User Controls

Upphovsrätt för detta verk

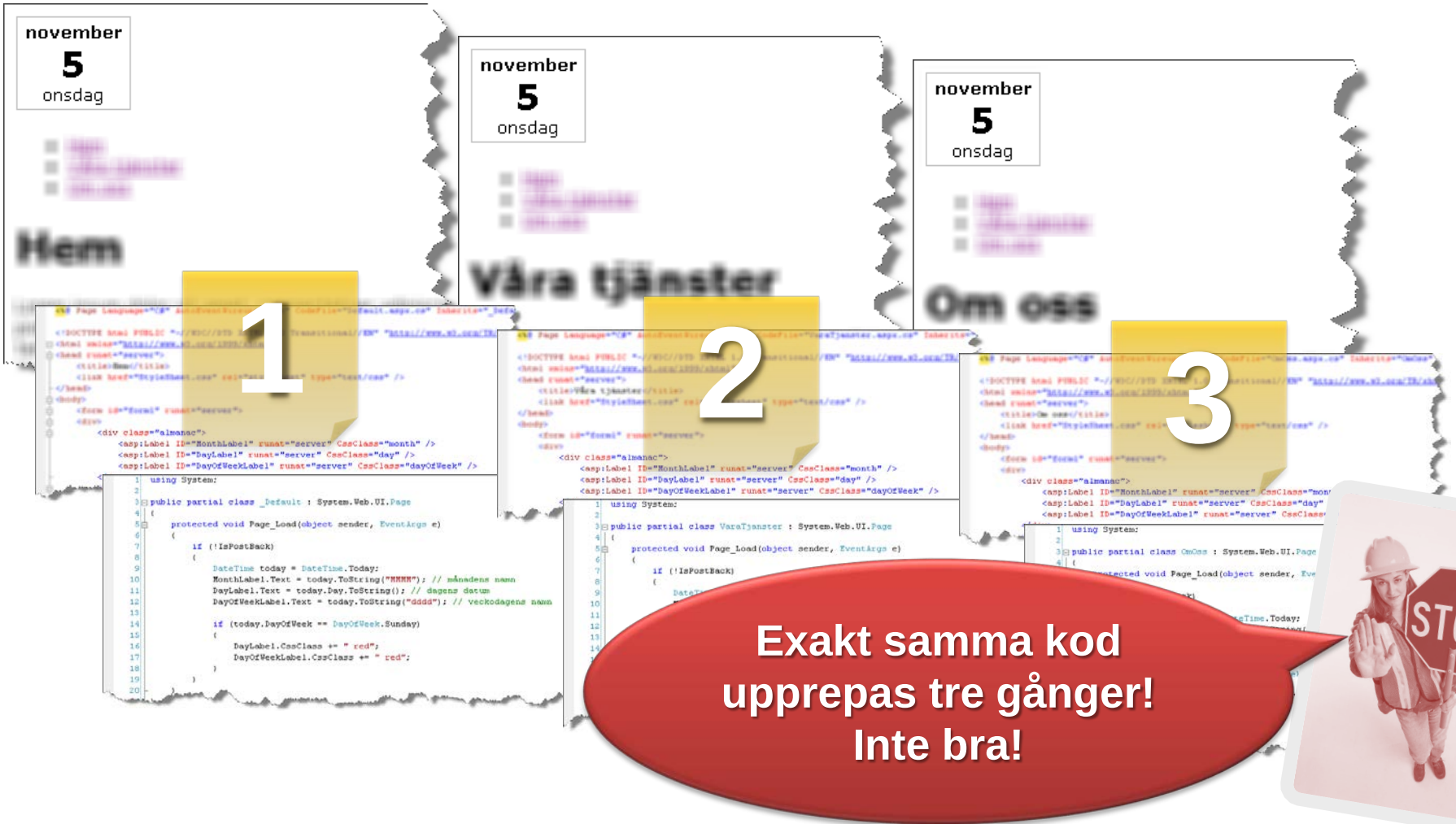
Detta verk är framtaget i anslutning till kursen ASP.NET Web Forms vid Linnéuniversitetet.

Du får använda detta verk så här:

Återanvändning av kod?

ÄR DU SKYLDIG
till att någon gång i flera webbformulär upprepat i princip exakt samma kodavsnitt?

Redundant kod!

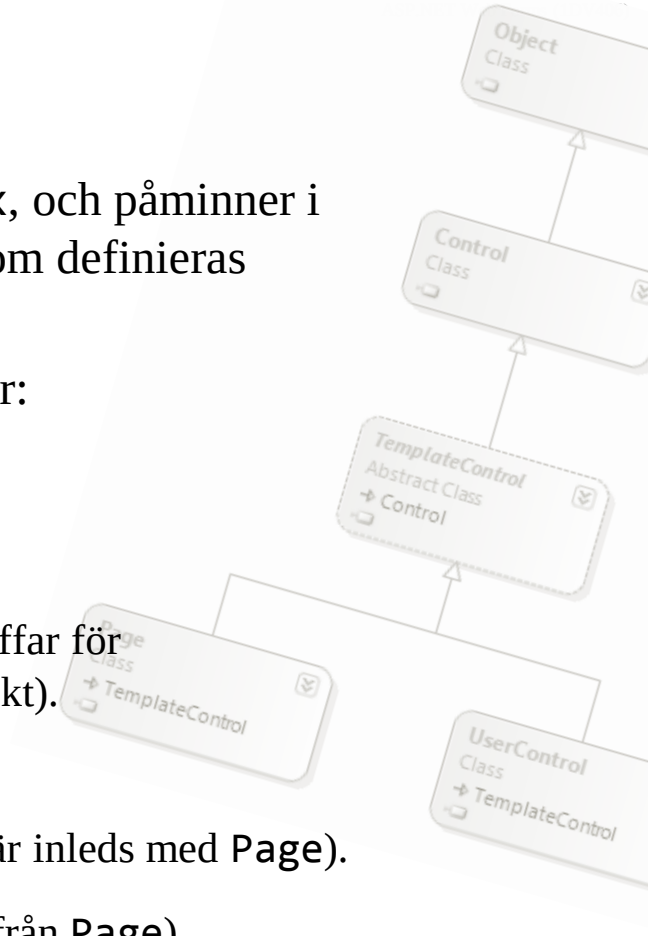


Egna kontroller

- Förutom att använda HTML- och serverkontrollerna kan du skapa och använda egna kontroller. Att skapa egna kontroller är ett sätt att återanvända kod.
- Egna kontroller fungerar som HTML- och serverkontrollerna och kan användas av flera webbformulär inom samma webbapplikation.
- Det finns två huvudkategorier av egna kontroller:
  - ”User controls” (enkelt att implementera)
    - En ”user control” är en kontroll som kan innehålla statisk HTML, webbserverkontroller och kod, d.v.s. allt ett vanligt webbformulär kan innehålla.
    - En ”user control” kan ha egna egenskaper och händelser.

  - ”Custom server controls” (inte fullt så enkelt att implementera)
    - En ”custom server control” är en kontroll du bygger upp helt programmatiskt och kompileras alltid till en ”assembly”.
    - Du kan rendera innehållet från grunden själv, ärva och utöka en befintlig kontroll eller genom att kombinera en grupp kontinuerliga kontroller (ibland kallad ”composite control”).

Grundläggande om ”user controls”



- En ”user control” definieras i en fil med ändelsen `.ascx`, och påminner i mycket stor utsträckning om ett vanligt webbformulär som definieras i filer med ändelsen `.aspx`.
- En ”user control” påminner mycket om ett webbformulär:
  - Kan innehålla olika typer av kontroller.
  - Kan ha en ”code-behind”-fil.
  - Samma händelser (`Init`, `Load`, …, `PreRender`, …) inträffar för en ”user control” som för ett webbformulär (ett `Page`-objekt).
- …men det finns skillnader också. En ”user control”…
  - …definition inleds med direktivet `Control` (webbformulär inleds med `Page`).
  - …ärver från klassen `UserControl` (webbformulär ärver från `Page`).
  - …kan inte efterfrågas direkt av en klient, utan kan bara hämtas om den ingår som en del i ett webbformulär.

En enkel ”user control”

- För att skapa en ”user control” väljer du **Website ► New Item** och **Web User Control**.

App/UserControls/Almanac.ascx

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="Almanac.ascx.cs" Inherits="App_UserControls_Almanac" %>
<div class="almanac">
    <asp:Label ID="MonthLabel" runat="server" CssClass="month" />
    <asp:Label ID="DayLabel" runat="server" CssClass="day" />
    <asp:Label ID="DayOfWeekLabel" runat="server" CssClass="dayOfWeek" />
</div>
```

App/UserControls/Almanac.ascx.cs

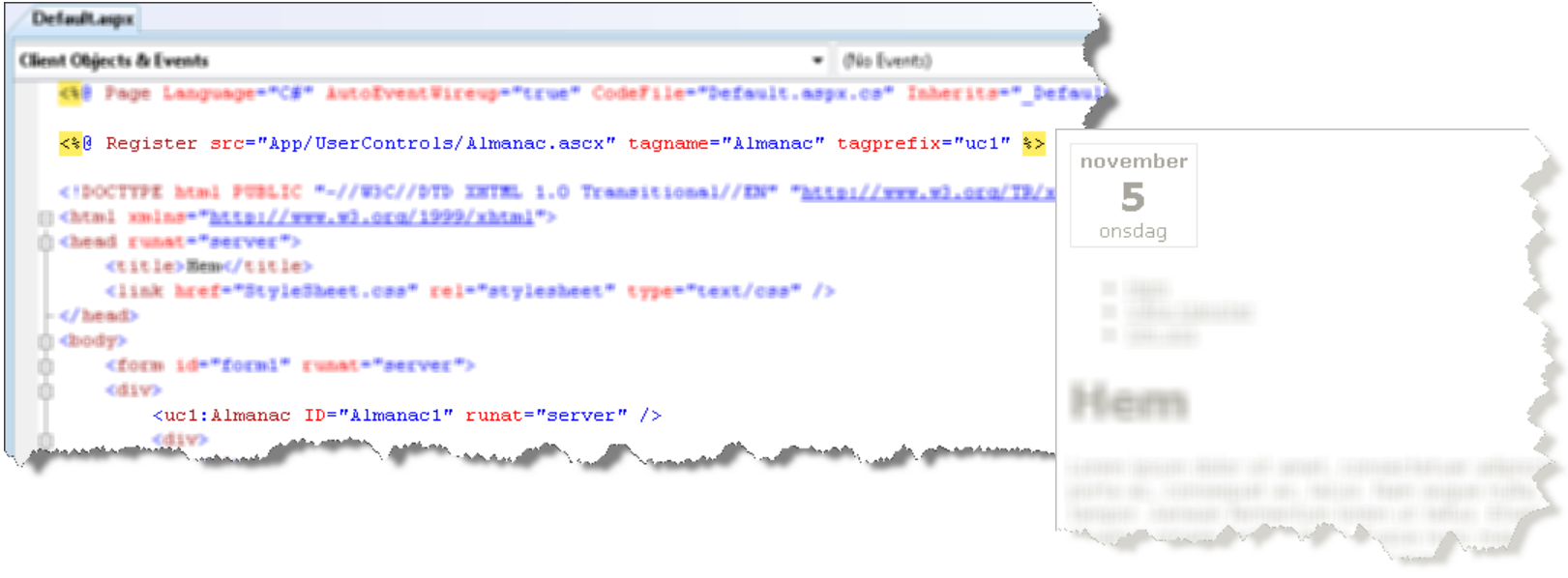
```
using System;

public partial class App_UserControls_Almanac : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            DateTime today = DateTime.Today;
            MonthLabel.Text = today.ToString("MMMM"); // månadens namn
            DayLabel.Text = today.Day.ToString(); // dagens datum
            DayOfWeekLabel.Text = today.ToString("dddd"); // veckodagens namn

            if (today.DayOfWeek == DayOfWeek.Sunday)
            {
                DayLabel.CssClass += " red";
                DayOfWeekLabel.CssClass += " red";
```

Bra, nu finns HTML, serverkontroller och C#-kod i ”code-behind”-filen bara på ett ställe.

Koden som upprepades av de tre webbformulären placeras i en ”user control”.

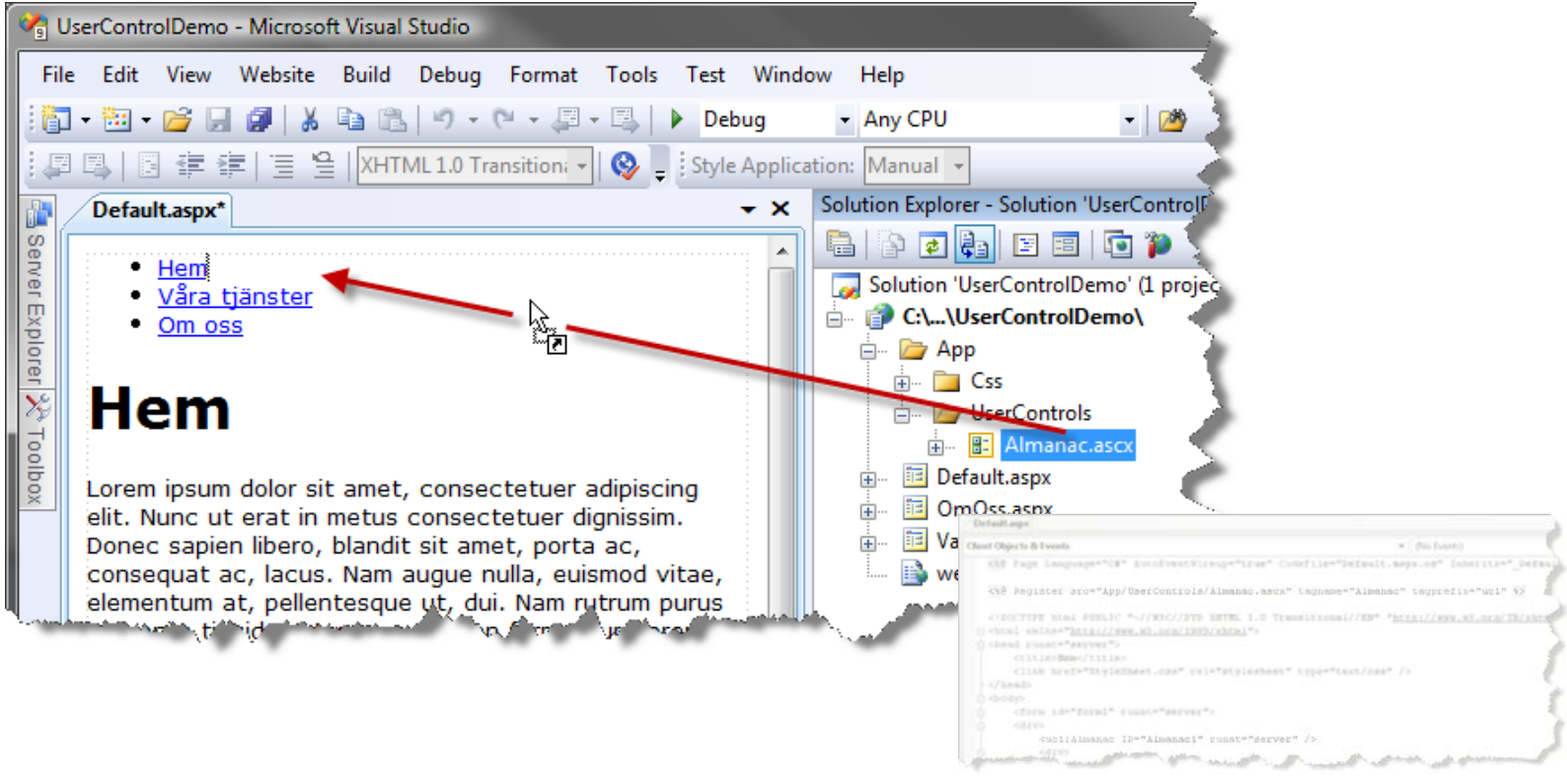
Att använda en ”user control”

- För att använda en ”user control” måste den placeras i ett webbformulär.
  - Med direktivet Register talar du om för webbformuläret att du vill använda en viss ”user control”.
  - Du lägger till en ”user control” i ett webbformulär genom att använda ett för aktuell ”user control” specifikt taggprefix (`uc1`) och taggnamn (`Almanac`).



Att lägga till en ”user control”

- Enklaste sättet att lägga till en ”user control” är att dra .ascx-filen från fönstret Solution Explorer och släppa den i webbformuläret, som måste befinna sig i designläge.



”User controls” och egenskaper

- För att göra en ”user control” mer flexibel, och därmed mer användbar, kan du komplettera den med en eller flera publika egenskaper.

```
1  using System;
2
3  public partial class App_UserControls_Almanac : System.Web.UI.UserCont
4  {
5      public DateTime? Date { get; set;}
6
7      protected void Page_Load(object sender, EventArgs e)
8      {
9          if (!IsPostBack)
10         {
11             DateTime today = Date.HasValue ? Date.Value : DateTime.Today;
12             MonthLabel.Text = today.ToString("MMMM"); // månadens namn
```

```
<div>
    <uc1:Almanac ID="Almanac1" runat="server" Date="2008-12-24" />
    <div>
```

Typen DateTime? kan ha värdet null, vilket blir fallet om egenskapen inte tilldelas ett värde. Har egenkapen Date värdet null används dagens datum.

Om du använder egenskaper av typer som int, double, DateTime, etc., kan du sätta deras värden med en sträng. ASP.NET typomvandlar automatiskt strängen till egenskapens typ.

Exponera kontroller i en ”user control”

- Vill du kunna påverka enskilda kontroller i en ”user control” från ett webbformulär måste du använda publika egenskaper (eller metoder).

```
1  using System;
2
3  public partial class App_UserControls_Almanac : System.Web.UI.UserControl
4  {
5      public DateTime? Date { get; set;}
6
7      public bool DayOfWeekVisible
8      {
9          get { return DayOfWeekLabel.Visible; }
10         set { DayOfWeekLabel.Visible = value; }
11     }
12
13     protected void Page_Load(object sender, EventArgs e)
```

Inget separat fält behövs för att lagra informationen eftersom Label-kontrollen redan har det som krävs.

```
<uc1:Almanac ID="Almanac1" runat="server" Date="2008-12-24" DayOfWeekVisible="false" />
<div>
```

I vilken ordning inträffar händelserna?

- ✓ Då du lägger till egenskaper till en ”user control” är det viktigt att du känner till i vilken ordning olika händelser inträffar.
  1. Webbformuläret efterfrågas av en klient.
  2. Den ”user control” webbformuläret använder skapas. Variabler initieras till sina standardvärden, eller så initieras i kontrollens konstruktor.
  3. Om någon egenskaps värde sätts i taggen i webbformuläret sker det nu.
  4. Hanterarmetoden `Page_Load` (för händelsen `Load`) i webbformuläret körs (det är fullt möjligt att initiera en ”user control” här – den har ju redan skapats).
  5. Hanterarmetoden `Page_Load` (för händelsen `Load`) i aktuell ”user control” körs (självklart kan kod som sköter initieringen av en ”user control” placeras här).

Fundera både en och två gånger innan du placerar kod som initierar en ”user control” i dess hanterarmetod `Page_Load`. Du kan oavsiktligt skriva över värden som satt via taggen i webbformuläret.

Ja precis, det är ju därför det är så viktigt att känna till i vilken ordning olika händelser inträffar.

”User controls” och händelser (1 av 2)

- Genom att använda händelser kan du låta en ”user control” meddela ett webbformulär att något inträffat.

```
using System;

public partial class App_UserControls_Almanac : System.Web.UI.UserControl
{
    // Definierar en publik händelsemedlem.
    public event EventHandler YesItIsMonday;

    public DateTime? Date { get; set; }

    public bool DayOfWeekVisible...

    protected void Page_Load(object sender, EventArgs e)
    {
        // En händlse ska bara utlösas om egenskapen Date inte har något värde och
        // om veckodagen är en måndag.
        if (Date.HasValue == false &&
            DateTime.Today.DayOfWeek == DayOfWeek.Monday)
        {
            // Utlöser en händelse bara om det finns en abonnent.
            if (YesItIsMonday != null)
            {
                YesItIsMonday(this, EventArgs.Empty);
            }
        }

        if (!IsPostBack)
```

Definierar en händelse som ett webbformulär kan abonnera på.

Utlöser själva händelsen och skickar med de argument som brukar skickas i samband med händelser.

”User controls” och händelser (2 av 2)

- Enklaste sättet för webbformuläret att abonnera på händelsen `YesItIsMonday` är att definiera det i taggen för aktuell ”user control”.

```
<form id="form1" runat="server">
<div>
    <uc1:Almanac ID="Almanac1" runat="server" OnYesItIsMonday="Almanac1_YesItIsMonday" />
    <asp:Image ID="LaughingImage" runat="server" ImageUrl="App/Images/laughing.jpg" Visible="false" />
<div>
<ul>
```

```
using System;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)...

    protected void Almanac1_YesItIsMonday(object sender, EventArgs e)
    {
        LaughingImage.Visible = true;
    }
}
```

Det är bara att lägga till prefixet On framför händelsen och ange namnet på den metod som ska köras då händelsen inträffar.

I webbformulärets ”code-behind”-fil finns hanterarmetoden med den signatur som krävs.

Du kan även ”manuellt” abonnera på en händelse genom att i `Page_Load` skriva `Almanac1.YesItIsMonday += new EventHandler(Almanac1_YesItIsMonday);`.

Händelser och egna argument (1 av 3)

- Det fullt möjligt att skicka med information med händelsen. För att göra det måste du skapa en egen klass som ärver från klassen `EventArgs`.

```
54  public class YesItIsMondayEventArgs : EventArgs
55  {
56      public int HoursToTuesday { get; private set; }
57      public int MinutesToTuesday { get; private set; }
58
59      public YesItIsMondayEventArgs(int hours, int minutes)
60      {
61          HoursToTuesday = hours;
62          MinutesToTuesday = minutes;
63      }
64  }
```

Här används autoimplementerade egenskaper som är ”read-only” . Eftersom set är privat kan deras värden endast sättas inne i klassen vilket är precis vad som sker i konstruktorn.

Aha, det enda jag behöver göra nu är väl att instansiera ett objekt av den här klassen och skicka med referensen till objektet istället för att använda `EventArgs.Empty`. Enkelt och smart!

Händelser och egna argument (2 av 3)

```
using System;

public partial class App_UserControls_Almanac : System.Web.UI.UserControl
{
    // Definierar ett nytt delegat som representerar signaturen som
    // händelsen YesItIsMonday har.
    public delegate void YesItIsMondayEventHandler(object sender,
        YesItIsMondayEventArgs e);

    // Definierar en publik händelsemedlem.
    public event YesItIsMondayEventHandler YesItIsMonday;

    public DateTime? Date { get; set;}

    public bool DayOfWeekVisible...

    protected void Page_Load(object sender, EventArgs e)
    {
        // En händelse ska bara utlösas om egenskapen Date inte har något värde o
        // om veckodagen är en måndag.
        if (Date.HasValue == false &&
            DateTime.Today.DayOfWeek == DayOfWeek.Monday)
        {
            // Utlöser en händelse bara om det finns en abonnent.
            if (YesItIsMonday != null)
            {
                TimeSpan diff = DateTime.Today.AddDays(1) - DateTime.Now;
                YesItIsMondayEventArgs ea =
                    new YesItIsMondayEventArgs(diff.Hours, diff.Minutes);
                YesItIsMonday(this, ea);
            }
        }
```

Definierar delegatet som händelsen YesItIsMonday använder (beskriver signaturen abonnentens hanterarmetod måste ha).

Definierar händelsen YesItIsMonday med hjälp av delegatet YesItIsMondayEventHandler.

Instansierar ett objekt av typen YesItIsMondayEventArgs och skickar med referensen till objektet då händelsen utlöses.

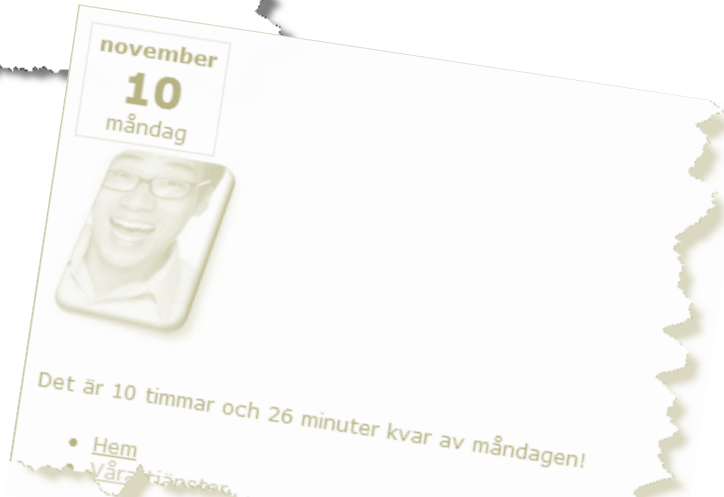
Händelser och egna argument (3 av 3)

- ✓ Hanterarmetoden till händelsen `YesItIsMonday` kan nu ta del av informationen som skickas som en del av händelsen.

```
using System;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void Almanac1_YesItIsMonday(object sender, YesItIsMondayEventArgs e)
    {
        LaughingImage.Visible = true;
        TuesdayLiteral.Text = String.Format("Det är {0} timmar och {1} minuter kvar av måndagen!",
            e.HoursToTuesday, e.MinutesToTuesday);
    }
}
```

Hanterarmetoden har den signatur som delegatet definierar och kan därmed ta hand om argumentet som skickades då händelsen utlöstes.

Vad bra! Nu har du ett sätt att skicka information från en ”user control” till ett webbformulär med hjälp av händelser. Dessutom så bestämmer webbformuläret själv vilka händelser den ska abonnera på.



Mer att läsa och lära….

- ✓ Dynamically Loading User Controls (695-699)
- ✓ Partial Page Caching (699-701)