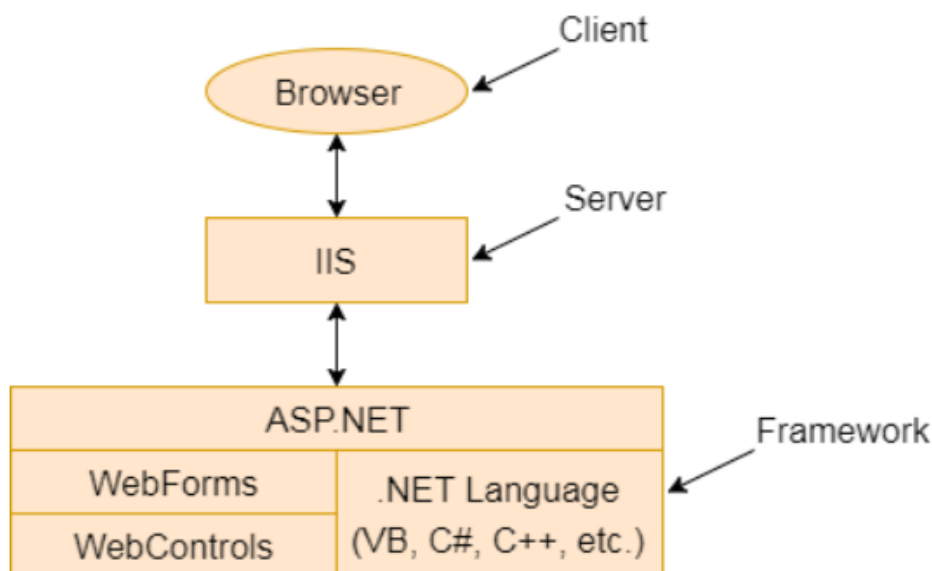


ASP.NET Web Forms. Componente ASP.NET. Controale ASP.NET (ASP.NET controls). Validari. Master Page

ASP.NET Web Forms

- **Web Forms** sunt pagini Web construite cu ajutorul tehnologiei ASP.NET
- Se executa pe server si genereaza output in browser
- Sunt compatibile cu orice browser si limbaj care suporta .NET
- In Visual Studio putem crea pagini de tip Web Form. Acest IDE ne permite, de asemenea, sa utilizam controale .NET prin **drag and drop** si in acelasi timp putem seta **proprietati**, **evenimente** si **metode** pentru acestea.

Componente ASP.NET



Controale ASP.NET

ASP.NET pune la dispozitie controale usor de utilizat, care sunt folosite la crearea componentelor HTML, deoarece fiecare control .NET genereaza cod HTML catre browser.

❖ **TextBox** – este utilizat pentru a prelua inputul scris de utilizator

Controlul gasit in ASP:

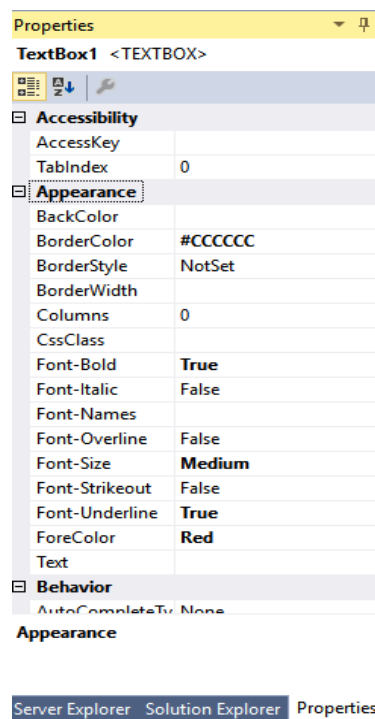
```
<asp:TextBoxID="TextBox1" runat="server" ></asp:TextBox>
```

Se genereaza urmatorul cod de HTML catre browser:

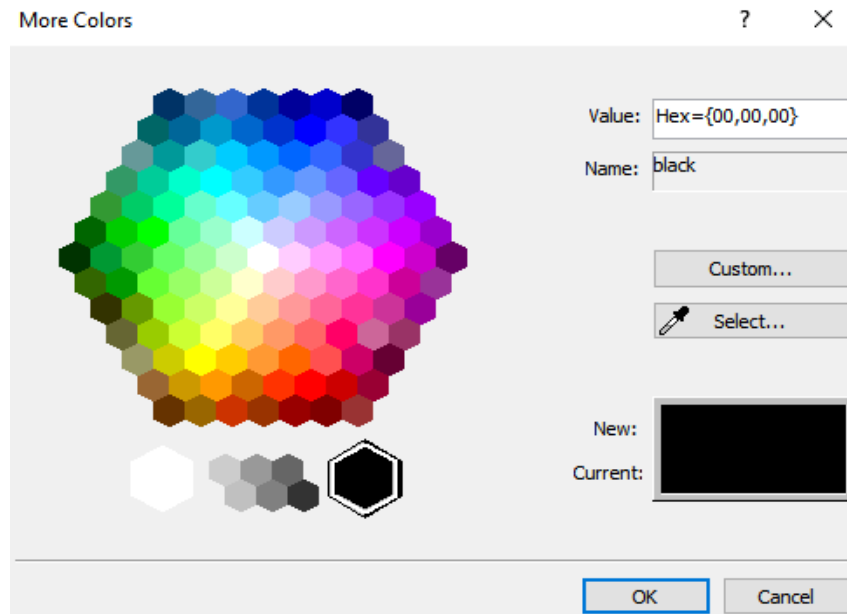
```
<input name="TextBox1" id="TextBox1" type="text">
```

Pentru includerea stilizarilor se pot folosi mai multe variante:

➤ Din **Properties – Appearance** se pot adauga elemente specifice stilizarii, dupa cum putem observa in imaginea urmatoare



Se pot selecta culori din culorile existente:

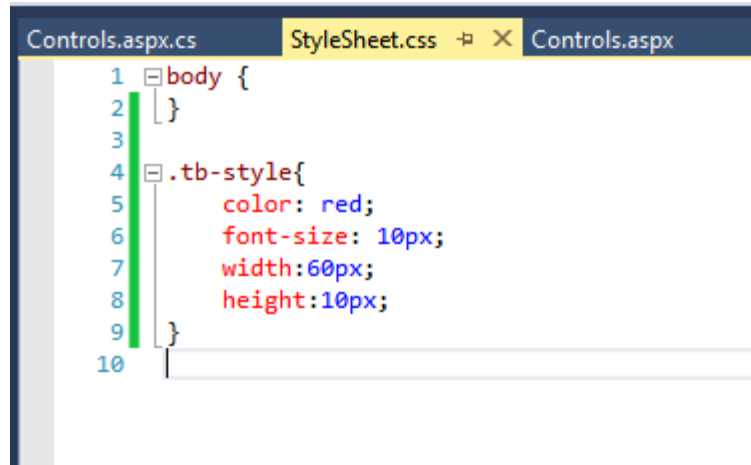


Controlul de tip TextBox-ul va arata in felul urmator:

```
<asp:TextBox ID="TextBox1" runat="server" ToolTip="Nume si  
Prenume" Font-Bold="True" Font-Size="Medium" Font-  
Underline="True" ForeColor="Red" BorderColor="#CCCCCC">  
</asp:TextBox>
```

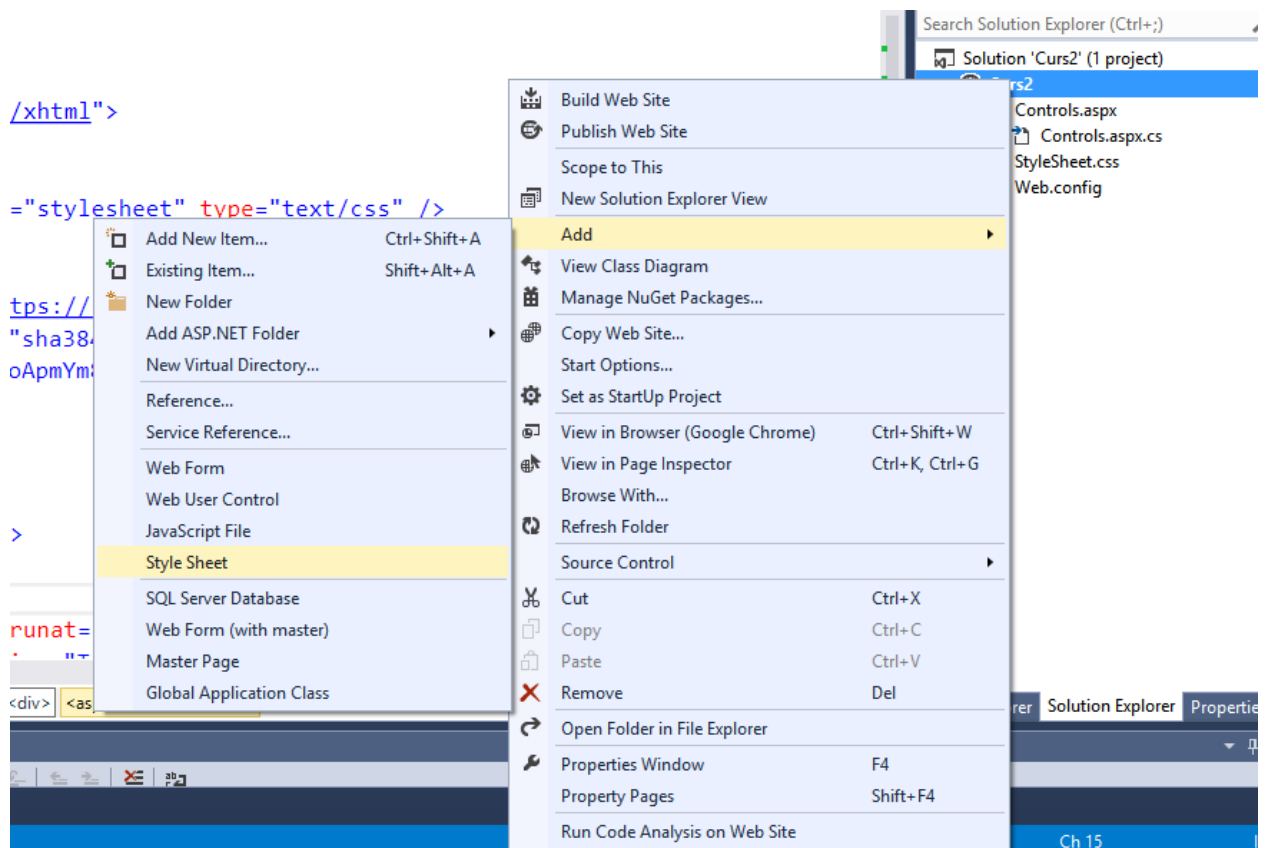


- Prin adaugarea unui fisier de tip text/css, in care vor fi scrise toate stilizarile necesare, dupa cum se observa in imaginea urmatoare:



```
1 body {  
2 }  
3  
4 .tb-style {  
5     color: red;  
6     font-size: 10px;  
7     width: 60px;  
8     height: 10px;  
9 }  
10
```

Cum se adauga fisierul **StyleSheet.css**:



Cum se include fisierul **StyleSheet.css** intr-o pagina aspx:

```
<head runat="server">  
    <link href="StyleSheet.css" rel="stylesheet" type="text/css"/>  
</head>
```

➤ De asemenea, se poate folosi **Bootstrap**

```
<head runat="server">  
  
    <link rel="stylesheet"  
        href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/boot  
strap.min.css" integrity="sha384-  
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPM0"  
        crossorigin="anonymous">  
  
</head>
```

❖ **Button** – este folosit pentru a efectua evenimente si pentru a trimite cererea clientului catre server

➤ **Controlul gasit in ASP**

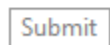
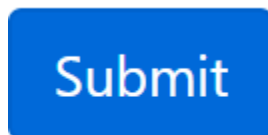
```
< asp:ButtonID="Button1" runat="server" Text="Submit" BorderStyle="Solid" ToolTip="Submit"/>
```

➤ **Se genereaza urmatorul cod HTML catre browser**

```
<input name="Button1" value="Submit" id="Button1" title="Submit" style="border-style:Solid;" type="submit">
```

- Clasele din Bootstrap se folosesc utilizand proprietatea **CssClass** a butonului:

```
<asp:Button ID="Button1" runat="server" Text="Submit"
BorderStyle="Solid" ToolTip="Submit" OnClick="ClickBttn"
CssClass="btn btn-primary"/>
```



- ❖ **Label** – este utilizat, in general, pentru a afisa informatii. In comparatie cu literalul acesta are mai multe proprietati, printre care **AssociatedControlID**. Aceasta proprietate face ca in momentul in care un utilizator face click pe elementul de tip **Label**, controlul asociat acestuia (de obicei este un **TextBox**) este focusat in mod automat.

Nume

- ❖ **Literal** – este utilizat pentru afisarea textului in mod dinamic in pagina

```
<asp:Label ID="LabelNume" runat="server" Text="Nume"
AssociatedControlID="TextBoxNume"></asp:Label>
```

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

```
<br />
```

```
<asp:Button ID="Button1" runat="server" Text="Submit"
BorderStyle="Solid" ToolTip="Submit" OnClick="ClickBttn" CssClass="btn
btn-primary"/>
```

```
<br />
```

```
<asp:Literal ID="LiteralAfisareNume" runat="server"></asp:Literal>
```

Pentru partea de CS:

```
protected void ClickBttn(object sender, EventArgs e)
{
    LiteralAfisareNume.Text = TextBoxNume.Text;
}
```

Nume

Popescu

❖ HyperLink

```
<asp:HyperLink ID="HyperLink1" runat="server"
NavigateUrl="www.google.com"> Google </asp:HyperLink>
```

- ❖ **RadioButton** – permite utilizatorului ca aleaga o optiune dintr-un grup de optiuni
- ❖ **CheckBox** – permite utilizatorului sa selecteze mai multe optiuni din setul de optiuni existent

Google

☒ Male ☐ Female

Your gender is Male

☒

❖ Calendar

```
<asp:Calendar ID="Calendar1" runat="server" SelectedDate="08-10-2018"></asp:Calendar>
```

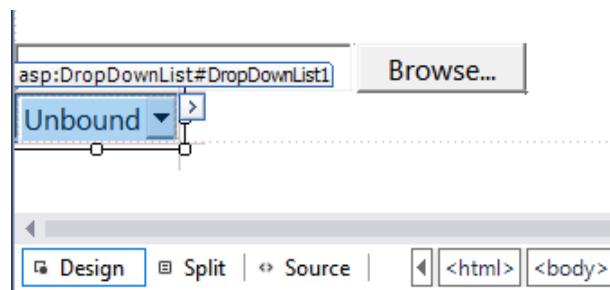
< October 2018 >						
Mon	Tue	Wed	Thu	Fri	Sat	Sun
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

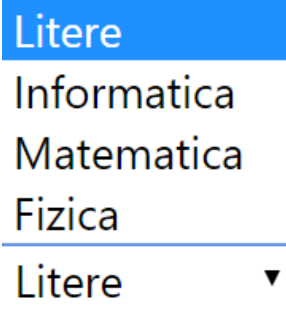
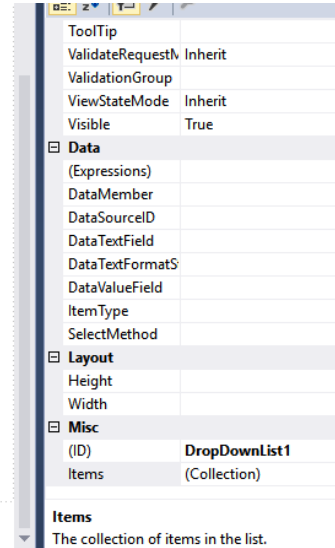
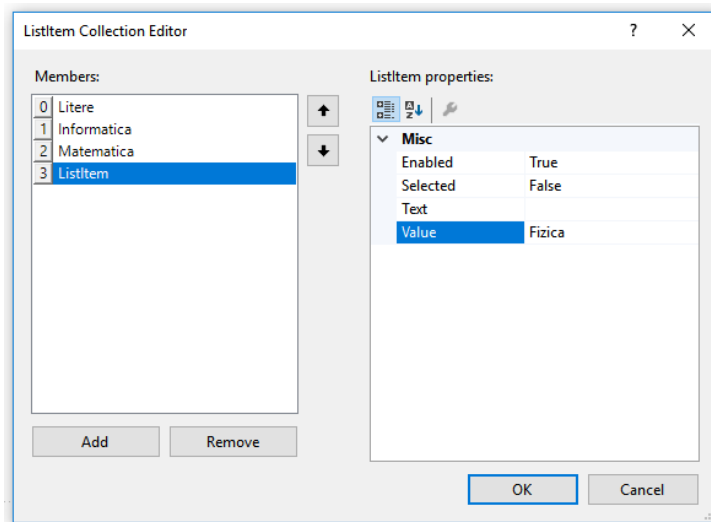
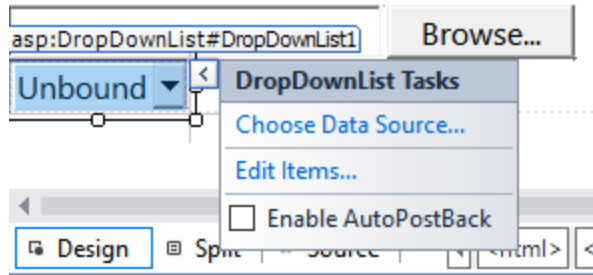
❖ FileUpload

```
<asp:FileUpload ID="FileUpload1" runat="server" />
```

❖ DropDownList

```
<asp:DropDownList ID="DropDownList1" runat="server">  
    <asp:ListItem Value="Litere"></asp:ListItem>  
    <asp:ListItem Value="Informatica"></asp:ListItem>  
    <asp:ListItem Value="Matematica"></asp:ListItem>  
    <asp:ListItem Value="Fizica"></asp:ListItem>  
</asp:DropDownList>
```





- ❖ **DataList** – este folosit pentru a afisa datele dintr-o sursa de date sub forma de lista. Sursa de date poate fi un tabel (DataTable) sau chiar un tabel din baza de date

.aspx

```
<asp:DataList ID="DataList1" runat="server">
    <ItemTemplate>

        <tr id="Tr1" runat="server">
            <td style="color: #ac00dc"><%#Eval("ID") %></td>
            <td style="color: #09c"><%#Eval("Nume") %></td>
            <td style="color: #ccc"><%#Eval("Email") %></td>
        </tr>

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

.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    DataTable tableDataList = new DataTable();
    tableDataList.Columns.Add("ID");
    tableDataList.Columns.Add("Nume");
    tableDataList.Columns.Add("Email");
    tableDataList.Rows.Add("1", "User1", "user1@example.com");
    tableDataList.Rows.Add("2", "User2", "user2@example.com");
    tableDataList.Rows.Add("3", "User3", "user3@example.com");
    tableDataList.Rows.Add("4", "User4", "user4@example.com");

    DataList1.DataSource = tableDataList;
    DataList1.DataBind();
}
```

```
1 User1 user1@example.com
2 User2 user2@example.com
3 User3 user3@example.com
4 User4 user4@example.com
```

❖ **DataGrid** – este folosit pentru afisarea datelor dintr-o sursa de date

.aspx

```
<asp:GridView ID="GridView1" runat="server"></asp:GridView>
```

.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    DataTable tableGridView = new DataTable();
    tableGridView.Columns.Add("ID");
    tableGridView.Columns.Add("Nume");
    tableGridView.Columns.Add("Email");
    tableGridView.Rows.Add("1", "User1", "user1@example.com");
    tableGridView.Rows.Add("2", "User2", "user2@example.com");
    tableGridView.Rows.Add("3", "User3", "user3@example.com");
    tableGridView.Rows.Add("4", "User4", "user4@example.com");

    GridView1.DataSource = tableGridView;
    GridView1.DataBind();
}
```

ID	Nume	Email
1	User1	user1@example.com
2	User2	user2@example.com
3	User3	user3@example.com
4	User4	user4@example.com

Validari ASP.NET



- Pentru a efectua validarea ASP.NET ofera controale care verifica input-ul utilizatorilor
- Atunci cand o aplicatie colecteaza date de la utilizator, trebuie sa ne asiguram ca datele colectate de la acesta sunt valide. Unii utilizatori nu sunt interesati sa-si petreaca suficient timp pentru a introduce informatiile corecte intr-un formular si in unele cazuri, utilizatorii ar putea introduce in mod intentionat informatii false pentru a obtine acces
- Unul din primii pasi este de a intelege **ce inseamna validarea datelor**. Validarea, in acest caz, nu inseamna ca daca Popescu Andrei isi scrie numele in campul din formular ca Ionescu Andrei, se trimite o alerta pentru a informa ca datele sunt false. Nu, inca nu avem capacitatea de a afla daca o afirmatie este adevarata
- Validarea ne ajuta sa determinam daca utilizatorul a introdus ceva in camp, dupa care se poate verifica daca a introdus un numar sau un caracter, daca acesta este in formatul corect, etc. Se pot compara datele introduse de utilizatori din diferite campuri sau fata de valorile care ar putea fi pastrate de exemplu intr-o baza de date

Diferenta intre validarile Server-Side si validarile Client-Side

In validarea **server-side**, informatiile sunt trimise catre server si validate. In cazul in care validarea nu reuseste, raspunsul este trimis inapoi clientului, pagina care contine formularul web este actualizata si un feedback este afisat. Aceasta metoda este sigura deoarece va functiona chiar daca JavaScript este oprit in browser si nu poate fi usor de ocolit de utilizatori rau intentionati. Pe de alta parte, in acest caz utilizatorii trebuie sa completeze informatiile fara a primi un raspuns deoarece o sa primeasca raspunsul doar dupa trimiterea formularului. Acest lucru duce la un raspuns lent de la server.

Validarea server-side este suficienta pentru a avea o validare sigura si cu succes a formularului. Cu toate acestea, pentru o mai buna experienta a utilizatorilor, se recomanda utilizarea validarii **client-side**. Acest tip de validare se face pe client, folosind de exemplu JavaScript. Astfel, input-ul utilizatorilor poate fi validat pe masura ce acestia tasteaza.

In cazul validarii **client-side**, formularul nu se transmite niciodata daca validarea nu reuseste, utilizatorii primind feedback imediat.

Principalul dezavantaj al validarii client-side este ca se bazeaza pe JavaScript. Daca utilizatorii dezactiveaza JavaScript-ul, pot trece cu usurinta validarea. De aceea, validarea ar trebui intotdeauna implementata atat pe client, cat si pe server. Prin combinarea metodelor bazate pe server si client, putem obtine cele mai bune rezultate: raspuns rapid, validare mai sigura si o mai buna experienta a utilizatorilor.

Validările in ASP.NET

ASP.NET ofera posibilitatea de a utiliza controalele serverului de validare. Ceea ce face ca aceste controale ale serverului de validare sa fie eficiente este ca atunci cand este ceruta o pagina ASP.NET care contine aceste controale, **ASP.NET decide daca va efectua validarea pe client sau pe server**, in functie de browserul care realizeaza cerere. Prin urmare, functionalitatea paginii se modifica in functie de browserul solicitant.

Implementarea validărilor pentru următorul exemplu:

Nume *

Email *

Parola

Confirmare

Data nasterii

Varsta

- Completati numele
- Completati adresa de email

Nume

Email *

Parola

Confirmare

Data nasterii

Varsta

- Adresa de email nu este valida

Nume

Email

Parola

Confirmare *

Data nasterii

Varsta

- Reconfirmarea parolei nu este corecta

Nume

Email

Parola

Confirmare

Data nasterii *

Varsta *

- Completati data in format LL/ZZ/AAAA
- Dati varsta in intervalul 0-70

Nume

Email

Parola

Confirmare

Data nasterii

Varsta

Va multumim pentru completarea datelor!

Prezentare validatori

- **RequiredFieldValidator** – este folosit pentru a specifica faptul ca un anumit camp este obligatoriu
- **CompareValidator** - este folosit pentru a compara valoarea dintr-un input cu valoarea din alt input sau pentru a compara tipul de date
- **RangeValidator** – verifica daca o valoare dintr-un anumit input face parte dintr-un interval
- **RegularExpressionValidator** – verifica valoarea dintr-un input si determina daca aceasta se potriveste cu un pattern (model) definit de o expresie regulata
- **ValidationSummary** – afiseaza o lista pe pagina cu toate mesajele de eroare
- **CustomValidator** – este un validator custom, adica poate fi definit astfel incat sa respecte o anumita functionalitate implementata de noi

Validations.aspx

```
<body>
  <form id="form1" runat="server">

    <div>

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

      <asp:TextBox ID="TextBoxNume" runat="server"></asp:TextBox>

      <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ControlToValidate="TextBoxNume" ErrorMessage="Completati
numele" ToolTip="Completati numele">*</asp:RequiredFieldValidator>

      <br /><br />
    </div>
  </form>
</body>
```



```

<asp:Label ID="Label10" runat="server" Text="Email"></asp:Label>

<asp:TextBox ID="TextBoxMail" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator8"
Display="Dynamic" runat="server" ControlToValidate="TextBoxMail"
ErrorMessage="Completati adresa de email">*</asp:RequiredFieldValidator>

<asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server" ErrorMessage="Adresa de email nu este valida"
ValidationExpression="\w+([-+.']\w+)*@\w+([-.\w+)*\.\w+([-.\w+)*"
ControlToValidate="TextBoxMail"
Display="Dynamic">*</asp:RegularExpressionValidator>

<br /><br />

<asp:Label ID="Label8" runat="server" Text="Parola"></asp:Label>

<asp:TextBox ID="TextBoxParola" runat="server"
TextMode="Password"></asp:TextBox>

<br /><br />

<asp:Label ID="Label9" runat="server" Text="Confirmare"></asp:Label>

<asp:TextBox ID="TextBoxConfirmare" runat="server"
TextMode="Password" CausesValidation="True"></asp:TextBox>

<asp:CompareValidator ID="CompareValidator1" runat="server"
ControlToCompare="TextBoxConfirmare" Operator="Equal"
ControlToValidate="TextBoxParola" ErrorMessage="Reconfirmarea parolei nu este
corecta">*</asp:CompareValidator>
<br /><br />

<asp:Label ID="Label4" runat="server" Text="Data
nasterii"></asp:Label>

<asp:TextBox ID="TextBoxData" runat="server"></asp:TextBox>

<asp:CompareValidator ID="CompareValidator3" runat="server"
ControlToValidate="TextBoxData" Display="Dynamic" ErrorMessage="Completati
data in format LL/ZZ/AAAA" Operator="DataTypeCheck"
Type="Date">*</asp:CompareValidator>

<br /><br />

<asp:Label ID="Label6" runat="server" Text="Varsta"></asp:Label>

<asp:TextBox ID="TextBoxVarsta" runat="server" Width="33px">

```

```

        </asp:TextBox>

        <asp:RangeValidator ID="RangeValidator1" runat="server"
ControlToValidate="TextBoxVarsta" Display="Dynamic" ErrorMessage="Dati varsta
in intervalul 0-70" MaximumValue="70" MinimumValue="0"
Type="Integer">*</asp:RangeValidator>

        <br /><br />

        <asp:Button ID="Button1" runat="server" Text="Trimite"
OnClick="Button1_Click" />

        <asp:Literal ID="LiteralRaspuns" runat="server"
Visible="False"></asp:Literal>

        <asp:ValidationSummary ID="ValidationSummary1" runat="server" />

</div>
</form>
</body>

```

Validations.aspx.cs

```

protected void Button1_Click(object sender, EventArgs e)
{
    LiteralRaspuns.Visible = true;
    LiteralRaspuns.Text = "Va multumim pentru completarea datelor!";
}

```



Tips and Tricks

- Utilizati intotdeauna **Page.IsValid** inainte de a trimite date. In afara de celelalte beneficii, utilizarea acestuia impiedica trimiterea datelor din browserele vechi care nu accepta JavaScript
- Proprietatea de afisare (**display**) pentru controalele de validare ASP.NET are 3 setari: **None**, **Static** (implicit) si **Dynamic**. Daca proprietatea este **Static**, in HTML o sa fie automat **visibility: hidden** ceea ce inseamna ca daca exista mai mult de un control de validare plasat langa acel camp, primul control de validare ocupa spatiul de pe ecran chiar si atunci cand nu exista eroare. In cazul in care cel de-al doilea control de validare declanseaza un mesaj de eroare, mesajul va aparea la distanta de control, deoarece primul control de validare ocupa spatiul de pe ecran. Daca se seteaza proprietatea **Display** a unui control de validare la **Dynamic**, atunci in HTML o sa fie **visibility: none**, ceea ce face ca mesajul de eroare sa fie afisat exact langa control
- Pentru a impiedica ca validarea sa aiba loc la apasarea altui buton decat cel de Submit/Trimite se poate seta proprietatea **CausesValidation** la **False**

```
<asp:Button ID="btnCancel" Runat="server"  
CausesValidation="False" Text="Cancel" />
```

- Utilizati proprietatea **InitialValue** din **RequiredFieldValidator** pentru a valida controalele care au o valoare implicita, dupa cum urmeaza:

```
<asp:DropDownList ID="DropDownList1" runat="server">

    <asp:ListItem Value="--Select--" />
    <asp:ListItem Value="Item1" />
    <asp:ListItem Value="Item2" />
    <asp:ListItem Value="Item3" />

</asp:DropDownList>

<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ErrorMessage="RequiredFieldValidator"
ControlToValidate="DropDownList1"
InitialValue="--Select--"></asp:RequiredFieldValidator>
```

- Pentru a detecta daca un anumit camp este liber sau necesar se foloseste **RequiredFieldValidator**. Restul controalelor de validare nu ofera aceasta functionalitate de detectare daca un input este liber. De aceea, atunci cand este necesar se pot utiliza mai multe controale de validare in acelasi timp. De asemenea, **CustomValidator** face o exceptie de la regula deoarece are proprietatea **ValidateEmptyText** care in momentul in care este setata la valoarea **True** verifica daca acel camp este liber
- In caz de eroare, controalele de validare permit setarea focusului in mod automat in campul care a determinat eroarea

```
<asp:RequiredFieldValidator SetFocusOnError="true"
ControlToValidate="TextBox1"
ID="RequiredFieldValidator1" runat="server"
Text="Error!"></asp:RequiredFieldValidator>
```

- Este best practice ca **informatiile obligatorii** sa fie intotdeauna precedate de un asterisk (*) pentru a evidentia acest lucru

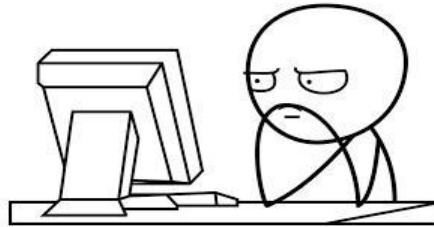
- Feedback-ul de validare trebuie implementat vizibil. Daca un camp are eroare este foarte bine ca acesta sa fie evidentiata cu un border rosu si o culoare de fundal rosie
- **/!/ Pentru toate erorile trebuie folosit font de culoare rosie. Erorile pot fi insotite de o iconita reprezentativa sub forma unui semn de exclamare**
- **Pentru mesajele de informare trebuie folosit un font de culoare albastra. Mesajele de informare pot fi insotite de o iconita reprezentativa sub forma unui (i)**
- **✓ Pentru mesajele de succes este necesara folosirea unui font verde. Acestea pot fi insotite de o iconita sub forma de bifa.**
- Pentru erorile de validare se va folosi aceeaasi pagina din care utilizatorul face un request. O pagina universala pentru toate erorile din aplicatie inseamna un user experience slab
- Validarea instantata (in client, prin intermediul JavaScript – atunci cand mesajele de eroare apar imediat dupa completarea unui camp) este foarte buna insa abuzul acesteia poate produce un user experience mai putin bun. Abuzul acestei functionalitati poate distrage utilizatorul de la scopul sau
- Indicatoarele de ajutor (**tooltips**) pot oferi utilizatorilor informatii despre cum ar trebui completate formularele. Astfel, experienta de utilizare poate fi imbunatatita in mod considerabil
- Implementarea Captcha ajuta la eliminarea spam-ului. Desi este un pas in plus in experienta utilizatorilor finali, acesta imbunatateste considerabil informatiile ajunse la server

£3.00

£3.00



! Value must be greater than or equal to 5.

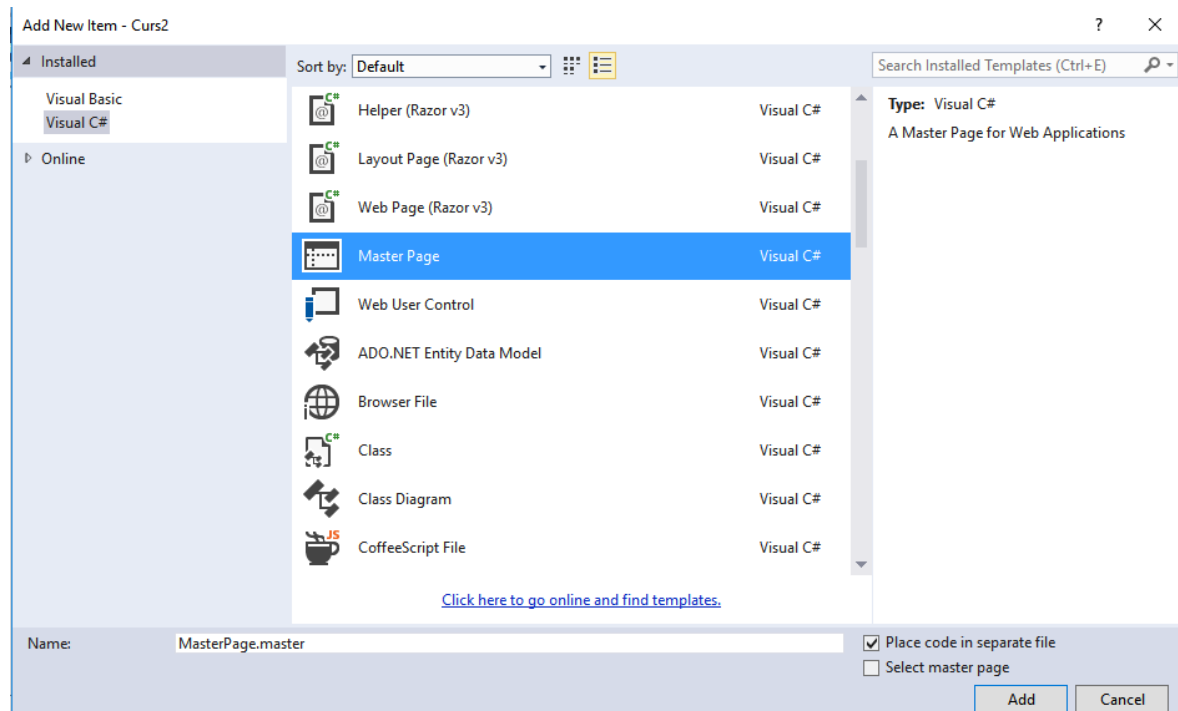


```
><td class="hide-on-small">...</td>
<td>
  <input type="number" name="updates[]" id=
    "updates_36217826625" class="quantity" value="1"
    min="1"> == $0
```

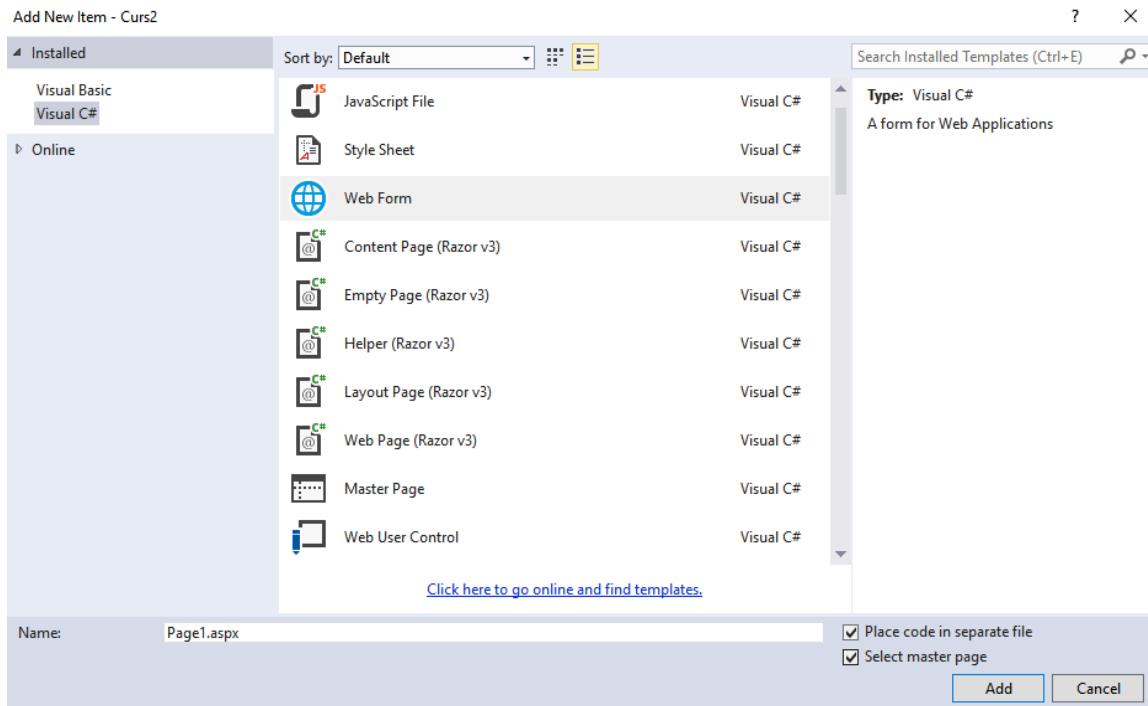
Master Pages

Sunt componente prin intermediul carora se pot construi layout-uri pentru paginile unei aplicatii web. Acestea inglobeaza elemente de interfata care vor fi disponibile in mod automat pe toate paginile existente.

Cum se creeaza o astfel de pagina:



Selectarea Master Page-ului pentru o pagina



MasterPage.master

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            <div style="width:auto; height: 100px; background-color:lightblue;
padding:10px">
                <h2>Restaurant</h2>

                <asp:HyperLink ID="HyperLink1" runat="server"
NavigateUrl="~/Page1.aspx">Acasa</asp:HyperLink>
```



```

        <asp:HyperLink ID="HyperLink3" runat="server"
NavigateUrl="~/Page2.aspx">Galerie foto</asp:HyperLink>

    </div>

    <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

        <--! Aici nu se pun tag-uri-->

    </asp:ContentPlaceHolder>

    <div style="width:auto; height: 100px; background-color:lightblue;
padding:10px">
        <div style="text-align:center">
            <p>Restaurant</p>
            <p>Adresa: Str. Academiei 14</p>
            <p>Telefon: 021 524-1234</p>
        </div>
    </div>
</div>
</form>
</body>
</html>

```

Page1.aspx

```

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true" CodeFile="Page1.aspx.cs" Inherits="Page1" %>

```

```

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">

```

Continut Page1

```

</asp:Content>

```

✚ **Pana saptamana viitoare profesorul de laborator trebuie sa primeasca lista cu temele pe care le-ati ales**