

# Serverkontroller

# Upphovsrätt för detta verk

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

## Du får använda detta verk så här:

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



Creative Commons Erkännande-IckeKommersiell-DelaLika 2.5 Sverige licens.

<http://creativecommons.org/licenses/by-nc-sa/2.5/se/>

## Det betyder att du i icke-kommersiella syften får:

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


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

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

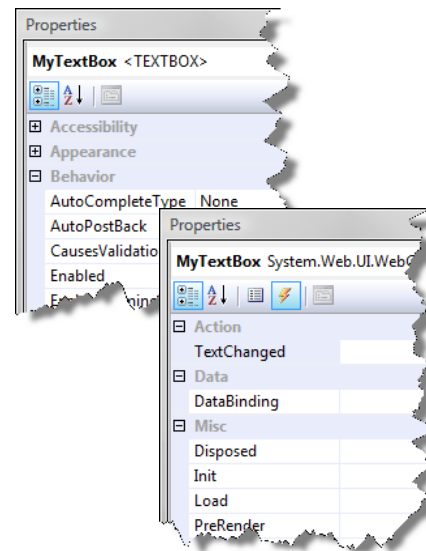
# Vad är en serverkontroll?

- ✓ Flertalet serverkontroller kan ses som objekt som (oftast) representerar ett visuellt element.
  - Då en sida renderas, renderas de serverkontroller sidan innehåller till HTML, JavaScript m.m.  

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



```
<input name="MyTextBox" type="text" id="MyTextBox" />
```
  - En serverkontroll har...
    - ...egenskaper för att du enklare ska kunna bestämma presentation och beteende.
    - ...olika händelser ("events") du kan abonnera på genom att implementera hanterarmetoder.



# “Server Controls”

## ✓ HTML Server Controls

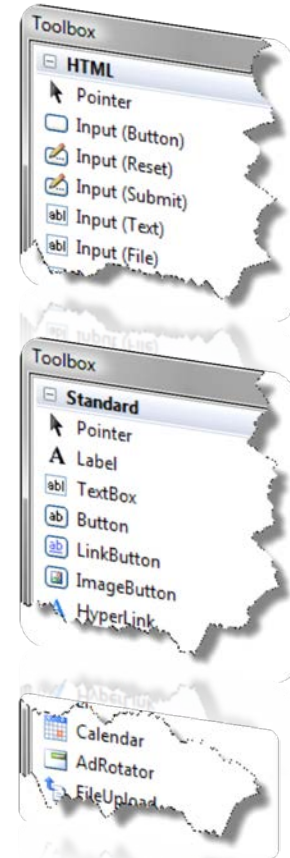
- Kontroller som kapslar in vanliga standard HTML-element fast med attributet `runat="server"`. Du hittar dem under fliken HTML i fönstret **Toolbox**.

## ✓ Web Controls

- Dessa är de vanligaste att jobba med då de har utökad funktionalitet jämfört med HTML Server Controls – mer genomtänkta egenskaper och metoder. Under fliken **Standard** hittar du kontroller som tillhör denna kategori.

## ✓ Rich Controls

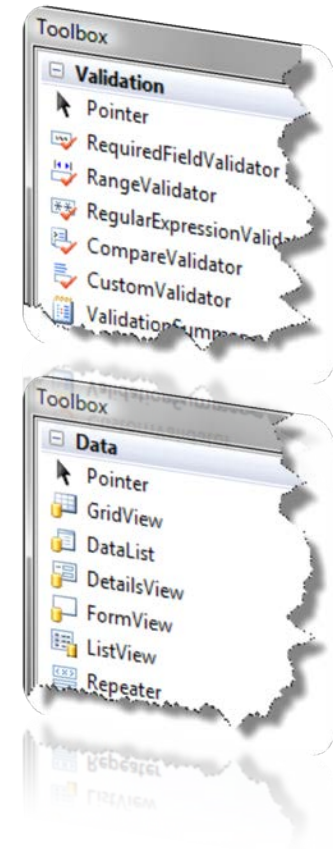
- Lite mer avancerade kontroller som ofta genererar flera XHTML-element och ibland även JavaScript. Exempel är **Calendar**, **AdRotator** och **TreeView**. De har ingen egen flik utan du finner dem under **Standard**.



# Fler "Server Controls"

- ✓ Validation Controls
  - Hjälper dig som utvecklare att hantera validering - både på klient och på server.
- ✓ Data Controls
  - Här hittar du kontroller som hjälper dig hämta och presentera data. Vi kan också få hjälp med kodkrävande saker som "*paging*", "*sorting*" och mycket mer.
- ✓ Navigation Controls, Login Controls, Web parts Controls, ASP.NET AJAX Controls, Reporting Controls

**Behandlas inte under kursen!**

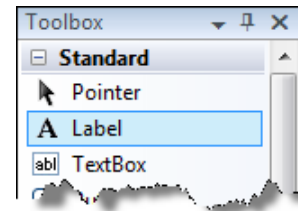


# Så kan du presentera text på en sida

✓ Vill du presentera text på en sida har du tillgång till kontrollerna...

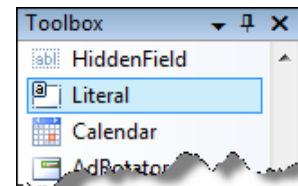
## ■ Label

- Då text i en sidan är dynamisk och du behöver ändra på den kan du använda dig av en **Label**-kontroll.
- Du kan tilldela egenskapen **Text** enkel text eller HTML.
- Som standard renderas en **Label**-kontroll som ett **span**-element.
- Tilldelar du egenskapen **AssociatedControlID** ett värde (sätt detta värde för att peka ut en serverkontroll som representerar ett formulärfält) renderas **Label**-kontrollen som ett **label**-element.



## ■ Literal

- **Literal**-kontrollen påminner om **Label**-kontrollen fast med skillnaden att dess egenskap **Text** inte renderas i något element. Det "enda" som renderas är själva texten i sig, ingenting annat.
- Kontrollen saknar egenskaper som t.ex. **CssClass** (inget element → inga attribut!).
- Med hjälp av egenskapen **Mode** kan du bestämma om texten som renderas ut ska HTML-kodas.



# Vilka alternativ för inmatningsfält finns?

✓ Det finns flera kontroller för insamling av data. Några du kan använda är...

## ■ TextBox

- TextBox-kontrollen kan du använda till att rendera tre olika typer av textfält.
- Några användbara egenskaper är `Text`, `Enabled`, `ReadOnly`, `MaxLength`.

## ■ CheckBox

- En CheckBox-kontroll använder du då du är i behov av en kryssruta. ☐ Ja, jag godkänner villkoren.

Skicka

## ■ RadioButton

- Två eller flera RadioButton-kontroller använder du då du behöver alternativknappar. En alternativknapp förekommer alltid i en grupp. Endast en knapp kan vara vald i gruppen.
- Du använder egenskapen `GroupName` för att gruppera RadioButton-kontroller som ska höra ihop.


Vad tycker du om ASP.NET?

- ☒ Kanon, detta verkar vara vad jag alltid önskat mig.
- ☐ Det verkar lite rörigt, men det blir nog bra.
- ☐ PHP är det enda alternativet som räknas!

Skicka

# Hur kan jag posta ett formulär?

- ✓ Du kan skicka ett formulär (göra en "postback") genom att använda tre kontroller...

- Button -----> 

- Renderar ett input-element.

```
<asp:Button ID="MyButton" runat="server" Text="Skicka" />
```

- LinkButton -----> 

- "Knappen" renderas som ett a-element.

```
<asp:LinkButton ID="MyLinkButton" runat="server" Text="Kommentarer (2)" />
```

- Använder JavaScript för att skicka formuläret till servern!

```
<a id="LinkButton1" href="javascript:__doPostBack('LinkButton1','')">Kommentarer (2)</a>
```

- ImageButton -----> 

- Renderar ett input-element.

```
<asp:ImageButton ID="MyImageButton" runat="server" ImageUrl="App/Images/searchWebButton.gif" />
```

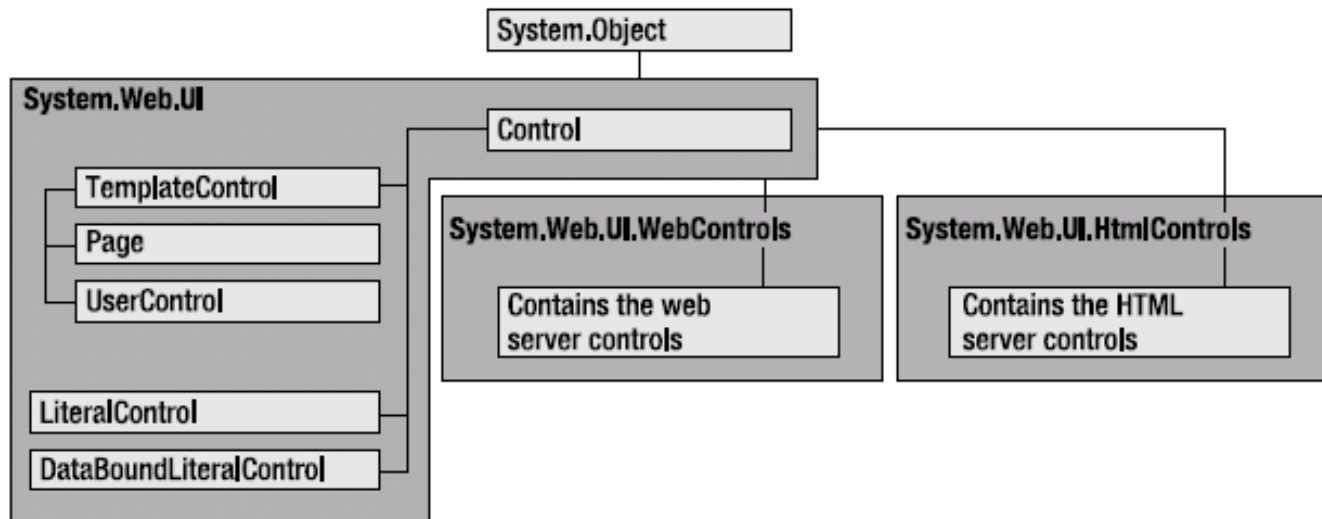


# Hur kan jag gruppera kontroller?

- ✓ Vill du arbeta med grupper av kontroller har du två möjligheter som kan vara användbara om t.ex. vill visa eller dölja ett antal kontroller.
  - **Panel**
    - Renderar i regel ett `div`-element (äldre webbläsare som inte vet vad ett `div`-element är får ett `table`-element istället.
    - Tilldelar du egenskapen `Grouping Text` ett värde renderas kontrollen som fieldset- och legend-element.
    - Kan agera som container, d.v.s. innehålla andra kontroller, HTML, etc.
    - Sätter du egenskapen `Visible` till `false` renderas varken kontrollen eller det den innehåller.
  - **Placeholder**
    - Ingen tagg renderas bara det kontrollen innehåller. (Jämför med `Label`- och `Literal`-kontrollerna.)
    - Kan agera som container, d.v.s. innehålla andra kontroller, HTML, etc.
    - Sätter du egenskapen `Visible` till `false` renderas inte kontrollens innehåll.

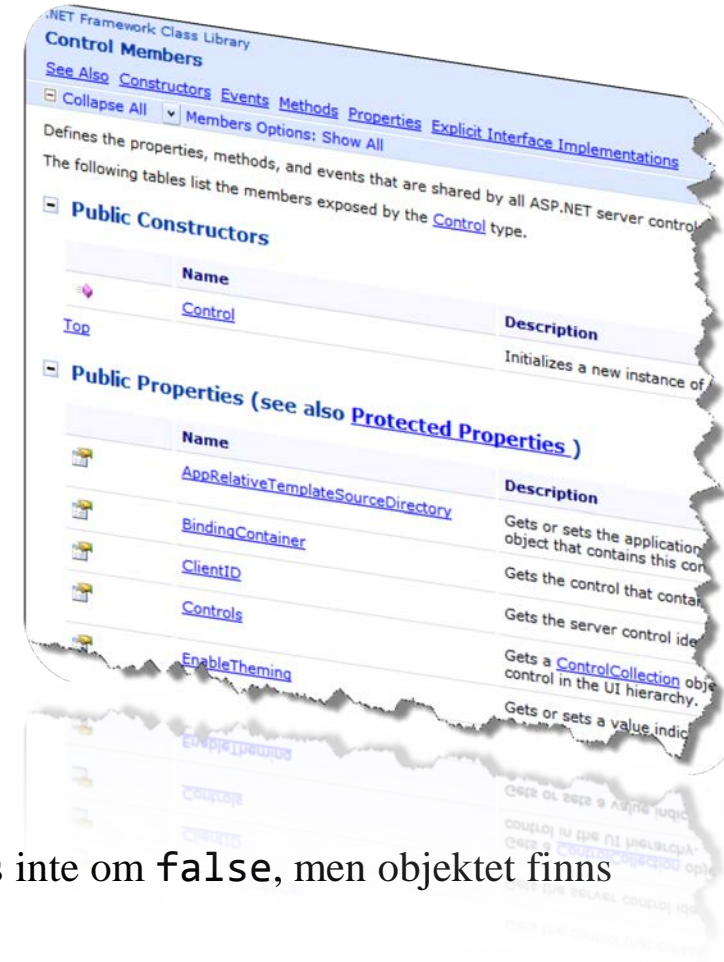
# Alla kontroller ärver från samma klass

- ✓ Alla serverkontroller, oavsett typ, ärver från basklassen `Control` som finns i namnområdet `System.Web.UI`.
- ✓ Det innebär att samtliga serverkontroller delar flera egenskaper, metoder och händelser.



# Egenskaper i klassen Control

- ✓ **ClientID**
  - Unikt ID som ASP.NET skapar vid initiering.
- ✓ **ID**
  - Det ID du skapar och använder dig av när du vill komma åt kontrollen från koden på servern.
- ✓ **Controls**
  - Ger en container med ev. "child controls".
- ✓ **ViewStateMode**
  - Ska kontrollen använda sig av "view state"?
- ✓ **Page**
  - Returnerar en referens till "parent page" som är ett objekt av typen Page.
- ✓ **Parent**
  - Returnerar en referens till "parent control".
- ✓ **Visible**
  - Ska kontrollen renderas eller inte? HTML genereras inte om false, men objektet finns på servern.



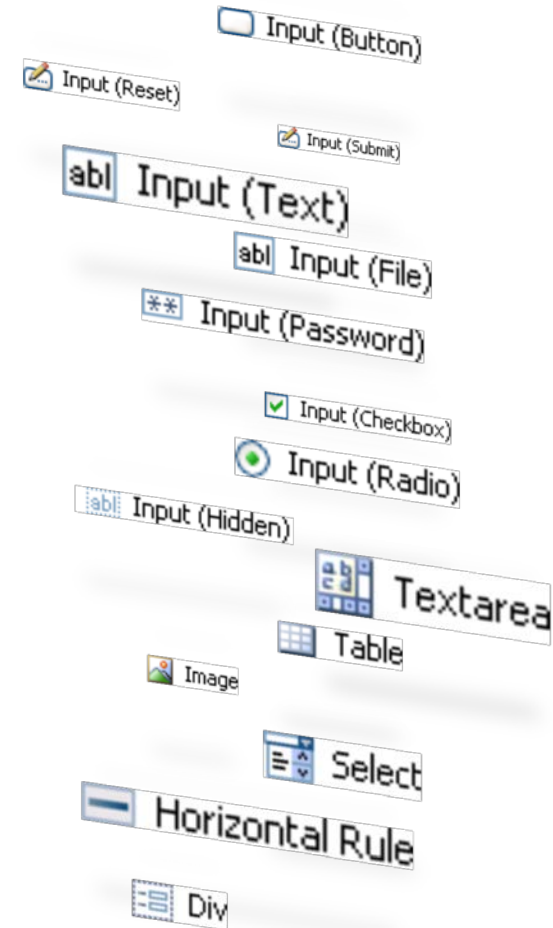
# Metoder i klassen Control

- ✓ FindControl()
  - Låter dig leta efter en "child control" med ett speciellt namn (ID).
- ✓ HasControls()
  - Kontrollerar om kontrollen har "childs".
- ✓ Render()
  - Skriver HTML-koden för kontrollens nuvarande status. Används av ASP.NET när kontrollen renderas ut till HTML m.m.
- ✓ DataBind()
  - Binder kontrollen och dess "childs" till en datakälla.

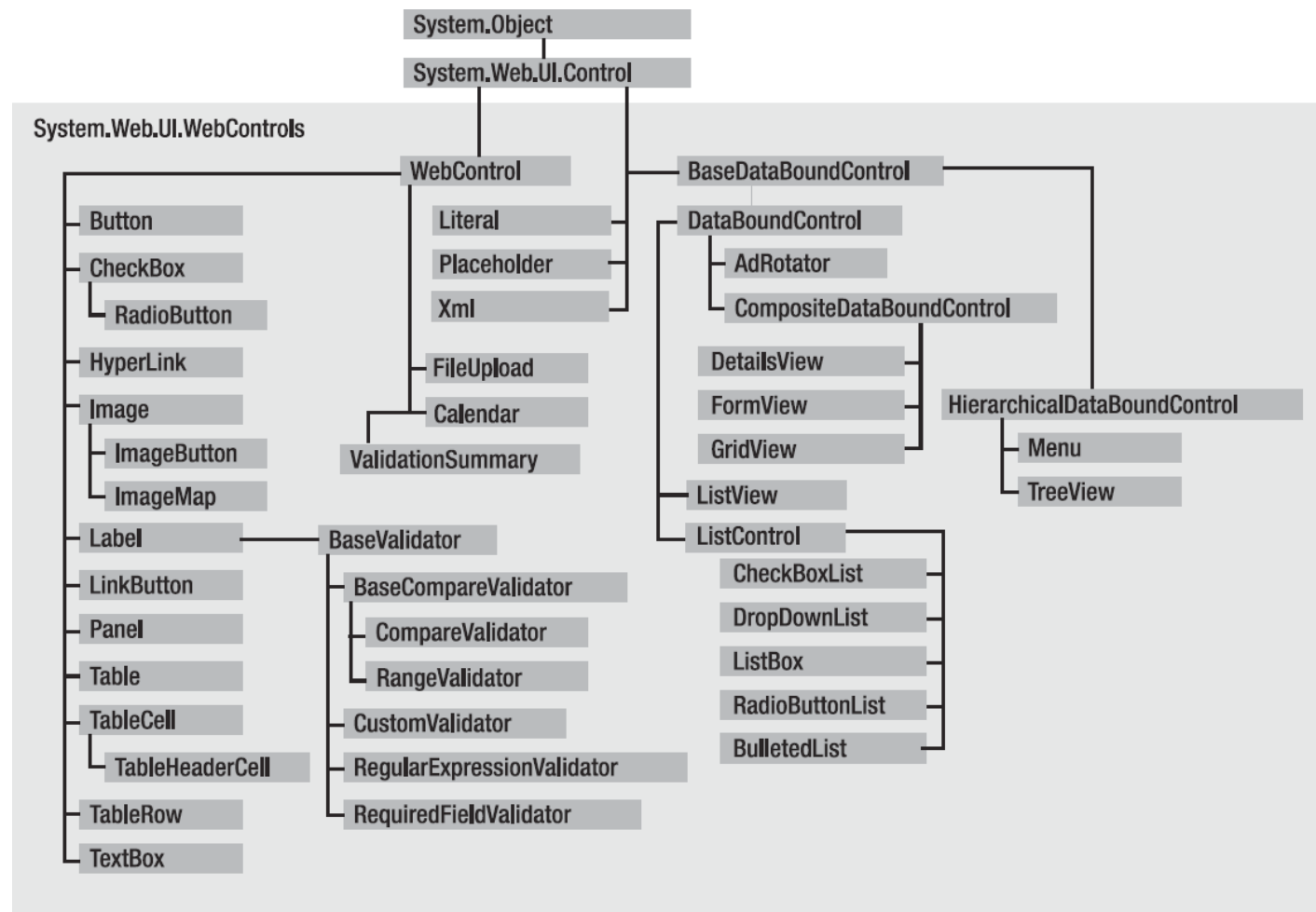


# HTML Server Controls

- ✓ Det finns ett 20-tal "HTML Server Controls".
- ✓ "HTML Server Controls" används i undantagsfall.  
Vanligare att använda kontroller i `System.Web.UI.WebControls`.
- ✓ "HTML Server Controls" har ett antal nackdelar jämfört med kontroller i `System.Web.UI.WebControls`.  
Webbkontroller...
  - ...har smartare namn och egenskaper.
  - ...är bättre på hantering av händelser.
  - ...kan rendera ut komplex HTML-struktur och JavaScript.
  - (...är inte låsta vid HTML-specifikationen på samma sätt.)



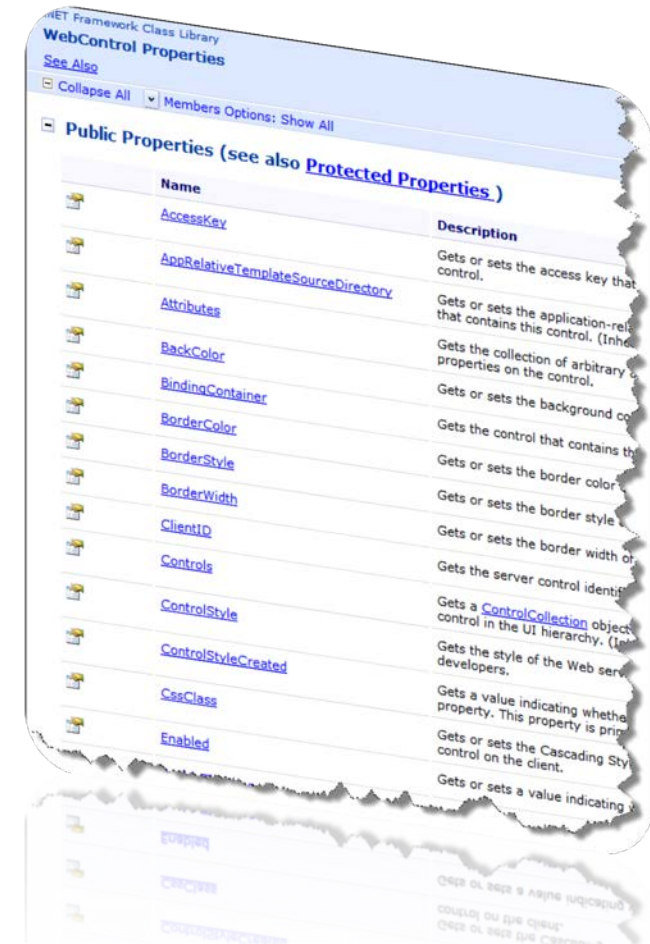
# Web Controls



# Basklassen WebControl

- ✓ Alla webbkontroller ärver från klassen `WebControl` som bl.a. har följande egenskaper:

- `CssClass`
  - Kan koppla en CSS-klass till kontrollen.
- `Enabled`
  - Ska användaren kunna använda kontrollen?
- `AccessKey`
  - Definierar "shortcuts".
- `TabIndex`
  - Definierar tabbordningen.
- `ToolTip`, `Width`, `Height`, `BackColor`, `BorderColor`, `BorderStyle`, `BorderWidth`, `ForeColor`



# Focus()

- ✓ ”Alla” webbkontroller kan anropa metoden Focus() för att styra att kontrollen ska ha fokus i formuläret på klienten. *OBS! Fungerar endast med kontroller som renderas som input-element.*
- ✓ På klienten sker en automatisk bläddring, om så behövs, till den kontroll som får fokus.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/1999/xhtml">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Untitled Page</title>
</head>
<body>
<form id="form1" runat="server" defaultfocus="TextBox1">
<div>
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

```

.aspx  
(deklarativt)

```
TextBox1.Focus();
```

.cs  
(programatiskt)



# Default Button

- ✓ En knapp som är definierad som "default button" är den knapp som "klickas" då användaren trycker ner Enter-tangenten.
- ✓ Fungerar med de webbkontroller som implementerar interfacet **IButtonControl**, t.ex. **Button**, **LinkButton**, **ImageButton**.
- ✓ Fungerar inte med "HTML Server Controls".

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server" defaultbutton="Button1">
        <div>
            <asp:Button ID="Button1" runat="server" Text="Button" />
        </div>
    </form>
</body>
</html>
```

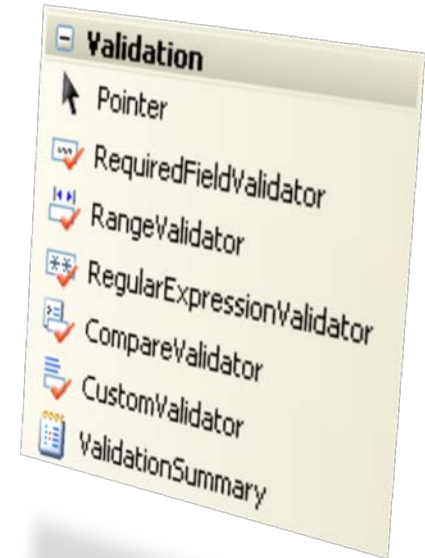
# Validering

- ✓ Alla webbapplikationer måste validera data innan det hanteras på servern.
- ✓ Validering ska alltid ske både på klient- och serversidan.
- ✓ Validering på klientsidan ger användaren snabb återkoppling om något är fel (och sparar resurser på serversidan).
- ✓ Validering på serversidan skyddar mot felaktigheter i datat som skickas och hackningsförsök (det går ju att smita förbi klientskript).
- ✓ Det finns ett antal valideringskontroller som underlättar valideringen för dig som programmerare.



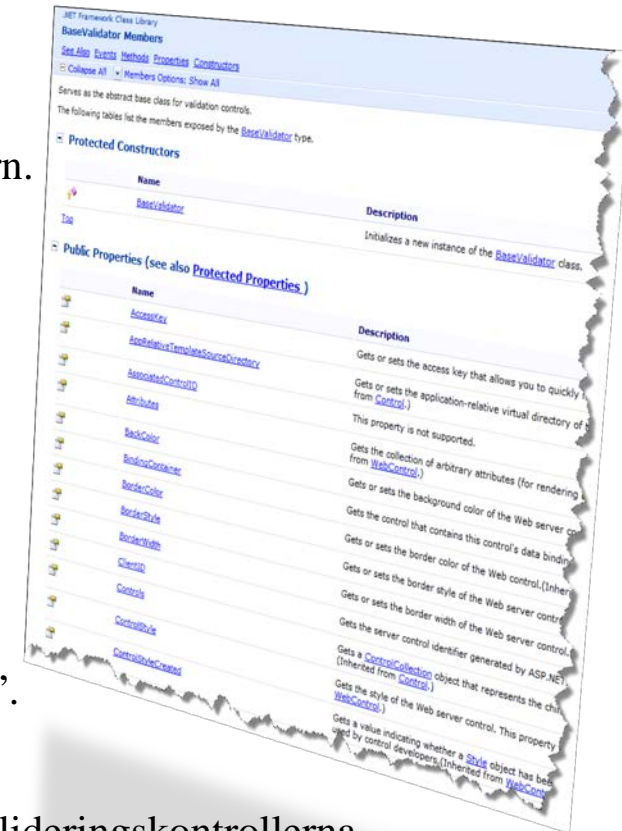
# Valideringskontroller

- ✓ **RequiredFieldValidator**
  - Kontrollerar så att den associerade kontrollen innehåller information.
- ✓ **RangeValidator**
  - Kan kontrollera att ett värde håller sig inom vissa värden.
- ✓ **RegularExpressionValidator**
  - Kontrollerar en kontrollis värde gentemot ett reguljärt uttryck.
- ✓ **CompareValidator**
  - Kör tester på värden (datatyp, större än, lika med, mindre än...).
- ✓ **CustomValidator**
  - Låter dig som utvecklare skriva både klient- och serverkod.
- ✓ **ValidationSummary**
  - Visar en sammanställning av de fel som fångats av valideringskontrollerna.



# Medlemmar i klassen BaseValidator

- ✓ **ControlToValidate**
  - Talar om vilken kontroll (oftast någon `TextBox`) som ska valideras.
- ✓ **EnableClientScript**
  - Sätter du denna egenskap till `false` genereras inget skript som exekveras på klient. Validering utförs endast på servern.
- ✓ **ErrorMessage**
  - Definierar vilket meddelande som ska visas för användaren av `ValidationSummary`-kontrollen.
- ✓ **Text**
  - Felmeddelande som visas vid kontrollen.
- ✓ **ValidationGroup**
  - Genom att grupper valideringskontrollerna kan man styra vilka som ska användas i samband med en "postback".
- ✓ **ValidationSummary**
  - Visar en summering av de eventuella fel som fångats av valideringskontrollerna.



# Så använder du valideringskontroller

- ✓ Valideringskontroller måste kopplas till en inmatningskontroll genom att tilldela attributet `AssociatedControlID` inmatningskontrollens ID.
- ✓ För att undersöka om ett textfält är tomt eller inte måste `RequiredFieldValidator` användas.
- ✓ Validering sker på klienten om JavaScript är tillåtet. Alltid på servern.

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server" CssClass="validation-summary-errors icon-error"
    HeaderText="Fel inträffade. Korrigera och försök igen." />
<div class="editor-label">
    <asp:Label ID="Label1" runat="server" AssociatedControlID="EmailTextBox" Text="E-post:" />
</div>
<div class="editor-field">
    <asp:TextBox ID="EmailTextBox" runat="server" />
    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="EmailTextBox"
        CssClass="field-validation-error" Display="Dynamic" ErrorMessage="E-postadress måste anges."
        SetFocusOnError="True" Text="*" />
    <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" ControlToValidate="EmailTextBox"
        CssClass="field-validation-error" Display="Dynamic" ErrorMessage="E-postadressen verkar inte vara korrekt."
        SetFocusOnError="True" Text="*" ValidationExpression="^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$" />
</div>
```



Fel inträffade:

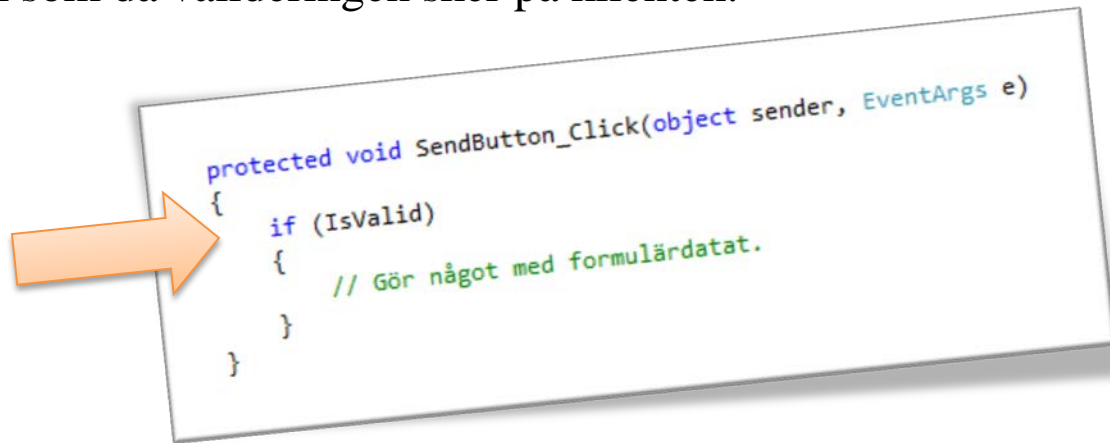
- E-postadress måste anges.

E-post:

Skicka

# Kontrollera alltid resultatet av valideringen på server

- ✓ Du kan inte lita på att validering av formulärdatat skett på klienten.
- ✓ Hanterarmetoder på servern exekveras som vanligt även då `IsValid` har värdet `false`.
- ✓ Innan behandling av formulärdatat på servern kontrollera alltid att egenskapen `IsValid` är `true`!
- ✓ Då resultatet av valideringen är `false` visas exakt samma felmeddelande på klienten som då valideringen sker på klienten.



```
protected void SendButton_Click(object sender, EventArgs e)
{
    if (IsValid)
    {
        // Gör något med formulärdatat.
    }
}
```

# Utökad användning av ValidationSummary

- ✓ Ibland kan det vara lämpligt att använda **ValidationSummary** för felmeddelanden som nödvändigtvis inte har med valideringen att göra, t.ex. om uppdatering av en post i en databastabell misslyckats.
- ✓ Ett felmeddelande kan läggas till programmatiskt i "code-behind"-filen genom att skapa ett **CustomValidator**-objekt som initieras och läggs till samlingen **Validators**.

```
var validator = new CustomValidator
{
    IsValid = false,
    ErrorMessage = "Ett fel inträffade då e-postadressen skulle uppdateras."
};
this.Page.Validators.Add(validator);
```



Fel inträffade. Korrigera fel.

- Ett fel inträffade då e-postadressen skulle uppdateras.

E-post:

ellen.nu@lnu.se

Skicka

# Mer om validering

- ✓ Som standard sker alltid validering i samband med att formulärdatat postas till servern.
- ✓ Genom att en Button-kontrolls egenskapen `CausesValidation` till `false` sker ingen validering vare sig på klienten eller servern.
- ✓ På servern kan validering alltid ske genom anrop av metoden `Validate()`. Ett anrop av `Validate()` måste ovillkorligen ske innan egenskapen `IsValid` kan anropas.
- ✓ Serverkontroller kan organiseras i valideringsgrupper genom att använda attributet `ValidationGroup` vilket kan vara användbart då vissa inmatningselement/kontroller ska valideras då användaren klickar på en knapp, och andra inmatningselement/kontroller ska valideras om användaren klickar på en annan kommandoknapp.

