
        Rank = rank;
        Number = number;
    }
}

public static List<BabyGirlName> names
{
    List<BabyGirlName> names = new List<BabyGirlName>();
    names.Add(new BabyGirlName("Wilma", 1, 922));
    names.Add(new BabyGirlName("Maja", 2, 907));
    names.Add(new BabyGirlName("Ella", 3, 860));
    names.Add(new BabyGirlName("Emma", 4, 835));
    names.Add(new BabyGirlName("Julia", 5, 816));
    names.Add(new BabyGirlName("Alice", 6, 806));
    names.Add(new BabyGirlName("Alva", 7, 781));
    names.Add(new BabyGirlName("Linnea", 8, 778));
    names.Add(new BabyGirlName("Ida", 9, 763));
    names.Add(new BabyGirlName("Ebba", 10, 758));
}
```

```
.grid
{
    font-family: Arial, Helvetica, sans-serif;
    font-size: 14px;
}

.grid th, .grid td
{
    padding: 5px;
}

.grid .header
{
    text-align: left;
    color: #fff;
    background-color: #000080;
}

.row td, .alternating td
{
    border-bottom: solid 1px #000080;
}

.grid .alternating
{
    background-color: #f0f0f0;
}
```

```
<table class="grid" cellspacing="0" border="0" id="GridView2" style="width: 100%; text-align: left;">
  <tr class="header">
    <th scope="col">
      Name
    </th>
    <th scope="col">
      Rank
    </th>
    <th scope="col">
      Number
    </th>
  </tr>
  <tr class="row">
    <td>
      Wilma
    </td>
    <td>
      1
    </td>
    <td>
      922
    </td>
  </tr>
  <tr class="alternating">
    <td>
      Maja
    </td>
    <td>
      2
    </td>
    <td>
      907
    </td>
  </tr>
  <tr class="row">
    <td>
      Ella
    </td>
    <td>
      3
    </td>
    <td>
      860
    </td>
  </tr>
  <tr class="alternating">
    <td>
      Emma
    </td>
    <td>
      4
    </td>
    <td>
      835
    </td>
  </tr>
  <tr class="row">
    <td>
      Julia
    </td>
    <td>
      5
    </td>
    <td>
      816
    </td>
  </tr>
  <tr class="alternating">
    <td>
      Alice
    </td>
    <td>
      6
    </td>
    <td>
      806
    </td>
  </tr>
  <tr class="row">
    <td>
      Alva
    </td>
    <td>
      7
    </td>
    <td>
      781
    </td>
  </tr>
  <tr class="alternating">
    <td>
      Linnea
    </td>
    <td>
      8
    </td>
    <td>
      778
    </td>
  </tr>
  <tr class="row">
    <td>
      Ida
    </td>
    <td>
      9
    </td>
    <td>
      763
    </td>
  </tr>
  <tr class="alternating">
    <td>
      Ebba
    </td>
    <td>
      10
    </td>
    <td>
      758
    </td>
  </tr>
</table>
```

Formaterad GridView

Name	Rank	Number
Wilma	1	922
Maja	2	907
Ella	3	860
Emma	4	835
Julia	5	816
Alice	6	806
Alva	7	781
Linnea	8	778
Ida	9	763
Ebba	10	758

GridView och händelsen RowDataBound

- ✓ GridView-kontrollen skapar händelser för varje rad som skapas och databinds – RowCreated och RowDataBound. Innehåller datakällan 20 objekt så skapas 20 RowCreated- och RowDataBound-händelser.
- ✓ Genom att skapa en hanterarmetod för händelsen RowDataBound kan du påverka flera olika saker för den enskilda raden i tabellen, t.ex. kan du utifrån dataobjektet som binds till raden bestämma hur raden ska formateras.

```
using System;
using System.Collections.Generic;

public enum Genders { Female, Male };

/// <summary>
/// Summary description for BabyName.
/// </summary>
public class BabyName : IComparable<BabyName>
{
    public string Name { get; set; }
    public Genders Gender { get; set; }
    public int Number { get; set; }

    public BabyName(string name, Genders gender, int number)
    {
        Name = name;
        Gender = gender;
        Number = number;
    }
}

<asp:GridView ID="GridView1" runat="server" AllowPaging="True" DataSourceID="ObjectDataSource1"
    CssClass="nameGrid" GridLines="None" OnRowDataBound="GridView1_RowDataBound">
    <HeaderStyle CssClass="header" />
</asp:GridView>

protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        BabyName babyName = (BabyName)e.Row.DataItem;

        if (babyName.Gender == Genders.Female)
        {
            e.Row.CssClass = "female";
        }
        else
        {
            e.Row.CssClass = "male";
        }
    }
}
```

```
.nameGrid .female
{
    border-bottom: solid 1px #000080;
}

.nameGrid .female
{
    background-color: #ffecec;
}

.nameGrid .male
{
    background-color: #ecffff;
}
```

Name	Number
William	1166
Lucas	1065
Elias	1047
Oscar	996
Hugo	964
Wilma	922
Maja	907
Viktor	892
Filip	868
Ella	860

1 2

DataList, ListView och Repeater...

- ✓ ...fungerar på liknande sätt som GridView och händelsen RowDataBound, men...
- ✓ ...händelsen heter ItemDataBound för dessa kontroller.

```
using System;
using System.Collections.Generic;
```

```
public enum Genders { Female
```

```
/// <summary>
```

```
/// Summary description fo
```

```
/// </summary>
```

```
public class BabyName : IC
```

```
{
```

```
    public string Name { g
```

```
    public Genders Gender
```

```
    public int Number { ge
```

```
}
```

```
    public BabyName(string name, Genders gen
```

```
{
```

```
        Name = name;
```

```
<asp:DataList ID="DataList1" runat="server" OnItemDataBound="DataList1_ItemDataBound"
```

```
    DataSourceID="ObjectDataSource1" CssClass="nameDataList" RepeatColumns="3"
```

```
    RepeatDirection="Horizontal">
```

```
<ItemTemplate>
```

```
<asp:Label ID="NameLabel" runat="server" Text="<%# Eval("Name") %>" />
```

```
<asp:Label ID="NumberLabel" runat="server" Text="<%# Eval("Number", "{0}") %>" />
```

```
</ItemTemplate>
```

```
</asp:DataList>
```

```
protected void DataList1_ItemDataBound(object sender, DataListItemEventArgs e)
```

```
{
```

```
    if (e.Item.ItemType == ListItemType.Item ||
```

```
        e.Item.ItemType == ListItemType.AlternatingItem)
```

```
    {
```

```
        BabyName babyName = (BabyName)e.Item.DataItem;
```

```
        if (babyName.Gender == Genders.Female)
```

```
        {
```

```
            e.Item.CssClass = "female";
```

```
        } else
```

```
        {
```

```
            e.Item.CssClass = "male";
```

```
        }
```

```
    }
```

```
.nameDataList .female
{
    background-color: #ffecec;
}
.nameDataList .male
{
    background-color: #ecffff;
}
```

William (1166)	Lucas (1065)	Elias (1047)
Oscar (996)	Hugo (964)	Wilma (922)
Maja (907)	Viktor (892)	Filip (868)
Ella (860)	Erik (857)	Emil (840)
Isak (838)	Emma (835)	Julia (816)
Alice (806)	Alva (781)	Linnea (778)
Ida (763)	Ebba (758)	

...och mycket mer finns att läsa...

- ✓ ...om "Data Binding" i kapitel 9.
- ✓ ...om "Rich Data Controls" i kapitel 10.

