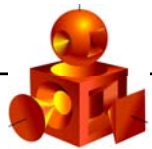
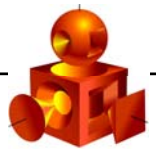
ASP.NET

By Arjan Einbu (http://blog.einbu.no)

(Revision: AE/01/06/09)

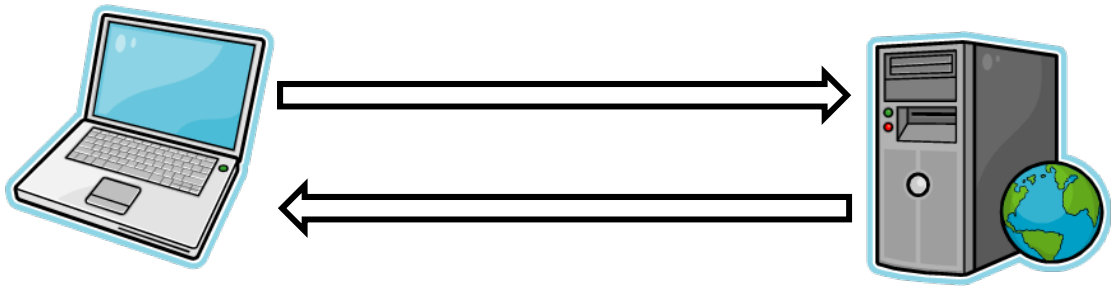


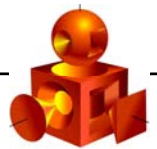
Chapter 1
Introduction to ASP.NET



HTML, HTTP and the World Wide Web

- **HTML was originally created as a way to create a link from one document to another**
- **HTML documents are read using a webbrowser**
- **The Webbrowser gets the files from the webserver over the HTTP protocol**
- **After fetching the document, the webserver and the webbrowser disconnect from each other**

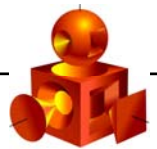




Dynamic Web Pages

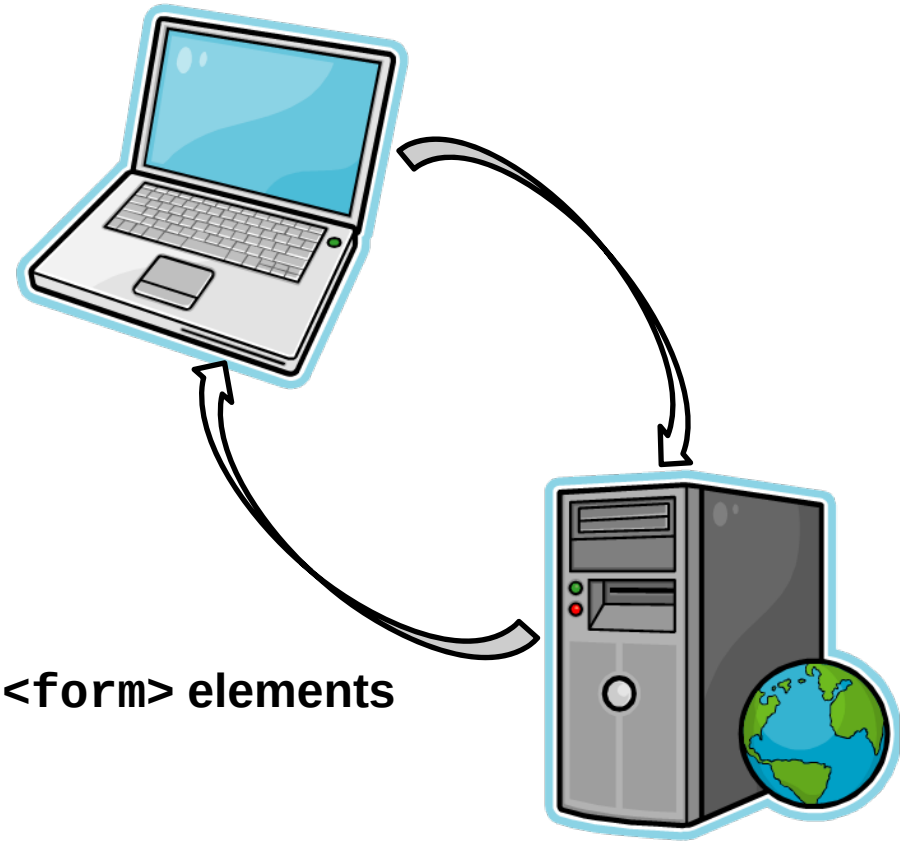
- **Dynamic pages are created at the server at the time of a request, by modifying or creating from scratch the documents as they're sent from the webserver**

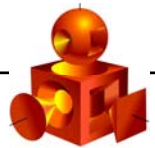
- **IIS and ASP.NET create the dynamic content**



PostBack

- **Server roundtrip**
- **Caused by an event on a control**
  - Clicking a Button
  - Selecting an item
  - Checking a CheckBox
  - Choosing a RadioButton
  - Selecting a date in a Calendar
- **Posts back data within the HTML `<form>` elements**
  - Which button was clicked
  - The text of a TextBox
  - The selection in a ListBox

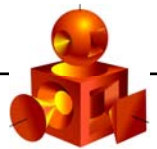




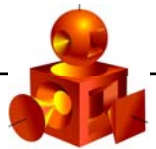
web.config

- **ASP.NET is configured using web.config**
  - We can add our own configuration informaton and elements

- **web.config represents a layered approach to configuration**
  - machine.config
  - web.config at machine level
  - web.config at site level
  - web.config at application level
  - web.config at folder level

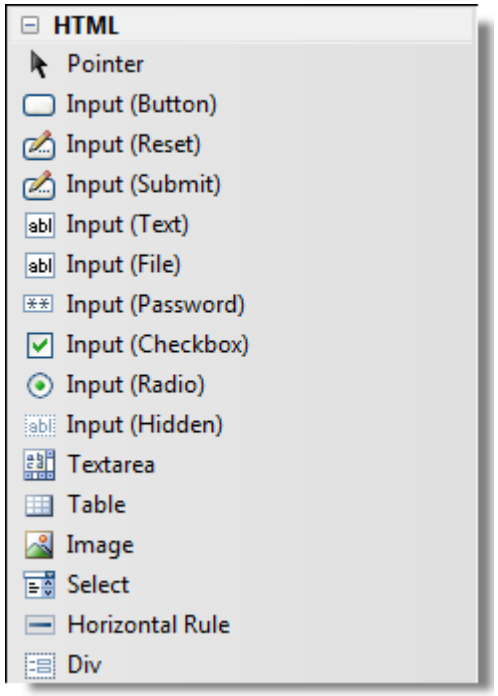


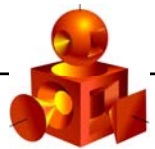
Chapter 2
Controls



HTML Controls

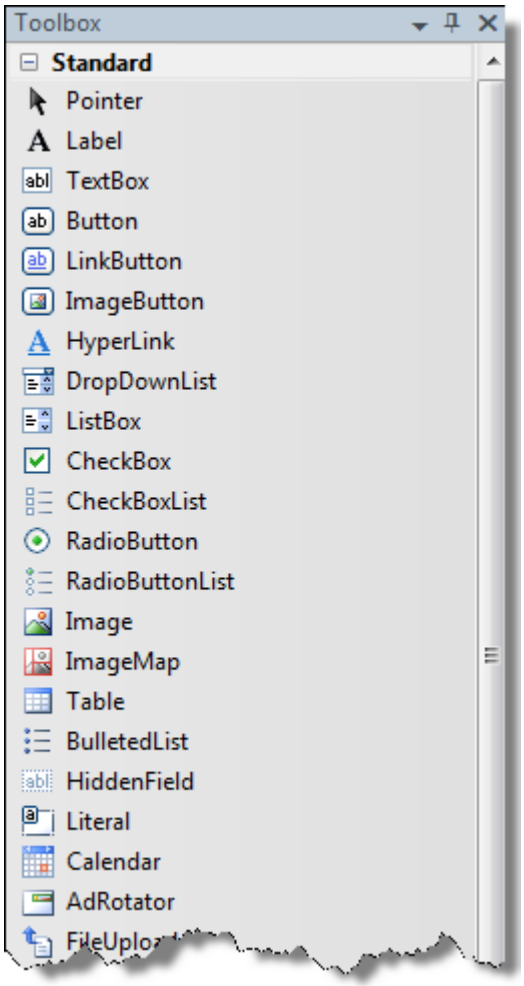
- **Copied Directly to HTML Output**
  - Except `runat="server"` attribute
- **Accesible from the CodeBehind**
  - By adding the `runat="server"` attribute
- **We can use HTML tables for layout**
  - HTML table has better designer support than `<asp:table />`

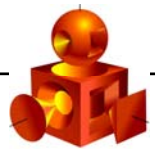




Server Controls

- **Attributes**
  - `ID="Label1"`
  - `runat="server"`
- **Programatic access to properties, methods and events**
- **Wrap functionality**
- **Emits standard HTML when rendered**
- **Can store state information in ViewState or ControlState**





Label

- **ASP.NET markup**

```
<asp:Label ID="Label1" runat="server" Text="Label">
</asp:Label>
```

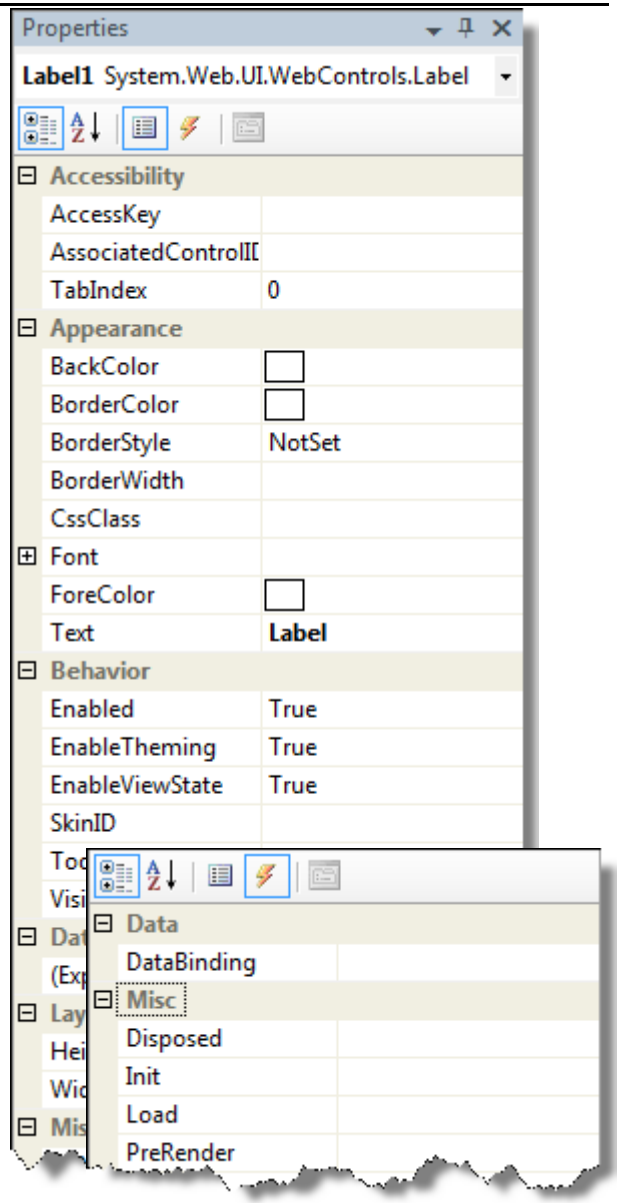
- **Renders as HTML**

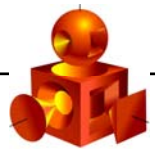
```
<span id="Label1">Label</span>
```

- **Programatic access**

```
Label1.Text = "Hello World!";
```

- **Has properties and events**



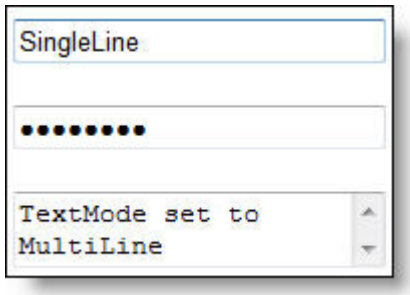


TextBox

- **ASP.NET markup**

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

- **TextMode**
  - SingleLine
  - Password
  - MultiLine




```
<input name="TextBox1" type="text"
id="TextBox1" />
```

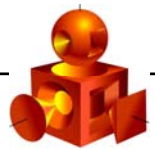
```
<input name="TextBox2" type="password"
id="TextBox2" />
```

```
<textarea name="TextBox3" rows="2"
cols="20" id="TextBox3" />
```

- **AutoPostback property**
  - Causes a PostBack when the Text property has changed

- **TextChanged event**

```
protected void TextBox1_TextChanged(object sender, EventArgs e)
{
    string userInput = TextBox1.Text;
}
```



Button

- **ASP.NET markup**

```
<asp:Button ID="Button1" runat="server" Text="Trykk her!" />
```

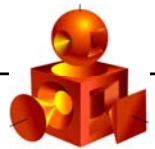
- **Renders as HTML**

```
<input type="submit" name="Trykk her!" value="Button" id="Button1" />
```

- **Causes postback when clicked**
  - Handle the Click event

```
protected void Button1_Click(object sender, EventArgs e)
{
    //Do something usefull...
}
```

- **The LinkButton and ImageButton controls provide the same funtionality, but with a different UI**
  - They implement the IButtonControl interface



DropDownList

- **ASP.NET markup**

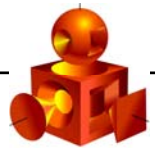
```
<asp:DropDownList ID="DropDownList1" runat="server">
    <asp:ListItem Value="1" Selected="True">Oslo</asp:ListItem>
    <asp:ListItem Value="2">Bergen</asp:ListItem>
    <asp:ListItem Value="3">Trondheim</asp:ListItem>
</asp:DropDownList>
```

- **Renders as HTML**

```
<select name="DropDownList1" id="DropDownList1">
    <option selected="selected" value="1">Oslo</option>
    <option value="2">Bergen</option>
    <option value="3">Trondheim</option>
</select>
```

- **Setting the AutoPostback property will cause a postback when the selected item has changed**

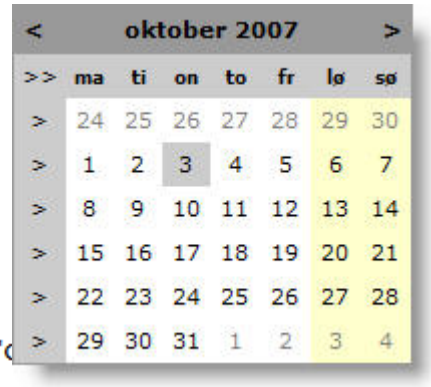
- **The ListBox provides similar functionality, but with a different UI**
  - CheckListBox, RadioButtonList



Calendar

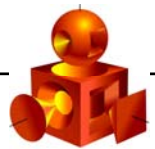
- **ASP.NET Markup**

```
<asp:Calendar ID="Calendar1"
runat="server"></asp:Calendar>
```



- **Renders as HTML with JScript**

```
                <table id="Calendar1" cellspacing="0" cellpadding="2" title="C
style="border-width:1px;border-style:solid;border-collapse:collapse;">
        <tr><td colspan="7" style="background-color:Silver;"><table cellspacing="0" border="0"
style="width:100%;border-collapse:collapse;">
                <tr><td style="width:15%;"><a href="javascript:__doPostBack('Calendar1','V2738')"
style="color:Black" title="Go to the previous month">&lt;</a></td><td align="center"
style="width:70%;">august 2007</td><td align="right" style="width:15%;"><a
href="javascript:__doPostBack('Calendar1','V2800')" style="color:Black" title="Go to the next
month">&gt;</a></td></tr>
        </table></td></tr><tr><th align="center" abbr="mandag" scope="col">ma</th><th align="center"
abbr="tirsdag" scope="col">ti</th><th align="center" abbr="onsdag" scope="col">on</th><th
align="center" abbr="torsdag" scope="col">to</th><th align="center" abbr="fredag"
scope="col">fr</th><th align="center" abbr="lørdag" scope="col">lø</th><th align="center"
abbr="søndag" scope="col">sø</th></tr><tr><td align="center" style="width:14%;"><a
href="javascript:__doPostBack('Calendar1','2767')" style="color:Black" title="30.
juli">30</a></td><td align="center" style="width:14%;"><a href="javascript:__doPostBack
('Calendar1','2768')" style="color:Black" title="31. juli">31</a></td><td align="center"
style="width:14%;"><a href="javascript:__doPostBack('Calendar1','2769')" style="color:Black"
title="1. august">1</a></td><td align="center" style="width:14%;"><a href="javascript:__doPostBack
('Calendar1','2770')" style="color:Black" title="2. august">2</a></td><td align="center"
style="width:14%;"><a href="javascript:__doPostBack('Calendar1','2771')" style="color:Black"
title="3. august">3</a></td><td align="center" style="width:14%;"><a href="javascript:__doPostBack
('Calendar1','2772')" style="color:Black" title="4. august">4</a></td><td align="center"
style="width:14%;"><a href="javascript:__doPostBack('Calendar1','2773')" style="color:Black"
title="5. august">5</a></td></tr><tr><td align="center" style="width:14%;"><a
```



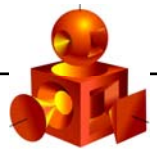
Calendar

- **Higly customizable through the DayRender event**

```
protected void Calendar1_DayRender(object sender, DayRenderEventArgs e)
{
    if (e.Day.Date.DayOfWeek == DayOfWeek.Wednesday)
    {
        e.Cell.Controls.Add(new LiteralControl("<br/>Lille Lørdag"));
    }
}
```

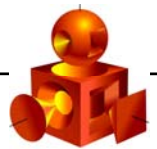
- **Automatically posts back page when clicked**
  - Month changed
  - Selection changed

- **SelectionMode property**
  - Day
  - DayWeek
  - DayWeekMonth
    - ✓ Use the SelectedDate and SelectedDates properties as needed



Design vs. Markup vs. C#

- **Properties Window**
  - (ID)
- **Attributes**
  - `ID="..."`
  - `runat="server"`
- **Classes and objects**
  - `Label lbl = new Label();`
  - `lbl.Text = "... ";`
  - `this.Controls.Add(lbl);`



Events and EventHandlers

- **EventHandlers follow a common pattern**
  - They are based on delegates

```
public delegate void EventHandler(object sender, EventArgs e);
```

  - They are methods

```
protected void Button1_Click(object sender, EventArgs e)
{
    //some code...
}
```

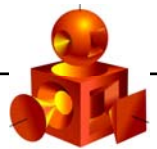
- **EventHandler methods are connected to Events**
  - In CodeBehind

```
Buton1.Click += new EventHandler(methodname);
```

  - Declarativly in markup

```
<asp:Button ID="Button1" runat="server" OnClick="methodname" Text="Button" />
```

- **AutoEventWireup automatically hooks up eventhandlers in codebehind to Page level events based on methodname**
  - Page_Init, Page_Load

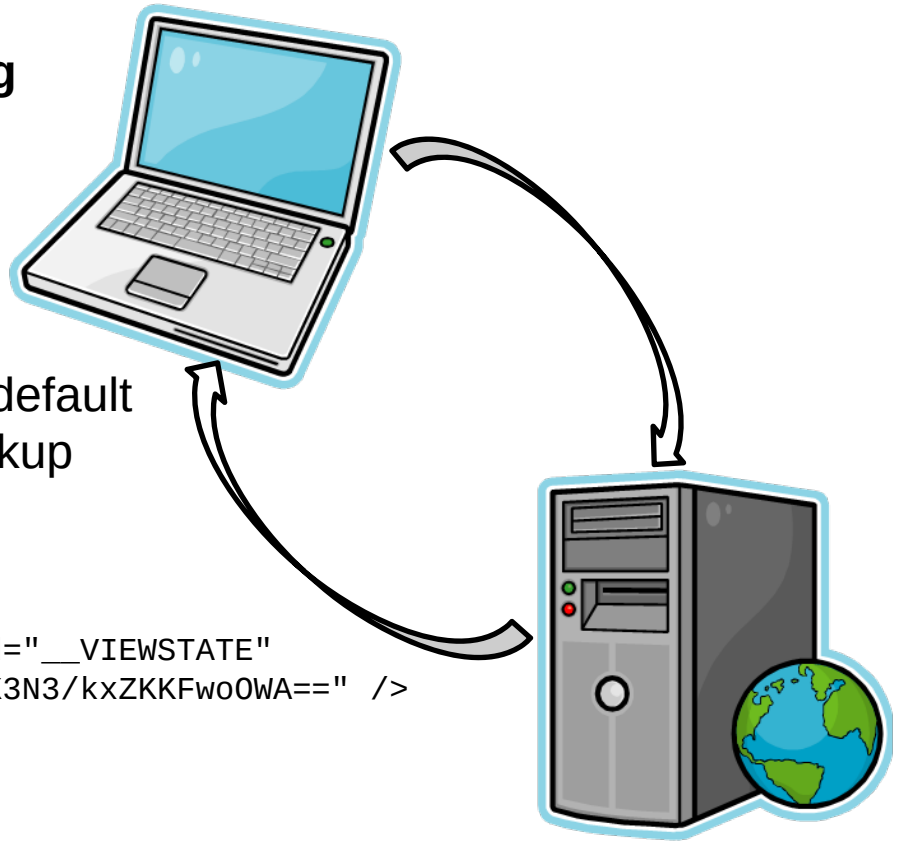


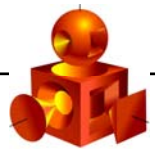
ViewState

- **Stores the state of controls during the postback's server roundtrip**
- **Keeps all information that is set or changed dynamically**
  - When actuall values differ from default values or from values set in markup
- **Renders as HTML**

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUJMjgzMDgzOTgzZGSuBv918qI3lHX3N3/kxZKKFwoOWA==" />
```

- **Can be disabled**
  - For controls
  - For pages



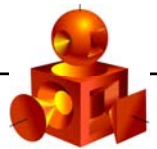


IsPostBack

- **The IsPostBack property is used to determine wether the page is requested as a result of a postback**
  - The click of a button
  - Selection in a list with the AutoPostBack property set
- **Typically from the same page**
- **Enables reuse of values over subsequent server roundtrips**

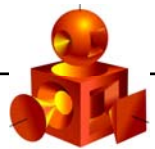
```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack) {
        List<Person> persons = Person.GetPersons();
        foreach (Person p in persons) {
            ListBox1.Items.Add(new ListItem(p.FullName, p.Id.ToString()));
        }
    }
}
```

- **State is kept in ViewState**



Container Controls

- **Can contain other controls**
- **Panel can be used to group controls together**
  - Enable or disable multiple controls
  - Hide or show
- **PlaceHolder**
  - Dynamically add controls at runtime
- **MultiView and Wizard**
  - Multiple pages of controls
  - One page visible at a time
- **DataGridView, DataList, FormsView, DetailsView, Repeater, LoginView...**



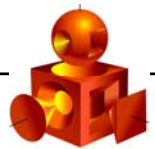
Panel Control

- **The Panel control creates a region to other controls in**
  - Panel element can contain any ASP.NET markup
  - Other controls can be dropped inside the Panel using the designer
  - Controls can be added dynamically

```
Panel1.Controls.Add(...);
```

- **Renders as a `<div>` element**
  - Can be styled with CSS

- **Operations on panels affect all controls inside the panel**
  - Enable/Disable
  - Visibility

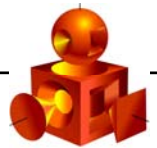


MultiView and View Controls

- **MultiView and View work together**
  - MultiView is the container for one or more View controls
  - Only one of the View controls is shown at one time
- **View controls can contain any ASP.NET markup**
  - Similar to the Panel control
- **No HTML of their own**
  - Only the controls/markup within the active view is rendered to HTML
- **Choose view to display**
  - Set the ActiveViewIndex on the MultiViewControl
  - Use the SetActiveView method

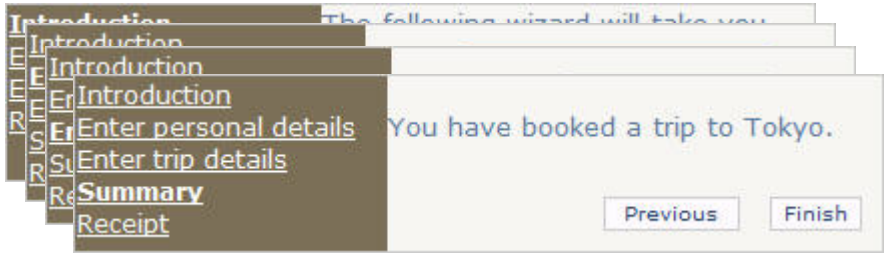
```
MultiView1.SetActiveView(View2);
```

- **Changing views is done serverside, requiring a server roundtrip**



Wizard Control

- **Similar to a MultiView with navigation**

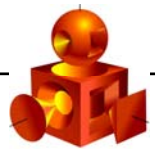


- **Templates**
  - Each page
  - Header
  - Navigation buttons

```
<StepNavigationTemplate>
    <asp:Button ID= "Btn1" runat="server" CommandName="MovePrevious" Text= "Back" />
    <asp:Button ID= "Btn2" runat="server" CommandName="MoveNext" Text="Next" />
</StepNavigationTemplate>
```

  - Sidebar

```
<SideBarTemplate>
    <asp:DataList ID="SideBarList" runat="server">
        <ItemTemplate>
            <asp:Button ID="SideBarButton" runat="server" Text="Button" />
        </ItemTemplate>
    </asp:DataList>
</SideBarTemplate>
```



Wizard Control

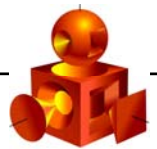
- **Events are fired when user navigates the Wizard**
  - ActiveStepChanged

```
protected void Wizard1_ActiveStepChanged(object sender, EventArgs e)
{
    if (Wizard1.ActiveStep.StepType == WizardStepType.Complete)
    {
        //Create the receipt
    }
}
```

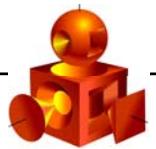
  - NextButtonClick, PreviousButtonClick, FinnishButtonClick, SideBarButtonClick

```
protected void Wizard1_FinishButtonClick(object sender, WizardNavigationEventArgs e)
{
    if (dataEntryIsComplete)
    {
        //Show the complete step
    } else {
        e.Cancel = true;
    }
}
```

  - CancelButtonClick

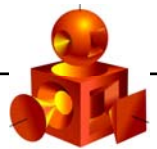


Chapter 3
Data and Databinding



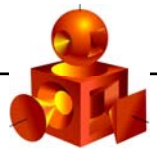
Categories of DataBound Controls

- **Simple (ListControl)**
  - ListBox, DropDownList, CheckedListBox, BulletedList, RadioButtonList
  - AdRotator
- **Composite (CompositeDataboundControl)**
  - GridView and DataList
  - DetailsView and FormView
  - Repeater and ListView
- **Hierarchical (HierarchicalDataBoundControl)**
  - TreeView
  - Menu



ListControl

- **ListControl is the base class for**
  - DropDownList
  - ListBox
  - CheckBoxList
  - RadioButtonList
  - BulletedList
- **Contains ListItems**
  - Text and Value
- **Can be DataBound**
  - DataTextField and DataValueField



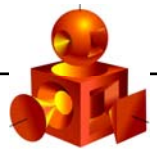
Populating a ListControl

- **Manually populating a DropDownList in CodeBehind**

```
foreach (Person person in PersonFactory.GetPersons())
{
    DropDownList1.Items.Add(new ListItem(person.FullName, person.Id.ToString()));
}
```

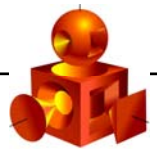
- **DataBinding the DropDownList in CodeBehind**

```
DropDownList1.DataSource = PersonFactory.GetPersons();
DropDownList1.DataTextField = "FullName";
DropDownList1.DataValueField = "Id";
DropDownList1.DataBind();
```



DataSource Controls

- **Represents data**
  - By declaration
  - Methods for fetching and changing data
  - Can take parameters
- **Designtime support**
  - Taskpad
  - Wizards
  - Property window
- **No visual interface during runtime**



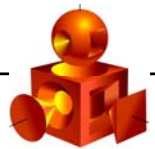
Declarative Binding

- **Create a DataSource control in markup**

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
    TypeName="PersonFactory" SelectMethod="GetPersons" >
</asp:ObjectDataSource>
```

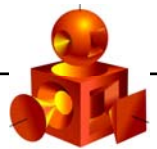
- **Connect it to the databound control**

```
<asp:DropDownList ID="DropDownList1" runat="server"
    DataSourceID="ObjectDataSource1"
    DataTextField="FullName" DataValueField="Id">
</asp:DropDownList>
```



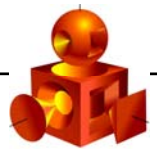
CompositeDataBoundControl

- **CompositeDataBoundControl is the base class for**
  - GridView
  - DataList
  - ListView
  - Repeater
  - DetailsView
  - FormView
- **Can display multiple columns**
- **Support for add, edit and remove**



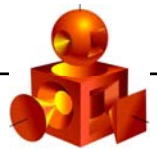
Templating CompositeDataBoundControls

- **Collections are shown with**
  - GridView
  - DataList
  - ListView
  - Repeater
- **Single items are shown with**
  - DetailsView
  - FormView



Working With Templated Controls

- **Controls placed inside templated controls don't get their own holder variables in codebehind**
  - Use the FindControl method
  - CommandName, CommandArgument and CommandSource in ItemCommand and RowCommand events
- **We define the DataBinding**
  - In markup or using the designer
- **Seperate templates for displaying, editing and updating**



DataBinding syntax

- **<%# and %> are used to evaluate expressions once for each row displayed**

```
<%# GetPostalName(Eval("PostCode")) %>
```

- **Can be used to set any property on a control in a template**

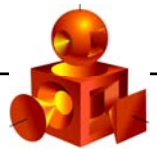
```
<asp:Label ID="Label1" runat="server" Text='<%# Eval("PostCode") %>' />
```

- **DataBinder object to get any property on current row's dataitem**
  - Eval method for one-way (read-only) databinding

```
<asp:Label ID="Label1" runat="server"
    Text='<%# Eval("ProductName", "Order {0}") %>' />
```

  - Bind method for two-way databinding

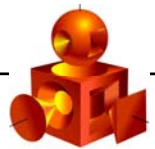
```
<asp:DropDownList ID="DropDownList1" runat="server"
    SelectedValue='<%# Bind("CategoryID") %>' />
```



DataBinding to a Lookup Table

1. **Create a DropDownList in the appropriate template,**
2. **DataBind the DropDownList to the lookup table**
3. **DataBind the DropDownList's SelectedValue to the data in the CompositeDataboundControls**


```
<EditItemTemplate>
    <asp:DropDownList ID="CategoryDropDownList" runat="server"
        DataSourceID="CategoriesDataSource"
        DataTextField="CategoryName" DataValueField="CategoryID"
        SelectedValue='<%# Bind("CategoryID") %>' />
</EditItemTemplate>
```

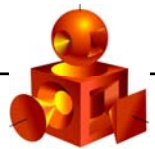


CommandName and CommandArgument

- **Properties on IButtonControl interface**

- **Identifies buttons in a CompositeDataBoundControl**

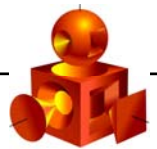
```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
    DataKeyNames="ProductID" DataSourceID="SqlDataSource1"
    OnRowCommand="GridView1_RowCommand">
    <Columns>
        <asp:BoundField DataField="ProductName" HeaderText="ProductName" />
        <asp:BoundField DataField="UnitPrice" HeaderText="UnitPrice" />
        <asp:TemplateField>
            <ItemTemplate>
                <asp:Button ID="Button1" runat="server"
                    CommandArgument='<% # Eval("ProductID") %>'
                    CommandName="Order"
                    Text='<% # Eval("ProductName", "Order {0}") %>' />
            </ItemTemplate>
        </asp:TemplateField>
    </Columns>
</asp:GridView>
```



ItemCommand and RowCommand events

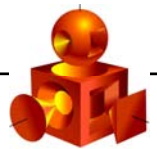
- **EventArgs include the CommandName and CommandArgument**
  - CommandName identifies the action for a button
  - CommandArgument identifies the row of the pressed button

```
protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
{
    switch (e.CommandName)
    {
        case "Order":
            int productID = int.Parse(e.CommandArgument);
            //Code to add product to shoppingcart...
            break;
    }
}
```



Standard CommandNames in databound controls

- **EditItemTemplate**
  - Update, Cancel
- **InsertItemTemplate**
  - Insert, Cancel
- **ItemTemplate, AlternatingItemTemplate**
  - Edit, Delete, New



ListView

- **LayoutTemplate wraps around the list**
- **ItemTemplate represents the items in the list**
  - ItemTemplate replaces the itemplaceholder element

```
<asp:ListView ID="_customersListView" runat="server"
    DataSourceID="_customersDataSource" ItemPlaceholderID="itempPlaceHolder">

    <LayoutTemplate>
        <ul>
            <asp:PlaceHolder id="itemPlaceHolder" runat="server" />
        </ul>
    </LayoutTemplate>

    <ItemTemplate>
        <li>
            <%# Eval("CompanyName") %>
        </li>
    </ItemTemplate>

</asp:ListView>
```

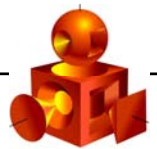
```
<ul>
    <li>Alfreds Futterkiste</li>
    <li>Ana Trujillo Emparedados y helados</li>
    <li>...</li>
</ul>
```

ListView

- **ListView enables grouping in ways no other controls can**
  - You can control the grouping with GroupTemplate

```
<asp:ListView ID="_customersListView" runat="server"
    DataSourceID="_customersDataSource" GroupItemCount="3">

    <LayoutTemplate>
        <table>
            <asp:PlaceHolder id="groupPlaceHolder" runat="server" />
        </table>
    </LayoutTemplate>
    <GroupTemplate>
        <tr>
            <asp:PlaceHolder id="itemPlaceHolder" runat="server" />
        </tr>
    </GroupTemplate>
    <ItemTemplate>
        <td>
            <%# Eval("CompanyName") %>
        </td>
    </ItemTemplate>

</asp:ListView>
```

ALFKI	ANATR	ANTON
AROUT	BERGS	BLAUS
BLONP	BOLID	BONAP
BOTTM		

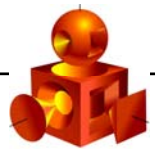


DataPager

- **Controls paging of any control implementing IPageableItemContainer**
  - The ListView control

- **Can use preset paging controls, templates or a combination of both**

```
<asp:DataPager ID="DataPager1" runat="server" PagedControlID="_customersListView">
    <Fields>
        <asp:NextPreviousPagerField ButtonType="Link" ShowFirstPageButton="True"
            ShowNextPageButton="False" ShowPreviousPageButton="True" />

        <asp:NumericPagerField />

        <asp:NextPreviousPagerField ButtonType="Link" ShowLastPageButton="True"
             ShowNextPageButton="True" ShowPreviousPageButton="False" />

        <asp:TemplatePagerField>
            <PagerTemplate>
                Put your custom content here...
            </PagerTemplate>
        </asp:TemplatePagerField>
    </Fields>
</asp:DataPager>
```



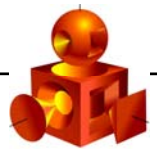
HierarchicalDataBoundControl

- **Connects to Hierarchical data**
  - Sample XML file

```
<Organization>
    <Department ID="d1" Name="Administration">
        <Person ID="p1" Name= "Kjersti Sandberg"/>
        <Department ID="d1-1" Name="Operations">
            <Person ID="p2" Name="Martine Volder"/>
        </Department>
    </Department>
    <Department ID="d2" Name="Instructors">
        <Person ID="p3" Name="Arjan Einbu"/>
        <Person ID="p4" Name="Arne Laugstøl"/>
    </Department>
</Organization>
```

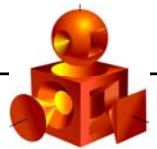
- **Markup**

```
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
          DataFile="~/App_Data/PersonsAndDepartments.xml"
          XPath=”Organization/Department">
</asp:XmlDataSource>
```



Different Types of DataSource Controls

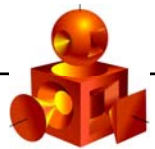
- **SqlDataSource**
- **AccessDataSource**
- **ObjectDataSource**
- **LinqDataSource**
- **EntityDataSource**
- **XmlDataSource**
- **SiteMapDataSource**



SqlDataSource and AccessDataSource

- **SqlDataSource**
  - Connects to any RDBMS data source
    - ✓ Uses ADO object to connect to data source
  - Contains SQL statements
    - ✓ SELECT
    - ✓ INSERT, UPDATE, DELETE

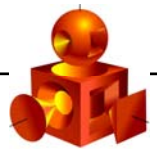
- **AccessDataSource**
  - Similar to SqlDataSource, but connects only to Access .mdb files



Parameters

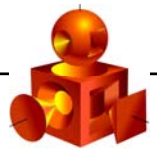
- **Parameters are passed on to the DataSource when executing queries**
- **Parameters can be automatically bound to**
  - QueryString
  - Forms
  - Controls
  - Cookies
  - Session
  - ...or custom parameter types
- **Setting a parameter from CodeBehind**
  - Create the parameter using markup
  - Set its value in CodeBehind

```
Parameter p = SqlDataSource1.SelectParameters["CategoryID"];
p.DefaultValue = "2";
```



ObjectDataSource

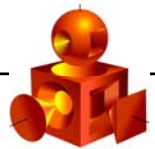
- **Connects to business objects**
  - Methods to fetch and update data
  - Properties and fields to display data
- **Connects to ADO.NET objects**
  - DataSet and DataTable
- **Connects to WebServices**



LinqDataSource and EntityDataSource

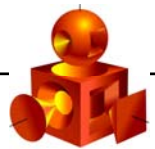
- **Connects to a LINQ-to-SQL or Entity Framework context**

- **The context handles:**
  - Select
  - Filtering and aggregates
  - Paging and ordering
  - Insert, Update, Delete

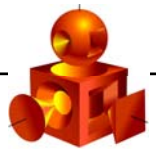


XmlDataSource and SiteMapDataSource

- **XmlDataSource**
  - Connects to XML files
  - Can apply XSLT transform to XML file
  - Can use XPath to select parts of the XML file
- **SiteMapDataSource**
  - Connects to the current sitemap provider
    - ✓ Defaults to the web.sitemap file
- **One way DataBinding**
  - Read-only

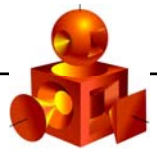


Chapter 7
Creating Controls



Creating Controls

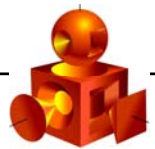
- **Encapsulates functionality**
- **Consistent interface**
- **Reusable**
  - Saves development time
  - Reduces bugs from duplicate code



Different Flavours of Controls

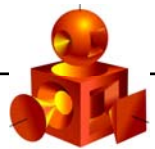
- **User Controls**
  - Similar to WebForms (ASP.NET pages)
  - Use the designer to layout the contents of the control
  - Quick to implement

- **Custom Control**
  - Inherits from WebControl or subclass
  - Highly customizable and flexible



Creating UserControls

- **@Control directive needed in .ascx file**
  - No boilerplate HTML to start or end the control
- **Same markup and codebehind model as ASP.NET pages**
- **Can be created from a ASP.NET page**
  - Remove tags outside of and including the <form> tags
  - Change @Page directive to @Control directive
  - Change inheritance to UserControl
  - Change file extension to .ascx



Using UserControls

- **Drag from Solution Explorer onto form**
  - Will register the control for use on the form

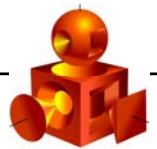
```
<%@ Register Src="SimpleCalculatorUserControl.ascx"
TagName="SimpleCalculatorUserControl" TagPrefix="uc1" %>
```

  - And add the markup for the control

```
<uc1:SimpleCalculatorUserControl ID="SimpleCalculatorUserControl1"
runat="server" />
```

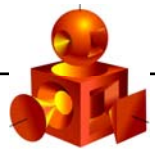
- **Add dynamically from CodeBehind**
  - Register the control in markup
  - Create it in CodeBehind

```
SimpleCalculatorUserControl myControl =
    (SimpleCalculatorUserControl)LoadControl("SimpleCalculatorUserControl.ascx");
form1.Controls.Add(myControl);
```



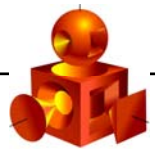
Creating Custom Controls

- **Choice of base class**
  - WebControl
  - CompositeControl
  - Existing controls
- **Compiles to a class library (.dll)**
  - Can containt multiple custom controls
  - Easy to reuse
- **No designer support**
  - Everything has to be done in code



Subclassing WebControl

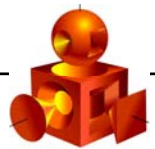
- **Full control over the rendered HTML**
  - By overriding the Render virtual method
- **Override the Render virtual method**



Subclassing CompositeControl

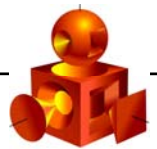
- **Controls that contain other controls**
- **Override the CreateChildControls virtual method**
- **Use the EnsureChildControls method before accessing a contained control**

```
public string Postcode
{
    get
    {
        EnsureChildControls();
        return _postcodeTextBox.Text;
    }
    set
    {
        EnsureChildControls();
        _postcodeTextBox.Text = value;
    }
}
```



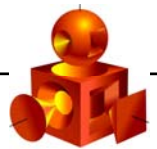
Basic Plumbing for different Web Custom Controls

```
public class MyControl : WebControl
{
    //Implement own fields, properties, events, methods etc. here!

    protected override void Render(HtmlTextWriter writer)
    {
        writer.WriteFullBeginTag("div");
        writer.Write("We have <em>full</em> control over the rendering!");
        writer.WriteEndTag("div");
    }
}

public class MyControl : CompositeControl
{
    //Implement own fields, properties, events, methods etc. here!

    protected override void CreateChildControls()
    {
        Controls.Add(new LiteralControl("We build this from pieces!"));
        Controls.Add(new TextBox());
    }
}
```



Adding Custom Controls to the ToolBox

- **Need to be in a seperate dll**
- **Add attributes**
  - Define a tagprefix
  - Set a designer class
  - Set a custom icon



Using Custom Controls

- **Drag from toolbox onto form**
  - Will register the control for use on the form
    - ✓ If it isn't allready registered in the config files

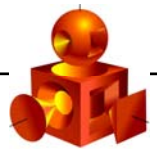
```
<% @ Register Assembly="Chap7" Namespace="Chap7" TagPrefix="ch7" %>
```

  - And add the markup for the control

```
<ch7:SimpleWebControl ID="SimpleWebControl1" runat="server" Text="" />
```

- **Add dynamically from CodeBehind**
  - No need to register the control in markup
  - Create it in CodeBehind

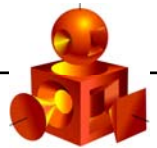
```
SimpleWebControl swc = new SimpleWebControl();
swc.Text = "testing";
form1.Controls.Add(swc);
```



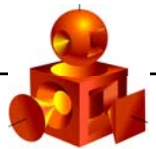
DataBinding

- **You can databind to UserControls and CustomControls**
  - You might have to edit markup directly

```
<ItemTemplate>
    <uc1:AddressUserControl ID="AddressUserControl1" runat="server"
        AddressLine1='<%# Bind("AddressLine1") %>'
        AddressLine2='<%# Bind("AddressLine2") %>'
        Postcode='<%# Bind("Postcode") %>' />
</ItemTemplate>
```

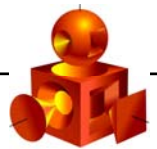


Chapter 8
Layout and Design



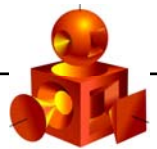
Design Elements

- **Consistent look and feel**
  - Copy markup between pages
  - Using include files for common elements
  - Frames
  - UserControls



MasterPages

- **Defines a template for multiple pages**
  - Layout
  - Design elements
  - Functionality
  - CodeBehind
- **Can’t be requested by themselves**
  - Always a part of an .aspx page
- **Similar to .aspx pages**
  - .master file extension
  - Inherits from MasterPage
  - `@Master` directive
  - `ContentPlaceHolder` elements



Using a MasterPage

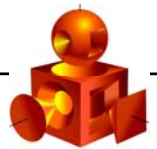
- **Create a new WebForm in Visual Studio**
  - Check ”Select master page” checkbox
  - Select a masterpage file

- **Or add `MasterPageFile` attribute to `@Page` directive**
  - Remove `<html>`, `<head>`, `<body>` and `<form>` elements
  - Add markup in the Content area(s)

- **Or configure it in web.config**

```
<system.web>
    <pages masterPageFile="MyMaster.master" />
</system.web>
```

- **Or set masterpage dynamically in CodeBehind**
  - Must be done on Page_PreInit event



Accessing the MasterPage

- **Access properties on masterpage**
  - Add the @MasterType directive

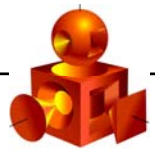
```
<% @ MasterType VirtualPath= "MyMaster.master"%>
```

  - Access through the page's Master property

```
Label1.Text = Master.MyProperty;
```

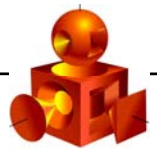
- **Access controls on masterpage**

```
Label myLabel = Master.FindControl("MyLabel") as Label;
```



Nested MasterPages

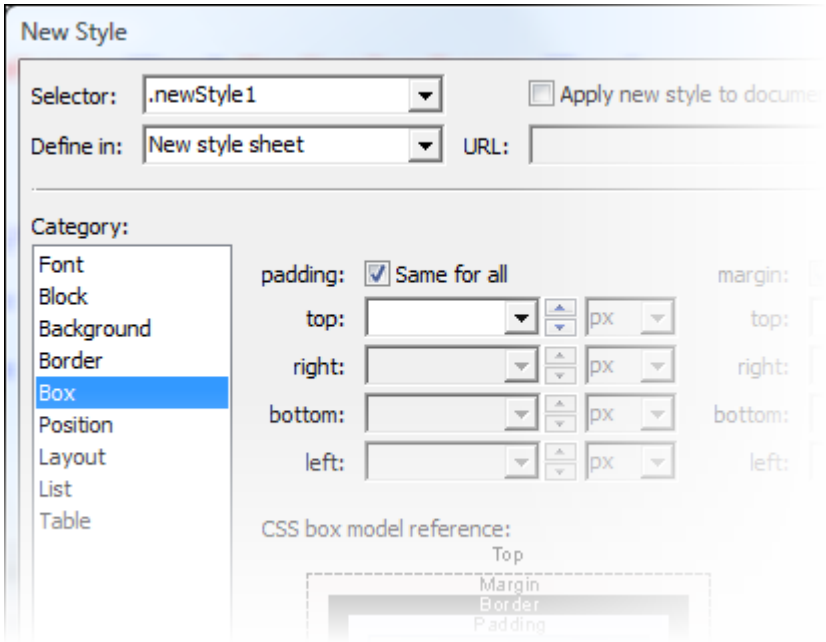
- **A masterpage can have a masterpage**
  - Add the MasterPageFile attribute to the @Master directive



CSS – Cascading StyleSheets

- **CSS defines how to display HTML elements**
  - Define styles for elements

```
h2
{
    font-family: Arial;
    background-color: Yellow;
}
```

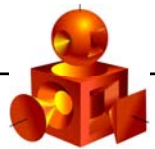


  - Define named styles

```
.myheading
{
    font-family: Verdana;
    font-weight: bold;
}
```

  - Define style for a specific control

```
#myControlId
{
    font-family: Garamond, Times New Roman, Times;
    padding: 5 10 5 10; /*top right bottom left*/
}
```



CSS Placement

- **External StyleSheets**
  - Can reference multiple stylesheets
  - Different stylesheets for different media

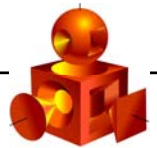
```
<head runat="server">
    <link type="text/css"  media="screen" href="~/Screen1.css" rel="stylesheet" />
    <link type="text/css"  media="screen" href="~/Screen2.css" rel="stylesheet" />
    <link type="text/css"  media="print" href="~/Print.css" rel="stylesheet" />
</head>
```

- **Internal StyleSheets**

```
<head runat="server">
    <style type="text/css">
    h1 { font-family: Arial; font-size:large; }
    </style>
</head>
```

- **Inline styles**

```
<asp:Label runat="server" ID="Label1" style="font-style:italic;" />
```



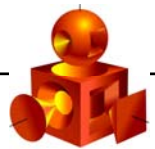
Using CSS

- **Refer to named styles with the `CssClass` attribute on server controls**

```
<asp:Label runat="server" ID="Label1" CssClass="myheading" />
```

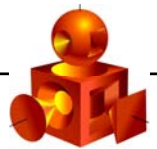
- **Refer to named styles with the `class` attribute on HTML controls**

```
<div class="myheading">
```



Themes

- **StyleSheets**
- **Skins**
- **Images and other resources**



Using Themes

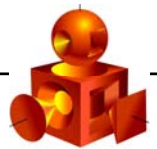
- **Adds skinning features to a website**
  - Whole site or folder

```
<system.web>
    <pages theme="MyTheme" stylesheetTheme="MyTheme" />
</system.web>
```

  - Per page

```
<%@ Page Language="C#" Theme="MyTheme" StylesheetTheme="MyTheme" %>
```

  - Dynamically set in CodeBehind
    - ✓ this.Theme must be set at PreInit stage

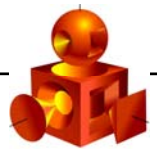


Skins

- **Skins are processed serverside**
- **Set any property on controls**
  - Also those that are not CSS'able

```
<asp:Calendar runat="server" NextMonthText="Next" PrevMonthText="Previous" />
```

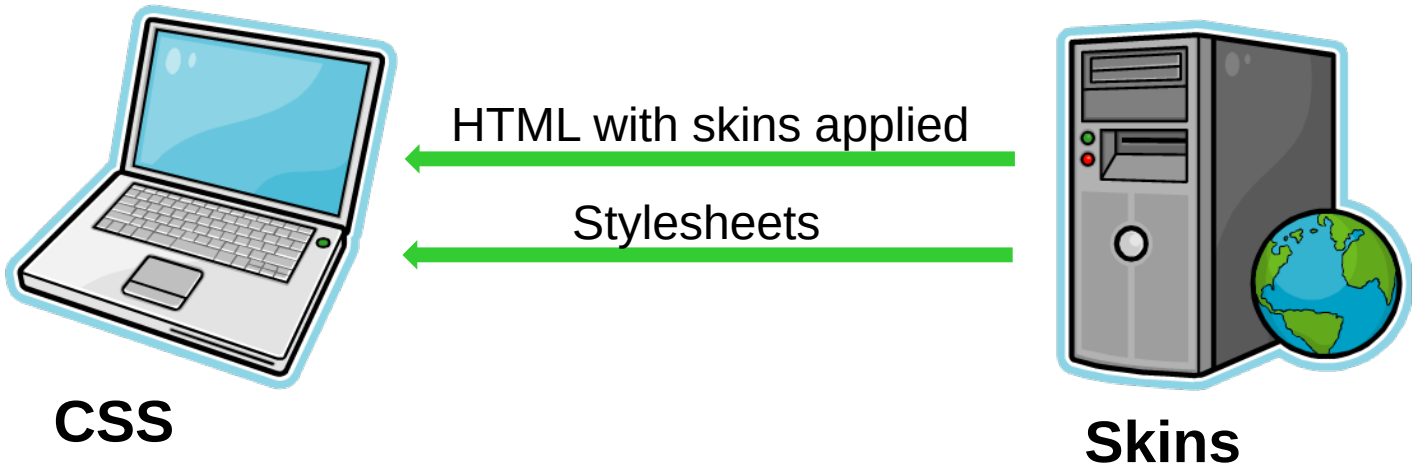
- **Named skins**
  - Use the SkinID attribute

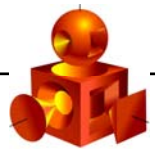


Skins vs. CSS

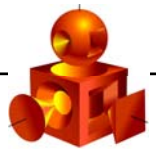
- **Skins are applied serverside**
  - We are 100% sure skins are applied
  - Any property on a control can be skinned

- **CSS is applied clientside**
  - Can be turned off by the user
  - CSS can't change the HTML



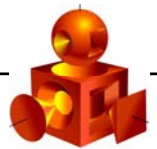


Chapter 14
AJAX



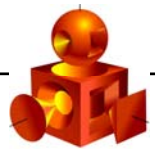
What is AJAX?

- **”Asyncronous JavaScript and XML”**
- **Use of clientside technologies**
  - XmlHttpRequest
  - JScript
  - The Document Object Model (DOM)
  - CSS
- **Enabling Out-of-band communication with server**
  - Without reloading the entire page
- **Changing the current page without reloading it**



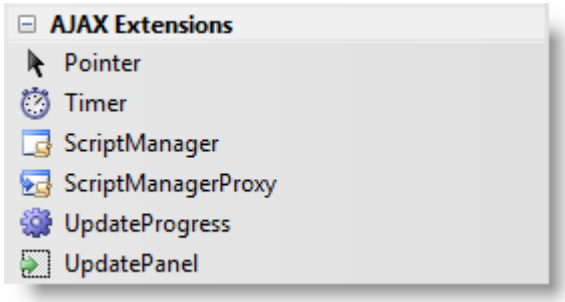
Whats Available in ASP.NET?

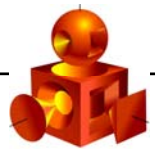
- **Some ASP.NET controls make use of out-of-band callbacks**
  - TreeView and Menu controls
  - GridView paging and sorting
  - RegularExpressionValidator



AJAX Extensions

- **Simplifies creation of custom AJAX features**
  - ScriptService
- **Additional controls**



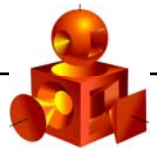


ScriptManager

- **Manages scripts**
  - References, Localization
- **Serves the correct scripts as needed**
  - ASP.NET AJAX Client Library
  - UpdatePanel, ProgressBar, Timer controls
  - Debug version or release version
  - Our own scripts

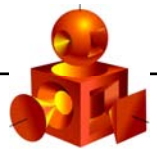
```
<asp:ScriptManager ID="ScriptManager1" runat="server">
    <Scripts>
        <asp:ScriptReference Path="myScript.js" />
    </Scripts>
</asp:ScriptManager>
```

- **Calls Sys.Application.initialize()**
  - Starts the lifecycle events



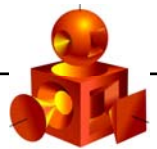
UpdatePanel

- **”Instant AJAX”**
- **AJAX-ifies ordinary PostBacks**
  - Compatible with most standard ASP.NET controls
  - Marks a part of the page for update via AJAX
    - ✓ Reduces screen flicker
- **Requires a ScriptManager**



UpdatePanel

- **UpdateMode defaults to Always**
  - Panel is automatically updated on all AJAX calls
- **ChildrenAsTriggers defaults to true**
  - All child controls in an updatepanel use AJAX
- **Triggers collection**
  - Add an outside button as AsyncPostBackTrigger
  - Add an inside button as PostBack Trigger

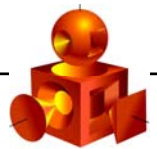


Timer

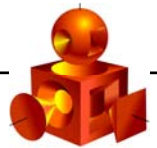
- **The Timer control is a trigger**
  - Will trigger a postback if neither
    - ✓ set as AsyncPostBackTrigger

      or
    - ✓ placed inside UpdatePanel

- **Don’t trust the accuracy of the timer**

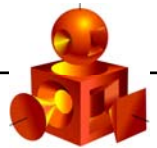


”When an operation takes time, it’s nice to tell the users we’re alive!”



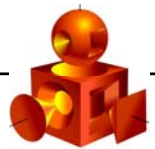
UpdateProgress

- **Displays content when waiting for content**
  - Acts as a container control
  - Typically a progress animation
- **DisplayAfter decides when to display**
- **DynamicLayout**



Calling Page Methods

- **Static methods on page can be called from browser**
  - Mark method as [WebMethod]
- **ScriptManager's EnablePageMethods attribute**
  - Set to true will create the PageMethods proxy object
- **Arguments when calling a page method**
  - All arguments the method take
  - Success callback
  - Failed callback
  - UserContext
- **Callbacks arguments**
  - result

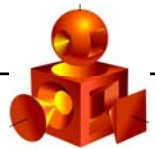


ScriptService

- **A method that is usable from multiple pages**
- **Is also exposed as a WebService**
  - Add the [ScriptService] attribute to the webservice class
    - ✓ Creates the proxy when appending /js after the webservice path
- **Register with the ScriptManager**

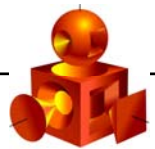
```
<asp:ScriptManager ID="ScriptManager1" runat="server" EnablePageMethods="True">
    <Services>
        <asp:ServiceReference Path="MyWebService.asmx" />
    </Services>
</asp:ScriptManager>
```

- **Same usage pattern as with calling PageMethods**



ASP.NET AJAX Application Services

- **Authentication with AJAX**
- **Access Profile properties clientside**
- **Access localized resource strings**



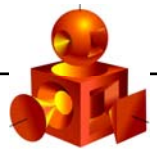
Application Services - Authentication

- **Enable AJAX logon in web.config**

```
<system.web.extensions>
    <scripting>
        <webServices>
            <authenticationService enabled="true"/>
        </webServices>
    </scripting>
</system.web.extensions>
```

- **Asyncronous methods**
  - Sys.Services.AuthenticationService.login
  - Sys.Services.AuthenticationService.logout

- **Clientside property**
  - Sys.Services.AuthenticationService.get_isLoggedIn



Application Service - Profile

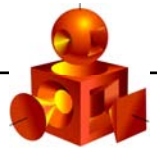
- **Enable AJAX access to profile properties**

```
<system.web.extensions>
    <scripting>
        <webServices>
            <profileService enabled="true"
                readAccessProperties="memberNo, favouriteFood"
                writeAccessProperties="favouriteFood"/>
        </webServices>
    </scripting>
</system.web.extensions>
```

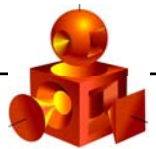
- **Asyncronous methods**
  - Sys.Services.ProfileService.load
  - Sys.Services.ProfileService.save

- **Pre-populate frequent profile fields**

- **Sys.Services.ProfilesService.properties.*favouriteFood***



Azure

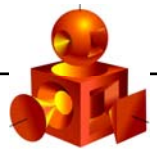


What is Windows Azure?

"The Azure™ Services Platform (Azure) is an internet-scale cloud services platform hosted in Microsoft data centers, which provides an operating system and a set of developer services that can be used individually or together."

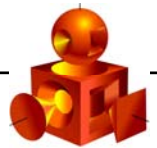
" Windows® Azure is a cloud services operating system that serves as the development, service hosting and service management environment for the Azure Services Platform. Windows Azure provides developers with on-demand compute and storage to host, scale, and manage internet or cloud applications."

www.microsoft.com/azure/whatisazure.mspx



ASP.NET and Windows Azure

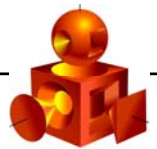
- **ASP.NET applications can run on Windows Azure Web Roles**
- **Some differences**
  - Not access to a local filesystem
  - New APIs for Azure specific functionality
- **Azure Storage Services**
  - Blob storage
  - Queues
  - Tables



Developing for Windows Azure

- **Azure SDK**
  - Allows running Azure aware apps on development machine (local fabric)

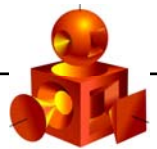
- **Register for an account at www.microsoft.com/azure/register.mspx**
  - Might take a day or three to get access...

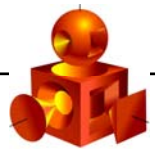


Porting a web app to Azure

- **Web site projects must first be converted to Web application projects**
- **Add an Empty Cload Service project to the solution**
- **Right click the Roles node, and select Add → Web Role Project in solution...**
  - Edit the web application's project file to be enable this action:
    - ✓ Unload the web application project
    - ✓ Edit the .csproj (or .vbproj) file
    - ✓ Add the "RoleType" element with the value of "Web" to the first PropertyGroup element

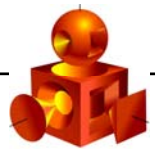
```
<PropertyGroup>
    <RoleType>Web</RoleType>
    <!– ...more properties... -->
</PropertyGroup>
```





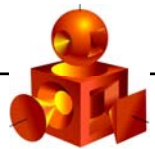
In the 5-day course there will be more about:

- **Chapter 2: Controls**
  - More about some controls
- **Chapter 3: Data and Databinding**
  - Parameters and custom parameters
  - More tips and tricks
- **Chapter 7: Creating Controls**
  - Creating a templated control
- **Chapter 8: Layout and Design**
  - More about everything
- **Chapter 15: AJAX**
  - More JavaScript
  - The scriptmanager
  - ASP.NET BCL



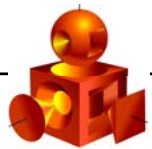
And also include these topics:

- **Chapter 4: Validation**
- **Chapter 5: Codebehind**
- **Chapter 6: Authentication and Authorization**
- **Chapter 9: Site Navigation**
- **Chapter 10: Webparts**
- **Chapter 11: Internationalization**
- **Chapter 12: Accessibility**
- **Chapter 13: Deployment**



And last, but not least…
Lots of hands-on exercises!

www.programutvikling.no



Further Information

- **Where can I go to get more information on ASP.NET and web development?**
  - www.asp.net/learn (Tutorials on ASP.NET)
  - www.w3schools.com (Tutorials on other web technologies)
  - www.codeproject.com (Articles and forums)
  - msdn.microsoft.com (Microsoft Developer Network)
  - stackoverflow.com (Question and answers site for devs.)