Telerik Academy

```
<ul>
    <asp:Repeater runat="server" ID="customers"
        ModelType="WebApplication6.Customer">
        <ItemTemplate>
            <li>
                First Name: <%# Item.FirstName %><br />
                Last Name: <%# Item.
            </li>
        </ItemTemplate>
    </asp:Repeater>
</ul>
```

Equals
FirstName
GetHashCode
GetType
LastName — string Customer.LastName
ToString

ASP.NET Data Binding

Binding UI Controls to Data Classes in Web Forms

ASP.NET Web Forms

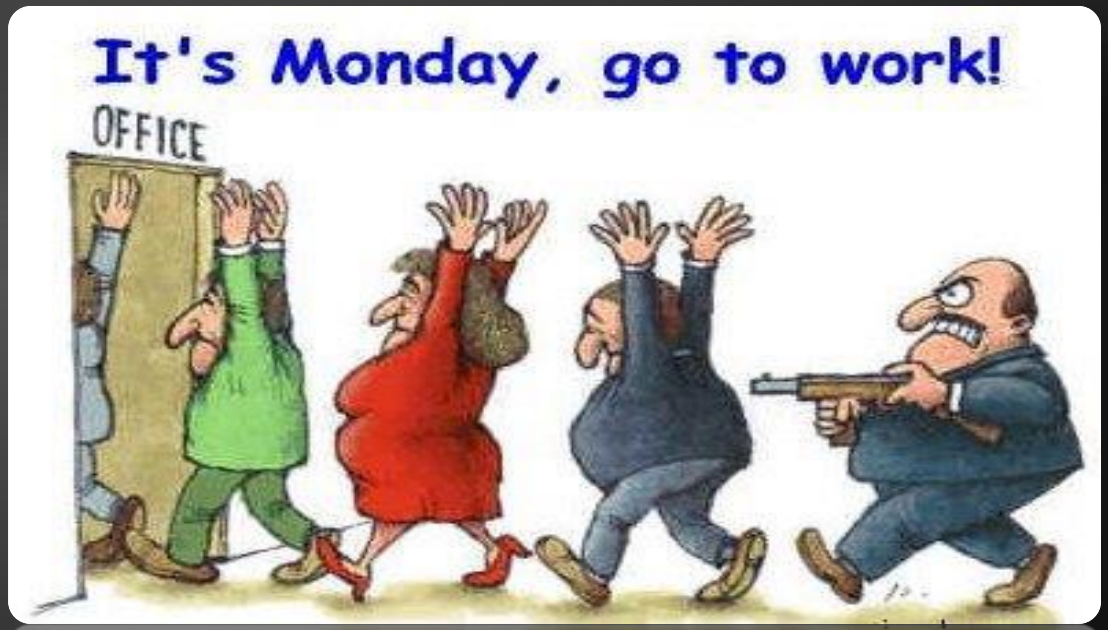
Telerik Software Academy

http://academy.telerik.com

Telerik Academy

Table of Contents

Telerik Academy



How Data Binding Works in ASP.NET?

What is Data Binding?

- Data binding is the process of filling data from a data source into a control
  - ASP.NET Web Forms controls supporting data binding have
    - A property `DataSource`
    - A method `DataBind()`
  - To bind a control we have to set the property `DataSource` and to call the method `DataBind()` after that
  - Binding is usually invoked in `Page_Load()`

Telerik Academy

Data Binding – Simple Example

- Example of static list control with items:

```
<asp:DropDownList ID="DropDownYesNo" runat="server">
  <asp:ListItem>Yes</asp:ListItem>
  <asp:ListItem>No</asp:ListItem>
</asp:DropDownList>
```

- Example of data-bound list control:

```
<asp:ListBox ID="ListBoxTowns" runat="server">
</asp:ListBox>
…
protected void Page_Load(object sender, EventArgs e)
{
  string[] towns = { "Sofia", "Plovdiv", "Varna" };
  this.ListBoxTowns.DataSource = towns;
  this.ListBoxTowns.DataBind();
}
```

Telerik Academy

Data Binding: Simple Example

Live Demo

List-Bound Controls

- Controls that are bound to a data source are called list-bound controls
- `ListBox`, `DropDownList`, `CheckBoxList`, `RadioButtonList`
- `DataList` – shows data in a template-based predefined pattern
- `GridView` – shows data in a table
- `Repeater` – shows data in a template designed by the programmer

Telerik Academy

When Does Binding Take Place?

- Data binding in ASP.NET can occur:
  - In `Page_Load()` or `Page_PreRender()`
    - Sometimes `Page.IsPostBack` is checked
    - `Page_Load()` – called before control events
    - `Page_PreRender()` – called after control events
  - In an event handler:
    - E.g. when a button is pressed, in the `ButtonLoad_Click()` event handler
- Data binding transfers the data from the data source to the control's internal structures

Sources of Data

- Any class deriving from `IEnumerable` can be used as data source in ASP.NET Web Forms
  - Arrays, e.g. `Town[]`
  - Lists, e.g. `List<Town>`
  - LINQ-to-Entities query over Entity data model
  - LINQ-to-XML, LINQ-to-Objects queries
  - Etc.
- ASP.NET data-source classes
  - `EntityDataSource`, `ObjectDataSource`, …
- `DataTable` and `DataSet` classes

Telerik Academy

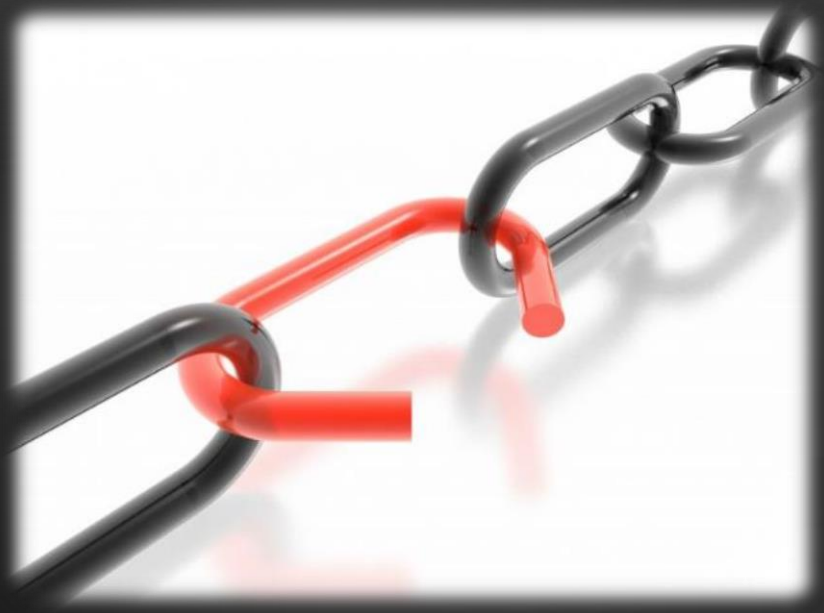
List-Bound Controls: Common Properties

- To connect a list-bound controls to a data source use the properties:
  - `DataSource`
    - Assigns the data source
  - `DataMember`
    - Optionally indicates the object inside the data source: a property or a property path
    - E.g. `DataSource = students`, `DataMember = Address.Town`

Common Properties (2)

- List controls (`ListBox`, `DropDownList`, `CheckBoxList` and `RadioButtonList`) have additional common properties
  - `DataTextField` – the column (property) which will be displayed on the page
    - E.g. `CustomerName`
  - `DataValueField` – the column that will provide the value for the control
    - E.g. `CustomerID`

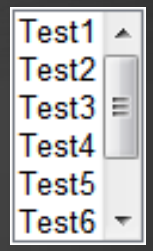
Telerik Academy

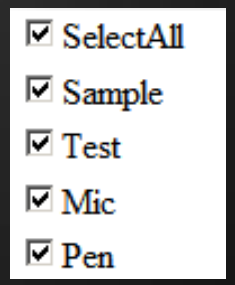


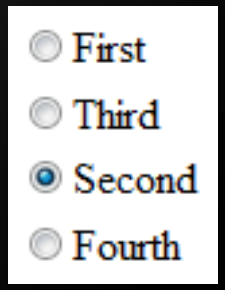
Binding List Controls

List Controls

- Abstract class `ListControl` is base class for all list controls
  - `ListBox`
  - `DropDownList`
  - `BulletedList`
  - `CheckBoxList`
  - `RadioButtonList`
  - …







Binding List Controls

- Common binding-related properties
  - `DataSourceID` – for declarative data binding
  - `DataTextField` – field to display
  - `DataTextFormatString` – field display format
  - `DataValueField` – field to take as result
  - `AutoPostBack` – forces postback on user click
  - `Items` – contains the list items
- Common events
  - `SelectedIndexChanged`

Telerik Academy

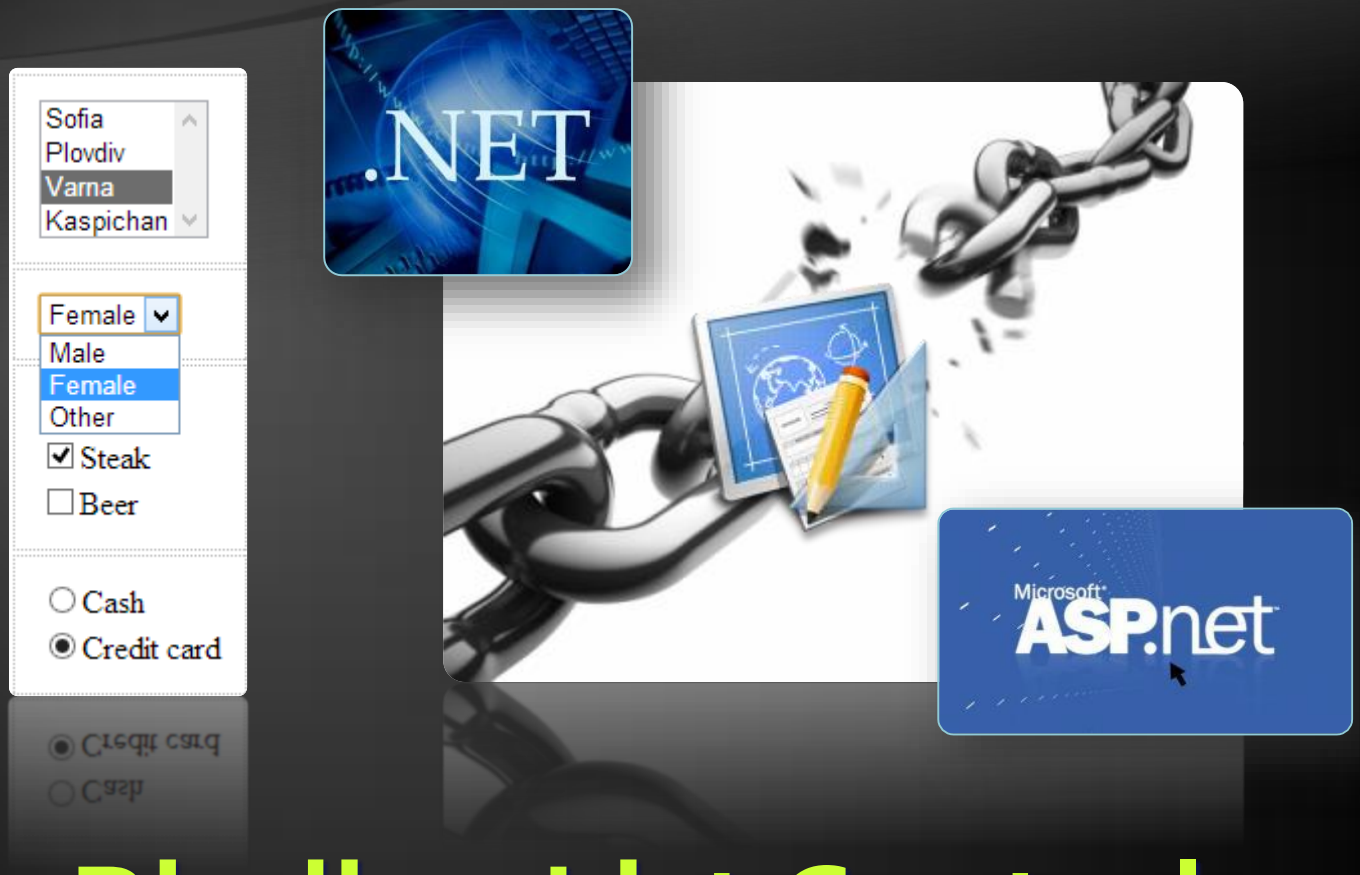
Binding List Controls – Example

```
<asp:BulletedList
  ID="BulletedListMenu"
  runat="server"
  DisplayMode="HyperLink"
  DataTextField="Text"
  DataValueField="Url">
</asp:BulletedList>
```

- Google
- Bing!
- MSDN

```
var urls = new[]
{
  new { Text="Google", Url="http://www.google.com" },
  new { Text="Bing!", Url="http://www.bing.com" },
  new { Text="MSDN", Url="http://msdn.microsoft.com" }
};
this.BulletedListMenu.DataSource = urls;
this.BulletedListMenu.DataBind();
```

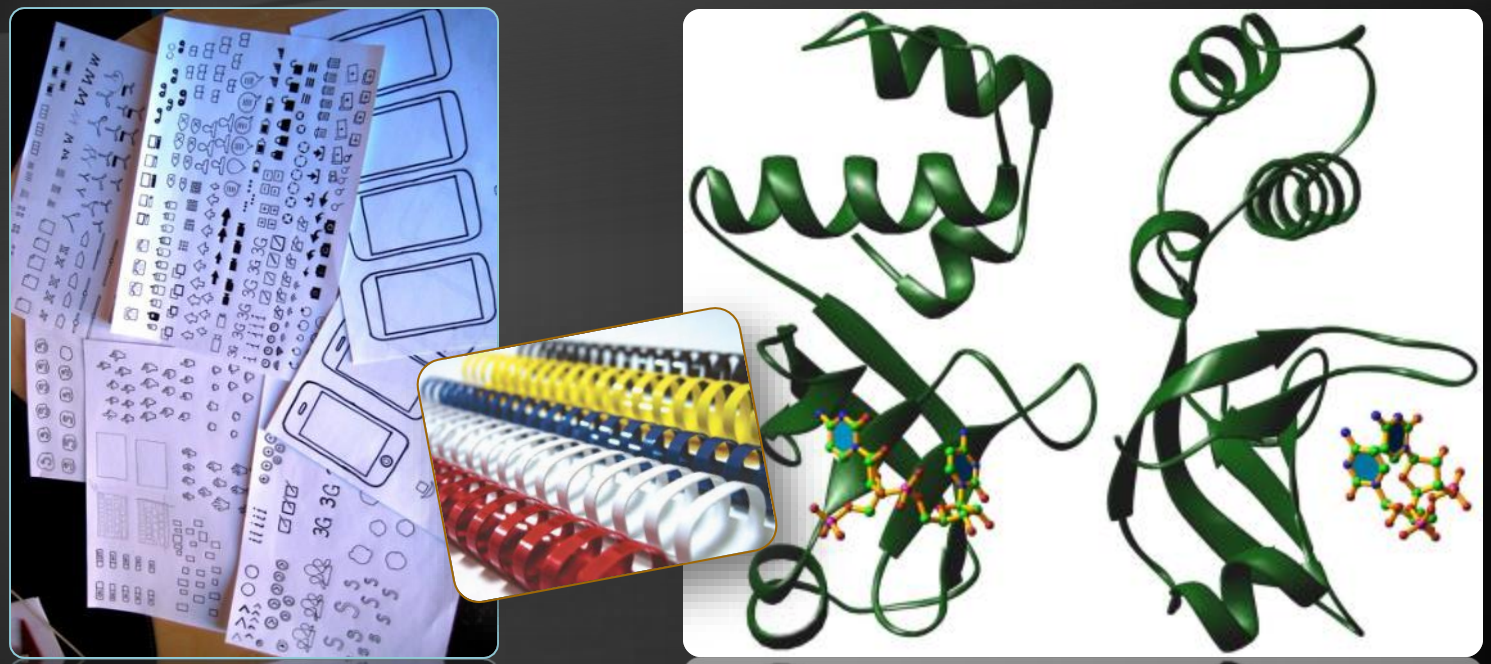
Telerik Academy



Binding List Controls

Live Demo

Telerik Academy



Declarative Data Binding in the ASP.NET Controls

Telerik Academy

Declarative Data Binding Syntax

- ASP.NET Web Forms offers a declarative syntax for data-binding

```
<%#: expression %>
```
– binding with escaping

```
<%# expression %>
```
– unescaped binding

- Evaluated when the `DataBinding` event of the corresponding control is fired for each item (i.e. record / row) in the data source
- The `DataBinder` class is used internally to retrieve the value in a column

Telerik Academy

Data-Binding Syntax – Example

```
// Binding to a property
Customer: <%# this.CustID %>

// Binding to a collection (e.g. array)
Orders: <asp:ListBox ID="ListBoxCountries"
  DataSource="<%# this.Arr %>" runat="server">

// Binding to an expression
Contact: <%#: (customer.FirstName + " " +
  customer.LastName) %>

// Binding to the output of a method
Outstanding Balance: <%# GetBalance(custID) %>

// Strongly-typed binding to a collection
City: <%#: Item.Address.City.Name %>
```

Telerik Academy

How Declarative Binding Works?

- Although declarative binding is similar to <% Response.Write()%> its behavior is different
- Response.Write(…) is evaluated (calculated) when the page is compiled
  - The declarative binding syntax is evaluated when the DataBind(…) method is called
  - If DataBind(…) is never called, the expression <%# … %> is not displayed
  - During the evaluation in binding, the current data item is accessible (Container.DataItem)

Telerik Academy

The DataBind(…) Method

- Page and all server controls have DataBind(…) method
- DataBind(…) is called in a cascading order for all controls in the parent control
  - Evaluates all the <%# … %> expressions
  - DataBind(…) is usually called in the Page_Load or Page_Prerender events

```
void Page_Load(Object sender, EventArgs e)
{
  Page.DataBind(); // Binds all control in the page
}
```

Telerik Academy

Declarative Binding – Example

```
<form runat="server">
  <asp:DropDownList id="lstOccupation"
    AutoPostBack="true" runat="server">
    <asp:ListItem>Manager</asp:ListItem>
    <asp:ListItem>Developer</asp:ListItem>
    <asp:ListItem>Tester</asp:ListItem>
  </asp:DropDownList>
  <p>
    You selected:
    <asp:Label id="lblSelectedValue"
      Text="<%#: lstOccupation.SelectedItem.Text %>"
      runat="server" />
  </p>
</form>
```

Strongly-Typed Binding

- List controls in ASP.NET have `ItemType` property to specify its bound item type
  - Strongly-typed ASP.NET controls have the property `Item` at runtime of type `ItemType`

```
<asp:Repeater ID="RepeaterPeople" runat="server"
  DataSource="<%# GetPeople() %>" ItemType="Person">
  <ItemTemplate>
    <%#: Item.FirstName %> <%#: Item.LastName %><br />
  </ItemTemplate>
</asp:Repeater>
```

Telerik Academy

Declarative Binding

Live Demo

Telerik Academy

Complex Data-Bound Controls

Telerik Academy

Complex DataBound Controls

- `GridView`
  - Displays a list of records as a table
  - Supports templates for header, body, items, …
- `DetailsView`
  - Visualizes the details of a record (fields)
  - Supports paging, header / footer, commands
  - Doesn't support templates
- `FormView`
  - Like `DetailsView` but supports templates

GridView

- `GridView` displays tabular data as HTML table
  - Consists of columns, header and footer
  - Columns can be auto-generated according to the data source or can be set explicitly
  - Supports paging, sorting, editing and deleting
  - Easy to adjust the appearance

```
<asp:GridView ID="GridViewCustomers" runat="server"
  AutoGenerateColumns="true" AllowPaging="true" />
```

GridView Columns

- Set `AutoGenerateColumns` to `false` to customize the columns in the `GridView`

Name	Description
`BoundField`	Renders a text column – data comes from the data source
`ButtonField`	Renders a column with buttons (`Button`, `ImageButton` or `Link`)
`CheckBoxField`	Renders a column with `CheckBox` (boolean data)
`CommandField`	Renders a column for the commands (edit, delete)
`HyperLinkField`	Renders a column with links in it
`ImageField`	Renders a column with an image. The URL comes from the data source
`TemplateField`	Renders a column based on an HTML template

GridView – Example

```
<asp:GridView ID="GridViewCustomers" runat="server"
    AutoGenerateColumns="false">
  <Columns>
    <asp:BoundField DataField="FirstName"
      HeaderText="First Name" />
    <asp:BoundField DataField="LastName"
      HeaderText="Last Name" />
    <asp:BoundField DataField="Phone"
      HeaderText="Phone" />
    <asp:BoundField DataField="EMail"
      HeaderText="E-Mail" />
    <asp:CheckBoxField DataField="IsSenior"
      HeaderText="Senior?" />
  </Columns>
</asp:GridView>
```

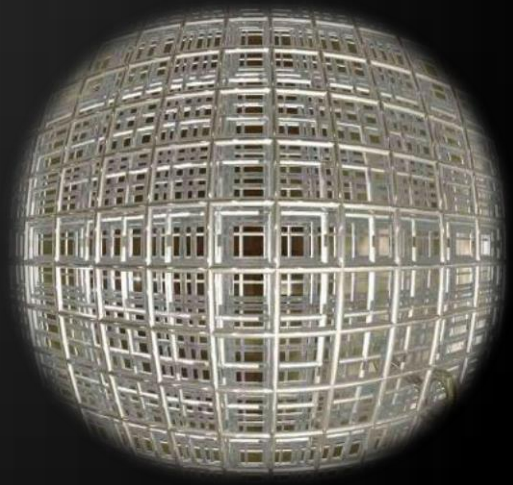
GridView – Example (2)

```
protected void Page_Load(object sender, EventArgs e)
{
  List<Customer> customers = new List<Customer>()
  {
    new Customer() { FirstName = "Svetlin",
      LastName = "Nakov", Email = "svetlin@nakov.com",
      Phone = "0894 77 22 53", IsSenior=true },
    new Customer() { FirstName = "Bai",
      LastName = "Ivan", Email = "bai.ivan@gmail.com",
      Phone = "0899 555 444" },
  };
  this.GridViewCustomers.DataSource = customers;
  this.GridViewCustomers.DataBind();
}
```

Telerik Academy

GridView

Live Demo



DetailsView

- Displays a single record
  - Usually used along with `GridView`
- Supports paging, inserting, updating, deleting
- Uses the same fields as `GridView`
  - Declared in a `<Fields>` element
- Easy to change the appearance
- Can auto-generate fields:
  - `AutoGenerateRows="true"`

FirstName	Kaka
LastName	Mara
Email	kaka.mara@kaval.net
Phone	095 955 955
IsSenior	
1 2 3	

DetailsView – Example

```
<asp:DetailsView ID="DetailsViewCustomer"
  AutoGenerateRows="true" AllowPaging="True"
  runat="server" onpageindexchanging =
    "DetailsViewCustomer_PageIndexChanging">
</asp:DetailsView>

protected void Page_Load(object sender, EventArgs e)
{
  this.DetailsViewCustomer.DataSource = customers;
  this.DetailsViewCustomer.DataBind();
}

protected void DetailsViewCustomer_PageIndexChanging(
  object sender, DetailsViewPageEventArgs e)
{
  this.DetailsViewCustomer.PageIndex = e.NewPageIndex;
  this.DetailsViewCustomer.DataSource = customers;
  this.DetailsViewCustomer.DataBind();
}
```

Telerik Academy

DetailsView

Live Demo

FormView

- Templated version of `DetailsView`
  - Doesn't use predefined a view
    - Requires the developer to define the view by using templates
  - Doesn't have commands
    - It has mode (view, edit, insert, …)
  - You can use many controls for the templates – `DropDownList`, `CheckBox`, `Calendar`, etc.

FormView (2)

- You are responsible to define some or all of the templates
  - `ItemTemplate`
  - `EditItemTemplate`
  - `InsertItemTemplate`

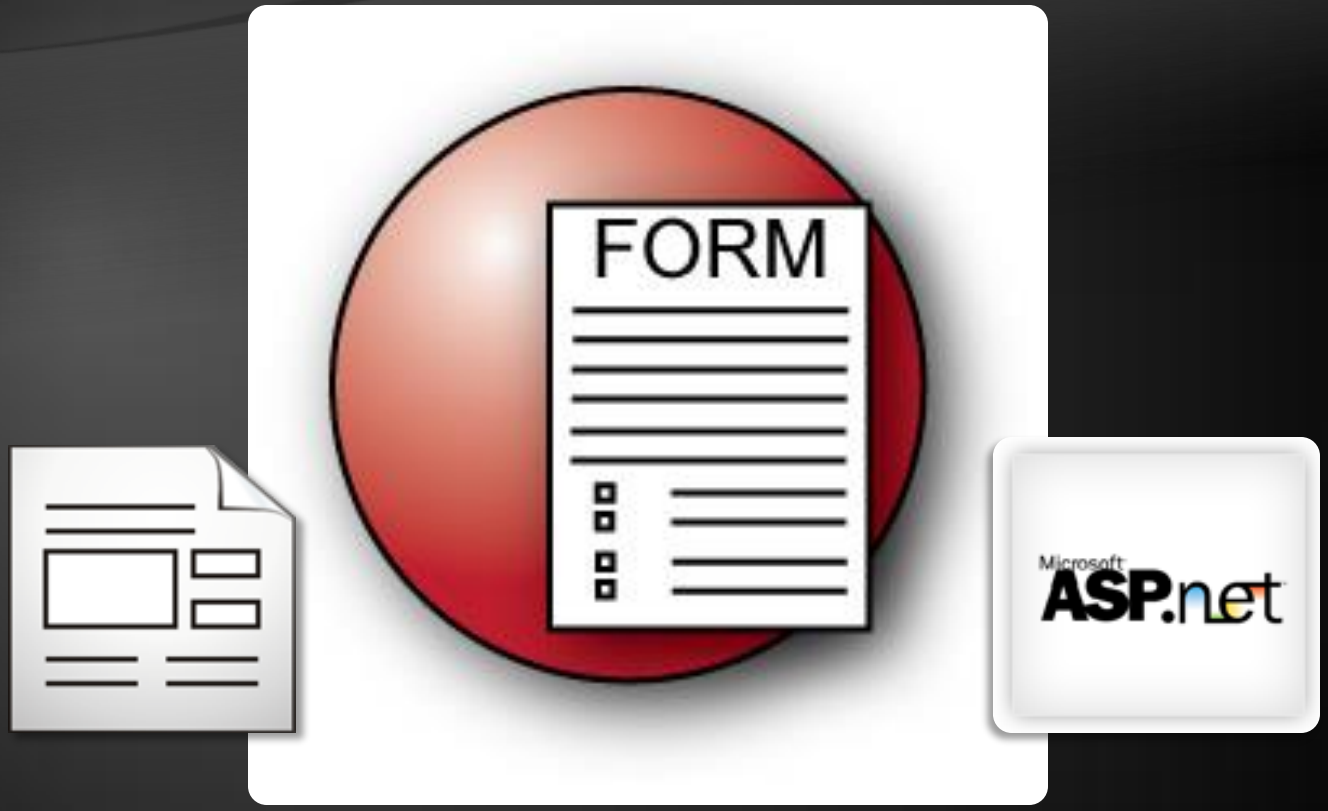
- You could change the view mode at run-time:

```
this.FormViewCustomer.ChangeMode(FormViewMode.Edit);
```

FormView – Example

```
<asp:FormView ID="FormViewCustomer" runat="server"
  AllowPaging="True" onpageindexchanging=
  "FormViewCustomer_PageIndexChanging">
  <ItemTemplate>
    <%#: Eval("FirstName") %>
    <%#: Eval("LastName") %>
  </ItemTemplate>
</asp:FormView>
…
protected void Page_Load(object sender, EventArgs e)
{
  this.FormViewCustomer.DataSource = this.customers;
  this.FormViewCustomer.DataBind();
}
```

Telerik Academy



FormView

Live Demo

The TreeView Control

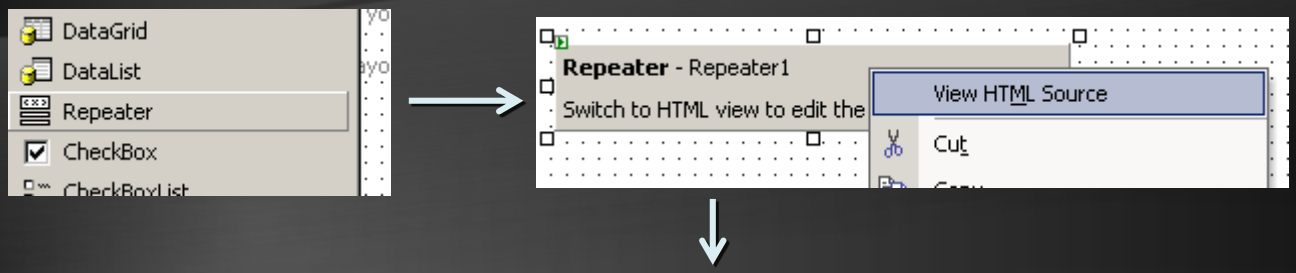
- `TreeView` is a fully functional control used to display hierarchical data
- Allows multiple visual adjustments
  - Node images, fold and expand images, connecting lines, checkboxes
- Supports navigation and postback
- You can create nodes declaratively or in code
  - We can fill nodes dynamically from the server when needed (when the data is too much)

Repeater

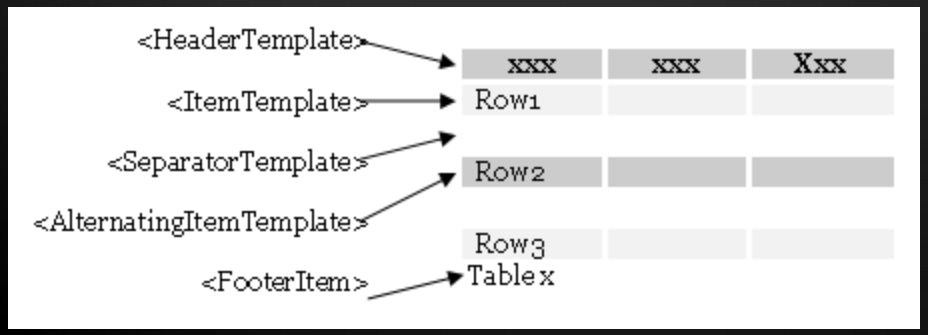
- `GridView` doesn't give you full control
  - Uses HTML tables (`<table>`)
- The `Repeater` control is the most flexible control to show a sequence of data rows
  - Template-based visualization
  - You write the HTML visualization code yourself
- Useful when you want to implement a non-standard visualization of read-only data
  - The output code is manually written

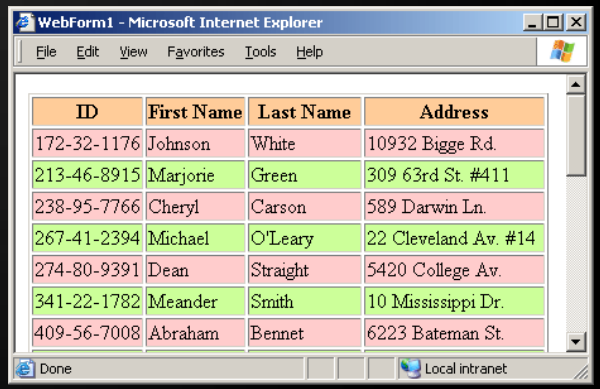
Telerik Academy

Repeater: How to Use It?




```
<body MS_POSITIONING="GridLayout">
    <form id="Form1" method="post" runat="server">
        <asp:Repeater id="Repeater1" runat="server">
        <
            AlternatingItemTemplate
            FooterTemplate
            HeaderTemplate
            ItemTemplate
            SeparatorTemplate

        </asp:Repeater>
    </form>
</body>
```





Telerik Academy

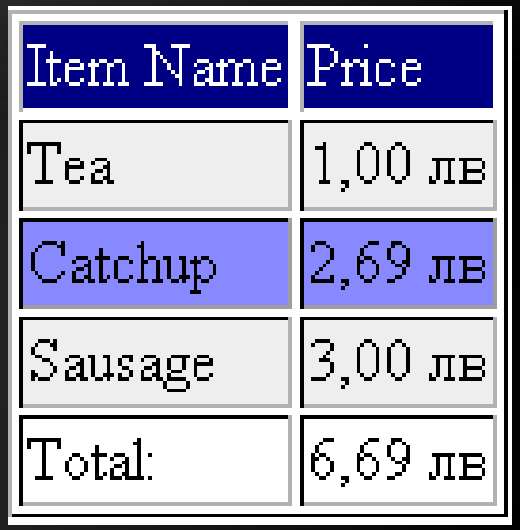
Templates, Eval(…) and Strongly-Typed Binding

Templates

- The `GridView`, `Repeater` and `FormView` offer rich customization based on templates
- Data templates
  - Display data in highly-customizable fashion
    - Format the appearance of data
  - `Eval(expression)` or `Item` (in strongly-typed binding) provide access to the current item
  - The current data-bound item is accessible through the `Container.DataItem`

Templates (2)

- <HeaderTemplate>
- <ItemTemplate>
- <AlternatingItemTemplate>
- <FooterTemplate>
- Example:




```
<AlternatingItemTemplate>
    <tr style="background: #8888FF">
        <td><%#: Eval("ItemName") %></td>
        <td><%#: Eval("Price", "{0:c}") %></td>
    </tr>
</AlternatingItemTemplate>
```

Accessing the Current Item

- Accessing at runtime the current item from the data source collection (`IEnumerable`):
  - `Container.DataItem`
    - Gets the current item as `object`
  - `DataBinder.Eval(container, expression, [format])`
    - Evaluates an expression through reflection
    - Slower than `Container.DataItem`
  - Just use `Eval(expression)` as shortcut

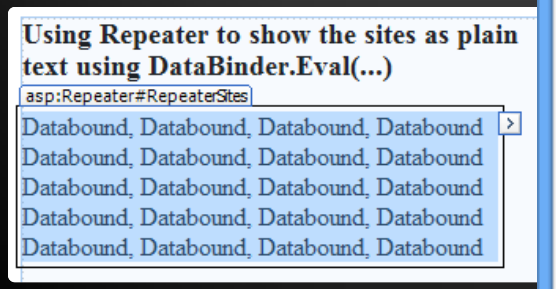
Telerik Academy

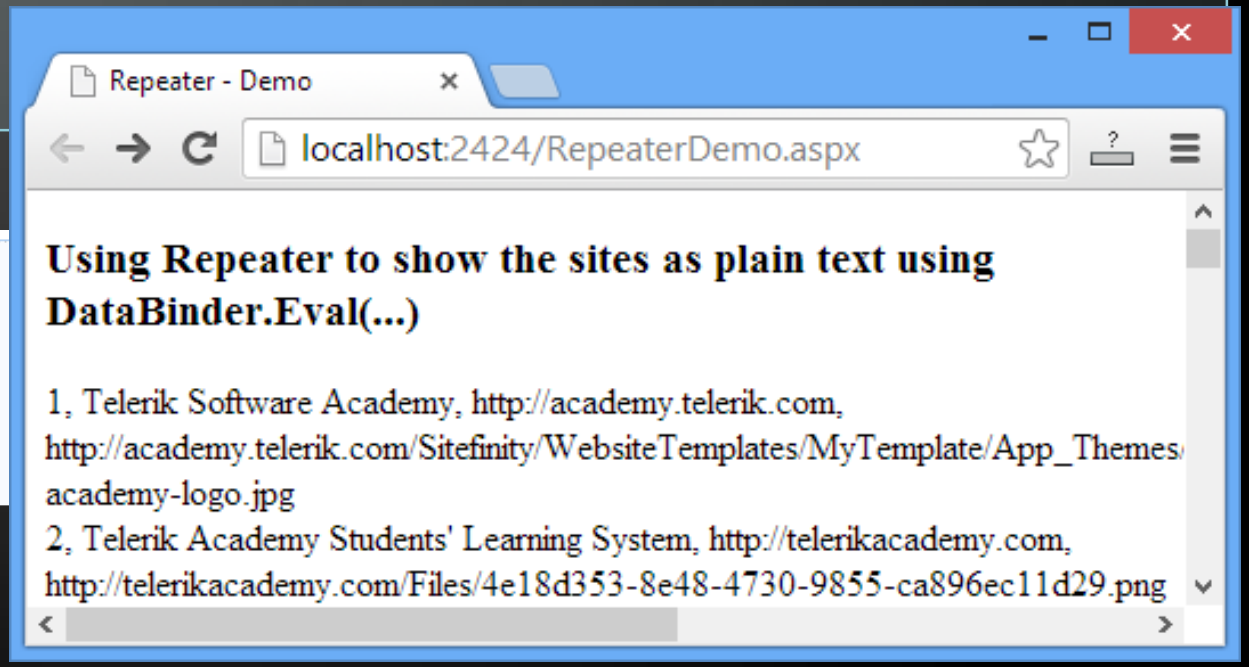
DataBinder.Eval() vs. Container. DataItem vs. Eval() vs. Item.Prop

```
// Using Container.DataItem
<%#: String.Format("{0:c}", ((DataRowView)
  Container.DataItem)["SomeIntegerField"])%>

// Using DataBinder.Eval(…)
<%#: DataBinder.Eval(Container.DataItem,
  "SomeIntegerField", "{0:c}") %>

// Using Eval(…)
<%#: Eval("SomeIntegerField", "{0:c}") %>

// Using Item.PropertyPath (strongly-typed)
<%#: Item.Address.Town.ID %>
```

Repeater – Example

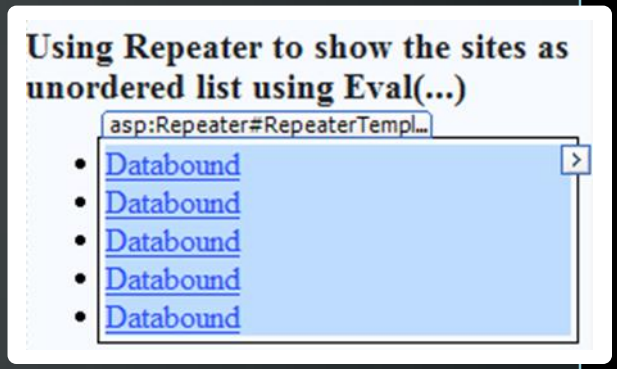
```
<asp:Repeater id="RepeaterSites" runat="server">
 <ItemTemplate>
  <%#: DataBinder.Eval(Container.DataItem, "Id") %>,
  <%#: DataBinder.Eval(Container.DataItem, "Name") %>,
  <%#: DataBinder.Eval(Container.DataItem, "URL") %>,
  <%#: DataBinder.Eval(Container.DataItem, "ImageURL") %>
</ItemTemplate>
</asp:Repeater>
```





Repeater – Example (2)

```
<asp:Repeater id="RepeaterTemplatedList" runat="server">
  <HeaderTemplate>
    <ul>
  </HeaderTemplate>
  <ItemTemplate>
    <li>
      <a href="<%#: Eval("URL") %>">
        <%#: Eval("Name") %>
      </a>
    </li>
  </ItemTemplate>
  <FooterTemplate>
    </ul>
  </FooterTemplate>
</asp:Repeater>
```



Using Repeater to show the sites as unordered list using Eval(...)

- Telerik Software Academy
- Telerik Academy Students' Learning System
- Telerik Academy Forums
- Telerik Academy Facebook Page

Repeater – Example (3)

```
<asp:Repeater ID="RepeaterImages" runat="server"
  ItemType="Site">
  <ItemTemplate>
    <p><a href='<%#: Item.URL %>'>
    <img src='<%#: Item.ImageURL %>' border="0"
      alt='<%#: Item.Name %>' /></a></p>
  </ItemTemplate>
</asp:Repeater>
```

Using Repeater to show the sites as images using strongly-typed binding (Item.PropertyPath)

Telerik Academy

Using Repeater with Templates

Live Demo

ListView

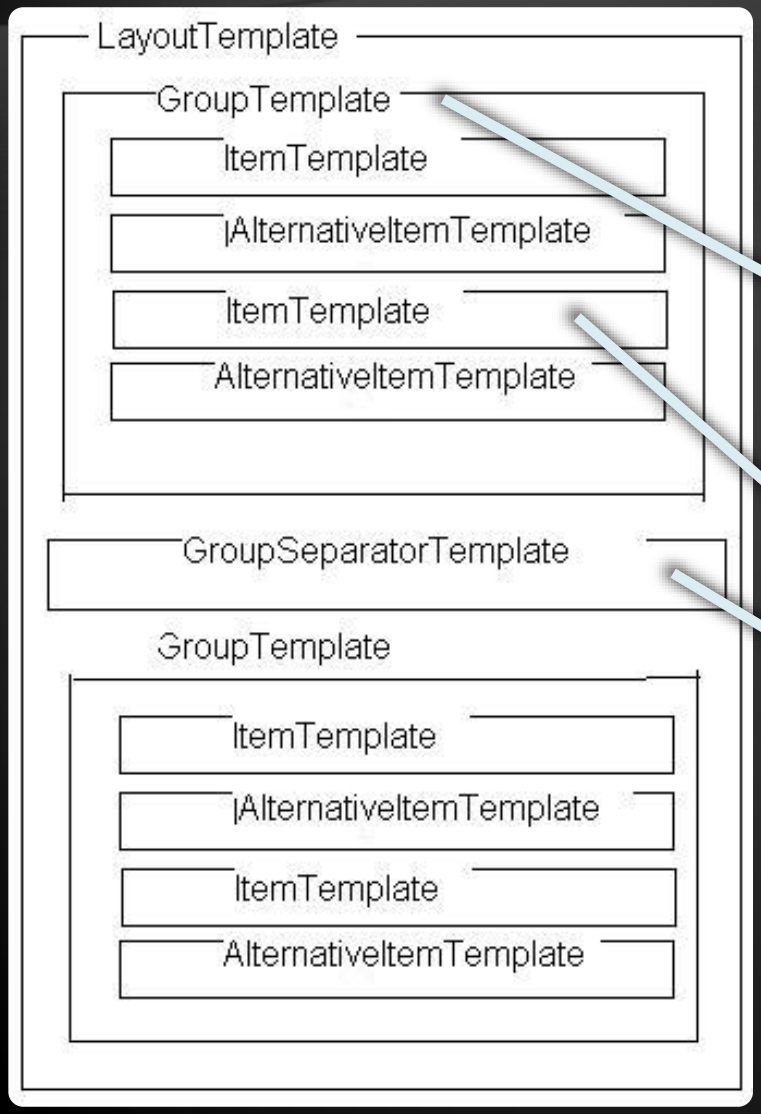
- `ListView` is an extremely flexible data-bound control for displaying lists and tables
  - Works similarly to `GridView`
  - Adds higher-level features
    - Selection, inserting and editing
  - More extensive set of templates than `GridView`
  - Supports paging (through `DataPager`)
  - Groups can display items in tiles (using `GroupTemplate` and `GroupItemCount`)

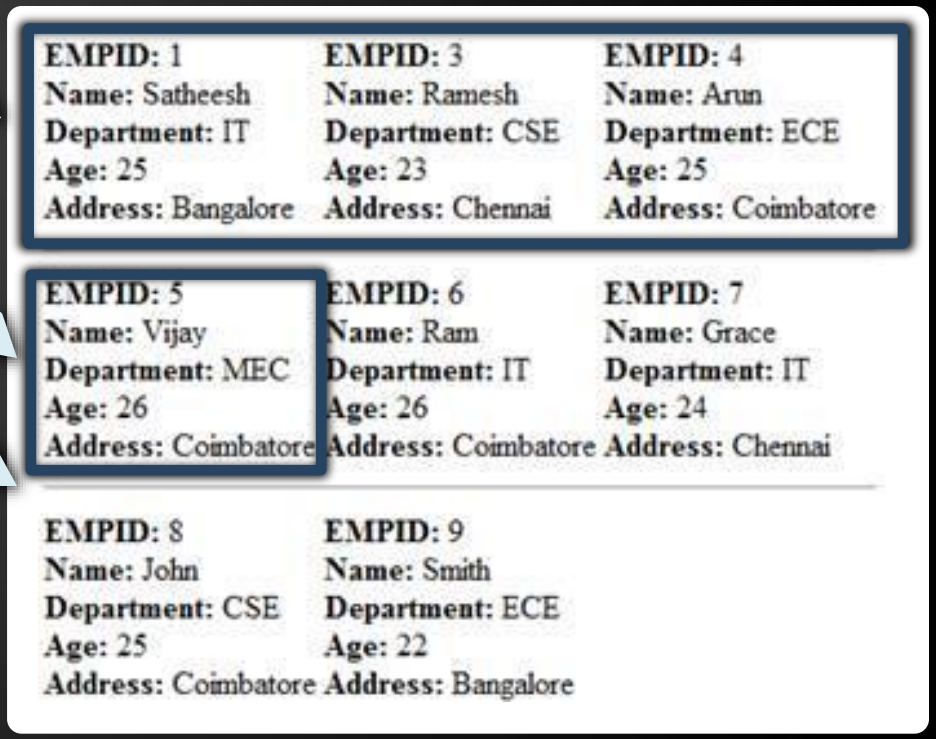
ListView – Templates

- ItemTemplate – sets the content of every data item
- ItemSeparatorTemplate
- SelectedItemTemplate
- EditItemTemplate, InsertItemTemplate
- GroupTemplate
- EmptyItemTemplate
- and etc.

Telerik Academy

ListView – Templates and Groups





DataPager

- DataPager gives you a single, consistent way to use paging with a variety of controls
- The ListView is the only control that supports the DataPager
- Pager Fields
  - NextPreviousPagerField
  - NumericPagerField
  - TemplatePagerField

Telerik Academy

ListView and DataPager

Live Demo

Telerik Academy

Using multiple controls

```
<asp:DropDownList runat="server“ ID="DropDownListCategory"
        OnSelectedIndexChanged=“ddlCategory_Changed“
        AutoPostBack="true">
    <asp:ListItem>all</asp:ListItem>
</asp:DropDownList>

<asp:DropDownList runat="server“ ID="DropDownListBrand"
        OnSelectedIndexChanged=“ddlCategory_Changed">
    <asp:ListItem>all</asp:ListItem>
</asp:DropDownList>
```

```
protected void ddlCategory_Changed(object sender, EventArgs e)
{
    DropDownList listView = sender as DropDownList;
    string selectedCategory = listView.SelectedValue;
    this.ListViewCars.DataSource =
        GetBrandsByCategory(selectedCategory);
    this.ListViewCars.DataBind();
}
```

Telerik Academy

ASP.NET Data Binding

Homework

1. Create a Web form for searching for cars by producer + model + type of engine + set of extras (see www.mobile.bg). Use two `DropDownLists` for the producer (e.g. VW, BMW, …) and for the model (A6, Corsa,…). Create a class `Producer` hodling `Name` + collection of models. Bind the list of producers to the first `DropDownList`. The second should be bound to the models of this producer. You should have a check box for each "extra" (use class `Extra` and bind the list of extras at the server side). Implement the type of engine as a horizontal radio button selection (options bound to a fixed array). On submit display all collected information in `<asp:Literal>`.

Homework (2)

2. Create a page `Employees.aspx` to display the names of all employees from `Northwind` database in a `GridView` as hyperlinks. All links should redirect to another page (e.g. `EmpDetails.aspx?id=3`) where details about the employee are displayed in a `DetailsView`. Add a back button to return back to the previous page.
4. Implement the previous task in a single ASPX page by using a `FormView` instead of `DetailsView`.
5. Display the information about all employees a table in an ASPX page using a `Repeater`.
6. Re-implement the previous using `ListView`.

Homework (3)

7. * Create a Web Form that reads arbitrary XML document and displays it as tree. Use the `TreeView` Web control on the left side to display the inner XML of the selected node on the right side.
8. * Create a Web Form that shows in a table the names, country and city of all employees from the `Northwind` database. Implement paging (10 employees on each page) and sorting by each column. Using AJAX, JavaScript and jQuery on mouse over display a popup DIV with additional info about the employee: photo, phone, email, address, notes. On mouse out hide the additional info.

Free Trainings @ Telerik Academy

- "Web Design with HTML 5, CSS 3 and JavaScript" course @ Telerik Academy
  - html5course.telerik.com

- Telerik Software Academy
  - academy.telerik.com

- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy

- Telerik Software Academy Forums
  - forums.academy.telerik.com