

**UNIVERSITETI I PRISHTINËS**

**FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE**



# **PUNIM DIPLOME**

## **ASP.NET KONTROLLAT**

Mentori  
Dr.Sc.Edmond BEQIRI

Kandidati  
Ardian IBRAJ

Prishtinë, Shtator 2004

# Përmbajtja

<b>Hyrje</b> -----	<b>4</b>
<b>HTML Server Kontrollat</b> -----	<b>5</b>
Koncepti i Server kontrollave-----	6
Rëndësia dhe kuptimi i HTML Server Kontrollave -----	7
Kontrolla HTMLAnchor-----	9
Kontrolla HTMLButton-----	11
Kontrolla HTMLForm -----	13
Kontrolla HTMLGeneric -----	14
Kontrolla HTMLSelect -----	14
Kontrolla HTMLTextArea -----	15
Kontrolla HTMLImage -----	16
Kontrolla HTMLInputButton -----	17
Kontrolla HTMLInputCheckBox-----	18
Kontrolla HTMLInputRadioButton-----	18
Kontrolla HTMLInputText-----	20
Kontrolla HTMLInputImage-----	20
Kontrolla HTMLInputFile-----	20
Kontrolla HTMLInputHidden -----	21
Kontrolla HTMLTable-----	22
Kontrolla HTMLTableRow -----	23
Kontrolla HTMLTableCell-----	23
<b>ASP Server Kontrollat</b> -----	<b>25</b>
Web kontrollat themelore (Basic Web Controls) -----	26
Label .....	26
TextBox .....	26
Buttons .....	26
CheckBox.....	27
ListControls .....	27
HyperLink .....	27
Image .....	28
Panel .....	28
RadioButton .....	28
Table .....	28
Xml .....	29
Përdorimi i ngjarjeve të kontrollave-----	29

<b>Kontrollat Validuese</b>	<b>32</b>
Kontrollat Validuese si Web Form Kontrolla	32
Kontrollat Validuese	32
Kontrolla RequiredFieldValidator	33
Kontrolla RegularExpressionValidator	34
Kontrolla CompareValidator	35
Kontrolla RangeValidator	35
Kontrolla CustomValidator	37
Kontrolla ValidationSummary	38
<b>Custom Kontrollat</b>	<b>39</b>
Custom Kontrolla ( e nënkuptuar ) e plotë	42
Atributet e Custom kontrollave	43
Metoda e Renderimit (ang. Render)	44
Krijimi i Custom kontrollave të derivuara	46
Krijimi i Custom Kontrollave të përbëra	49
Krijimi i kontrollës së përbërë Numrues_i_Librave	54
Interfejsi INamingContainer	55
Krijimi i kontrollës së përbërë Lista_e_Kërkesave_ të_Librave	56
Atributet ControlBuilder dhe ParseChildren	57
Renderimi	58
<b>Përmbyllje</b>	<b>61</b>
<b>Summary</b>	<b>62</b>
SHTOJCA A	63
SHTOJCA B	64
<b>Literatura:</b>	<b>66</b>

## Hyrje

ASP.NET-i (Active Server Pages), është teknologji për të zhvilluar web aplikacione dhe ofron zgjidhjen shumë të nevojshme për një problem shumë të vjetër-kontrollat e HTML formave. Në verzionet para .Net-it programerët e ASP-së duhej që të kalonin herë në HTML herë në ASP për të ofruar ndërveprim në mes web faqeve. Pra thënë shkurt ASP nuk ofronte shkallë të lartë të ndërveprimit. Me shfaqjen e ASP server kontrollave, e gjithë kjo ka ndërruar. Kështu ASP.NET bën të mundur që të kryhet validimi në kohën reale dhe paraqitja e mesazheve të gabimit nga faqja kur është tejkualuar ndonjë fushë e kërkuar. Gjithashtu ASP.NET bën të mundur zëvendësimin e të dhënave në faqe pa e detyruar shfrytëzuesin që ti qaset ndonjë faqe tjetër apo të ristartoj komplet faqen. Gjatë zhvillimit të web formave në ASP.NET, mund të përdoren këto lloje të kontrollave:

- **HTML Server kontrollat** – me të cilat manipulohet në anën e serverit. Para se të dërgohet forma në anën e klientit makina e ASP-së i konverton ato në HTML elementet përkatëse. Këto kontrollat janë të përfshira në nejmshpejsin *System.Web.UI.HtmlControls*.
- **ASP Server kontrollat (njihen edhe si Web Kontrollat apo ASP.NET Web Form kontrollat)** – Këto janë kontrollat të reja të zhvilluara nga Microsofti. Ato i kanë të përfshira një numër të madhë të veqorive si dhe një grup me vetitë standarde. Në HTML dhe .aspx, këto kontrollat referohen me prefiksin asp si p.sh. **asp:Label**, **asp:Button**, **asp:TextBox** etj. Përveq server kontrollave të tipit të formës siq janë: labelat, butonat, apo lista rënëse (ang. dropdown ) janë edhe një numër i kontrollave speciale si kontrollat *Calendar* dhe *AdRotator*. Makina ASP i konverton këto kontrollat në HTML kontrollat standarde para se të dërgohet faqja te klienti. Këto Web server kontrollat gjenden në nejmshpejsin *System.Web.UI.WebControls*
- **Kontrollat e Validimit (Validation Controls)** - ky grup i kontrollave ofron mundësinë e zhvillimit të shpejtë të aplikacioneve (Rapid Application Development RAD). Përdoren për të kontrolluar në mënyrë automatike të dhënat hyrëse të shfrytëzuesit. Këto kontrollat gjenden në nejmshpejsin *System.Web.UI.WebControls*
- **Custom kontrollat** – ASP.NET-i ofron mundësinë e zgjerimit të kontrollave ekzistuese për të shtuar funksionalitete sipas nevojës. Ekzistojnë dy versione të Custom kontrollave: **Web User kontrollat** dhe **Web Custom kontrollat**. Dallimi kyç ndërmjet tyre qëndron në atë se **Custom kontrollat** kompajlohen në fajllat .dll dhe pastaj përdoren njëllë siç përdoren të gjitha server kontrollat. **Web User kontrollat** janë të lehta për tu zhvilluar, dhe ruhen si fajllat me papashtesën .ascx. Këto ruhen të kompajluara.

## Pjesa e parë

### HTML Server Kontrollat

Web Aplikacionet janë aplikacione që mund të zhvillohen në dy pjesë: pjesa e klientit dhe pjesa e serverit. Pjesa e klientit në përgjithësi ka të bëjë me pjesën statike të programimit, kurse këtu do të trajtohet pjesa e serverit e cila edhe mund të quhet pjesë dinamike. I gjithë koncepti i kësaj pjese sqarohet thjeshtë me atë se një HTML dokument përdorë protokollin HTTP (HyperText Transfer Protocol) për t'iu qasur një Web Serveri. Është e qartë se kur futet një URL (Universal Resource Locator) për një faqe në “browser”, ai do të dërgohet si një mesazh në HTTP deri te serveri (kërkesë për faqen e dëshiruar). Ky mesazh zakonisht njihet si “Request” (kërkesë). Nëse faqja që kërkohet ka p.sh prapashtesën .htm ose .html, Web Serveri thjeshtë gjenë faqen në disk dhe nga disku i tij ai e dërgon prapa deri te kompjuteri juaj (“klienti”) nëpërmjet një mesazhi të ri në HTTP, i njohur si “Response”(përgjigje). “Browser-i” është ai i cili interpreton këtë kod në objektin e përgjigjur dhe e prezanton atë në ekran. Kjo është mënyra se si realizohet kërkesa e klientit deri te marrja e faqes së dëshiruar nga serveri.

Një ASP.NET fajll, që ka prapashtesën .aspx, zakonisht përmban elemente HTML, me kod të anës së serverit dhe të klientit.

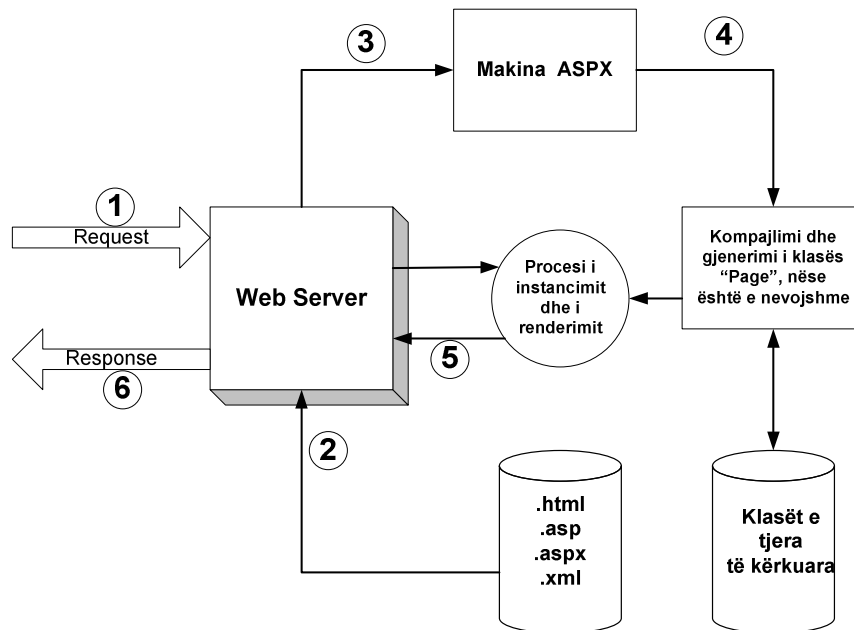


Figura 1. Hapat kryesor të përpunimit të kërkesës "Request" deri tek përgjigja "Response"

Siç shihet në figurën e mëposhtme, kur shfrytëzuesi kërkon ("Request") një faqe ASPX, serveri e merr atë nga disku dhe e dërgon tek makina ASPX për përpunim të mëtutjeshëm. Pastaj makina ASPX e kompajlon kodin e serverit dhe gjeneron fajllin e

klasës “page“, e instancon fajllin e klasës dhe i ekzekuton instruksionet që e zhvillojnë objektin “response“. Gjatë ekzekutimit sistemi i përpunon të dhënat e futura nga përdoruesi bazuar në instruksionet e programit (në pjesën e serverit). Në fund, serveri e transmeton objektin “Response” deri tek klienti.

Pra hapat kryesor të përpunimit të kërkesës deri tek përgjigja “Response” janë këto:

1. Serveri pranon kërkesën për faqen ASPX të dëshiruar.
2. Serveri gjen lokacionin e faqes në disk.
3. Serveri i dorëzon faqen makinës ASP.NET
4. ASP.NET-i e kompajlon faqen dhe gjeneron klasën “page“. Gjatë kompajlimit, mund të kërkoj edhe pjesë tjera të kodit siq janë kodi prapa komponenteve të ndryshme të përfshira në program
5. ASP.NET-i inicializon klasën, kryen përpunimin e domosdoshëm dhe gjeneron objektin “Response”
6. Pastaj Web serveri dërgon objektin “Response“ tek klienti.

## Koncepti i Server kontrollave

Para se të fillojmë të flasim në lidhje me HTML server kontrollat, të kuptojmë në përgjithësi konceptin themelor se çka janë server kontrollat në përgjithësi.

Server kontrollat janë tagje speciale të cilat janë të përpunuara nga Web Serverat në një mënyrë paksa të ngjajshme me mënyrën e tagjeve të HTML-së të interpretuara nga “browseri” juaj.

Tagjet e server kontrollave mund të identifikohen nga fakti se ato fillojnë me atributin **runat=”server”**. Kjo i ndihmon serverit t’i dalloj ato nga tagjet standarde të HTML-së.

Mjaft interesant është fakti se, në pjesën e klientit nuk ekziston atributi **runat=”client”**, dhe nëse kërkohet (“Request”) faqja me atributin “runat” të futur në pjesën e klientit (ose diçka të ngjajshme) do të paraqitet gabim me sqarimin se “Atributi **Runat** duhet të ketë vlerën **Server**” (ang. “The **Runat** attribute must have the value **Server**”).

Nganjëherë serveri gjenë kontrollat e serverit, ai krijon një objekt në memorje, të cilat prezantojnë server kontrollat. Këto objekte mund të kenë karakteristikat e tyre, metodat dhe mandej tregojnë aktivizimin e ngjarjeve të serverit gjatë përpunimit të faqes në ASP.NET. Dikur kur përpunimi mbaron, kontrollat shfaqin daljet e tyre në formë të HTML-së (ose shpeshherë ato dizajnohen për tu shfaqur) dhe dërgojnë të “browseri” pjesën e rezultatit të faqes.

## Rëndësia dhe kuptimi i HTML Server Kontrollave

Një nga shumë detyrat e zakonshme të zhvilluesit të Web-it do të jetë zhvillimi dhe koleksionimi i informatave të cilat postohen apo thirren nga shfrytëzuesi. Kjo thjeshtë mund të shpjegohet kur ne marrim emrin, e-mailin nga shfrytëzuesi ose përfshirja e disa informacioneve si adresa, numri i telefonit, fjalëkalimi (password), faksi, karta e kreditit etj. Sidoqoftë informacionet që dëshironi t'i mbledhni, përpunimi i tyre nuk mund të kryhet brenda kufijve të HTML-së vetëm në "browser". Prandaj, duhet t'i postojmë (dërgojmë) informacionet deri te Web Serveri për përpunim. Së pari Web Serveri do të pranojë informacionin e kërkuar, do t'i kryej ato punë të nevojshme dhe do t'ia kthej shfrytëzuesit.

Informacionet transmetohen përmes Web faqes nëpërmjet formave dhe në HTML. Në këto informacione ka tagje speciale për specifikimin e sjelljeve të tyre. HTML format përbajnë një grup me HTML kontrolla, siç janë *textbox*-et, *checkbox*-et etj., të gjitha këto ndihmojnë për bartjen e informacionit nga shfrytëzuesi deri te serveri. Për këto qëllime, ASP.NET ka futur kontrollat speciale për mënyrën e sjelljes me format.

HTML elementet brenda një fajlli ASP.NET janë trajtuar si tekst literal dhe janë programatikisht të paarritshme të zhvilluesi i faqeve. Që të jenë HTML elementet të arritshme duhet trajtuar ato si server kontrolla, me futjen e atributit **runat="server"**. Atributi unik "id" ju lejon t'i referoheni në mënyrë të programueshme kontrollave. Atributet janë përdorur për të deklaruar karakteristikat e argumenteve dhe ngjarjet që lidhin një instancë të server kontrollave.

Ajo që është shumë me rëndësi, që duhet ta kemi parasysh është se HTML server kontrollat duhet të vendosen në mes tagjeve **<form>** dhe **</form>** duke e përfshirë atributin **runat="server"**. Në ASP.NET duhet që çdo HTML server kontrollë të mbyllet me shenjën "/" ose me tagun mbyllës të plotë. Nëse HTML elementet nuk janë të mbyllura si duhet ASP.NET nuk e miraton atë element me çka elementi do të injorohet ose do të shfaqet një gabim i kompajlimit, varësisht nga ajo se si është krijuar.

Një instancë të ndonjë kontrole të formave mund të krijohet në tri rrugë të veçanta:

- Si një element në HTML dokument;
- Në mjedisin RAD (Rapid Application Development) i cili lejon që të merren kontrollat dhe të barten në faqe ( me *drag-and-drop*) dhe
- Në mënyrë të programueshme në kod duke deklaruar bllokun ose kodin prapa fajllit.

Këto kontrolla janë të përfshira në *namespace*-in **System.Web.UI.HtmlControls**. Duhet cekur se çdo klasë është një nënklasë e trashëguar nga një klasë më e lartë. Klasa **HtmlControls** është nënklasë e klasës **UI** e kjo është nënklasë e klasës **Web**, e gjithë kjo është një hierarki deri te klasa themelore (bazike). Çdo herë *namespace*-i është deri te pika e fundit, ku fillon klasa e fundit. Në rastin tonë *namespace*-i është **System.Web.UI**. *Namespace*-at mund të kuptohen si një kontejner i klasëve.

Një hierarki e klasëve të HTML kontrollave duket si në vijim:

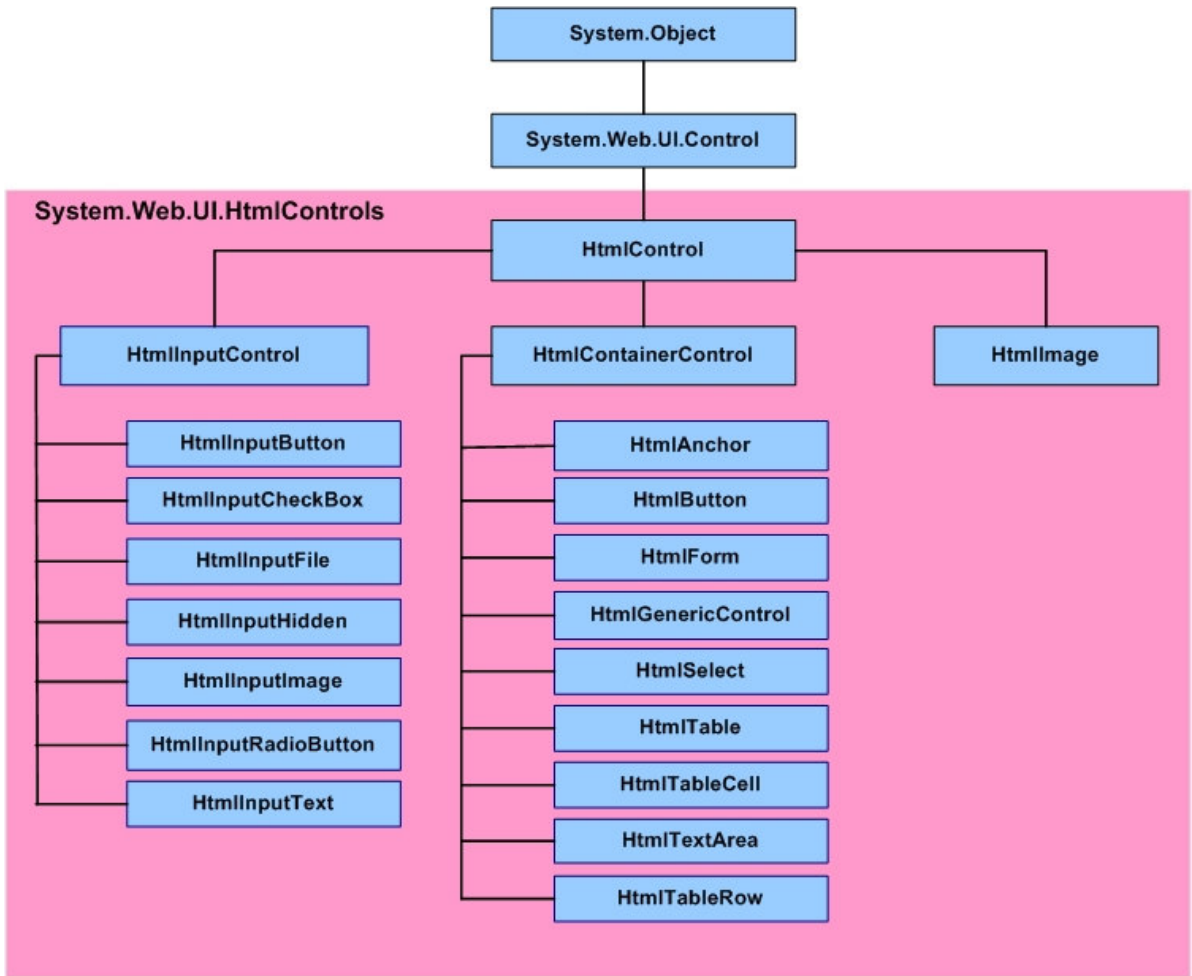


Figura 2. Hierarkia e HTML Server Kontrollave

Kontrolla	HTML Tagu
HtmlAnchor	<a>
HtmlButton	<button>
HtmlSelect	<select>
HtmlTextArea	<textarea>
HtmlInputButton	<input type="button">
HtmlInputCheckBox	<input type="checkbox">
HtmlInputRadioButton	<input type="radio">
HtmlInputText	<input type="text"> dhe <input type="password">
HtmlInputHidden	<input type="hidden">
HtmlInputImage	<input type="image">
HtmlInputFile	<input type="file">
HtmlForm	<form>
HtmlImage	<img>
HtmlTable	<table>
HtmlTableRow	<tr>
HtmlTableCell	<td>
HtmlGenericControl	Kontrollat tjera si p.sh <div>, <span> etj.

Figura 3. Tipi i kontrollave me HTML tagjet korresponduese

**Vërejtje:** Cekim edhe njëherë se brenda çdo tagu të kontrollave duhet të shënohet atributi **runat="server"**.

Në vazhdim do të definojmë secilën kontrollë me radhë duke u munduar ti shpjegojmë edhe përmes një shembulli të thjeshtë çdo kontrollë.

## Kontrolla HTMLAnchor

Kontrolla HTMLAnchor përdoret për të kontrolluar elementin **<a>**. Në HTML, elementi **<a>** përdoret për të krijuar një hiperlidhje. Hiperlidhjet shërbejnë për tu lidhur me ndonjë lokacion në kuadër të Web faqes tuaj ose ndonjë Web faqe krejtësisht tjetër. Kontrolla HTMLAnchor duhet të jetë mirë e formuar me hapjen e tagut **<a>** dhe mbylljen e tagut **</a>**.

Kjo kontrollë e serverit e zakonisht është shfrytëzuar për të modifikuar dinamikisht atributet dhe karakteristikat e elementit **<a>**, dhe ngjarjet e kontrollës për të gjeneruar kontrollën HTMLAnchor dinamikisht.

### Shembulli 1:

Në këtë shembull kemi deklaruar dy kontrollat HTMLAnchor në një fajll .asp, ku njëra prej lidhjeve na lidhë me lokacionin në kuadër të Webit tonë i cili lokacion ndodhet në disk, ndërsa tjetra na lidhë me një Web faqe tjetër (në rastin tonë me [www.hotmail.com](http://www.hotmail.com)), e gjithë kjo është kryer duke e ekzekutuar “subrutinën” në Web server. Funkzioni thirret përmes vlerës unike “**Id**” e cila është e vendosur në elementin **<a>**. Kodi për këtë shembull do të jetë:

```
<script runat="server">      <!--program i skriptuar si pjesë e serverit-->
Sub Page_Load                <!--sub rutina që do të ekzekutohet në Web
server-->
    linku1.HRef="http://www.hotmail.com"  <!--lidhja e parë me faqen e
hotmailit-->
    linku1.Target="_blank"
    linku1.Title="hotmail"      <!--titulli i linkut të parë "hotmail"-->
    linku2.HRef="E:\Ardi.htm" <!--lidhja e dytë me një faqe në disk-->
    linku2.Target="_blank"
    linku2.Title="faqjaime"
End Sub                        <!--fondi i subrutinës -->
</script>
<html>
<body>
    <form runat="server">      <!--fillimi i formës duke specifikuar se është
pjesë e serverit-->
    <a id="linku1" runat="server">Visitoni faqen www.hotmail.com!</a> <!--
deklarimi i elementit <a> duke i ndarë vlerën unike në vetinë "id" me
qëllim të thirrjes së funksionit për ekzekutim që ka emër të njejtë-->
    <br/>
    <a id="linku2" runat="server">Visitoni faqen kryesore!</a> <!--
elementit <a> i ndajmë në vetinë "id" vlerën unike për ta thirrur
funksionin me emrin linku2 dhe për ta ekzekutuar atë-->
    </form>                    <!--mbyllja e formës-->
</body>
</html>
```

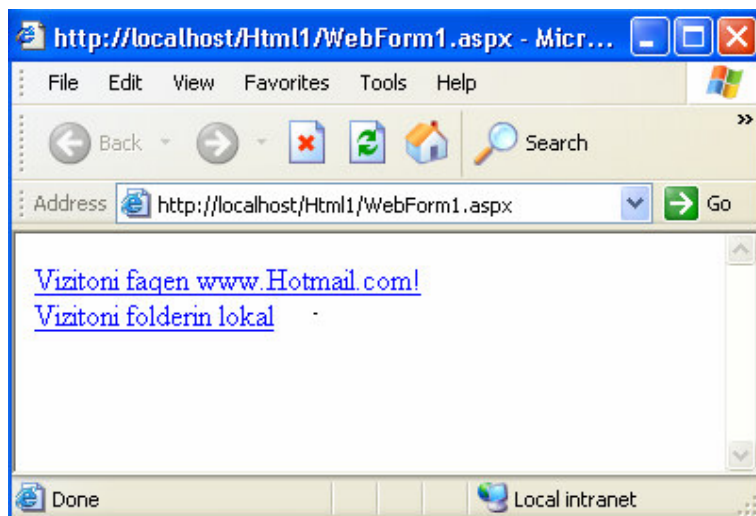


Figura 4. Pamja pas ekzekutimit, që përmban kontrollën Anchor

Pas ekzekutimit të programit fitohet rezultati si në fig 4. Nëse shtypet linku i parë do të hapet faqja e hotmailit, përderisa linku i dytë e hap një faqe në kuadër të Webit tonë, që gjindet në diskun e kompjuterit tonë.

## Kontrolla HTMLButton

Kontrolla HTMLButton përdoret për të kontrolluar elementin **<button>**. Me përdorimin e kësaj kontrole mund të krijohet një buton shtypës për të aktivizuar një ngjarje me “**serverclick**”, për të specifikuar aksionin kur kontrolla (butoni ) klikohet.

Në vazhdim tregohen disa rrugë të mundshme për të modifikuar pamjen e kontrollës HTMLButton.

- Mund të fusim atributin “**style**” në tagun e hapur të elementit, për ta përcaktuar gjatësinë, gjerësinë, ngjyrën e butonit etj.
- Mund të përfshijmë edhe elemente formatuese përreth tekstit të cilin e fusim brenda tagjeve (**<button>** teksti**</button>**) të kontrollës.
- Mund të përfshijmë imazhe ( “**image**” ) brenda elementit të butonit dhe
- Mund të fusim një ngjarje në kontrollën HTMLButton

### Shembulli 2:

Në vazhdim po e paraqesim një shembull të thjeshtë me dy butone, ku me rastin e shtypjes së butonit të parë fitojmë mesazhin “**Ju klikuat ne butonin me tekst!**” ndërsa kur shtypim butonin e dytë fitojmë mesazhin “**Ju klikuat ne butonin me fotografi!**”. Kodi për këtë shembull do të jetë:

```
<script runat="server">
Sub butoni1(Source As Object, e As EventArgs) <!--subrutina për butonin e parë-->
->
    p1.InnerHtml="Ju klikuat në butonin me tekst!" End Sub
Sub butoni2(Source As Object, e As EventArgs) <!--subrutina për butonin e dytë-->
->
    p1.InnerHtml="Ju klikuat në butonin me fotografi!" End Sub
</script>
<html>
<body>
    <form runat="server">
        <button id="b1" OnServerClick="butoni1" style="background-color:
#e6e6fa; height=25;width:100" runat="server"> <!--kontrolla "button" i
deklaruar për ta thirrë me një klik mbi të subrutinën me emrin "butoni1", ndërsa
vetia "id" është unike dhe në këtë rast i kemi caktuar vlerën "b1"-->
            Butoni tekstual!
        </button>
        <button id="b2" OnServerClick="butoni2" style="background-color:
#fff0f5; height=25;width:100" runat="server">  <!--në vend të
tekstit këtu mbi buton do të paraqitet një imazh-->
        </button>
        <p id="p1" runat="server" /><!--elementi përmes së cilës subrutina e
shkruan mesazhin-->
    </form>
</body>
</html>
```

pas shtypjes së butonit të parë rezultati do të fitohet si në figuren 5, ndërsa me shtypjen e butonit të dytë rezultati do të fitohet si në figuren 6.

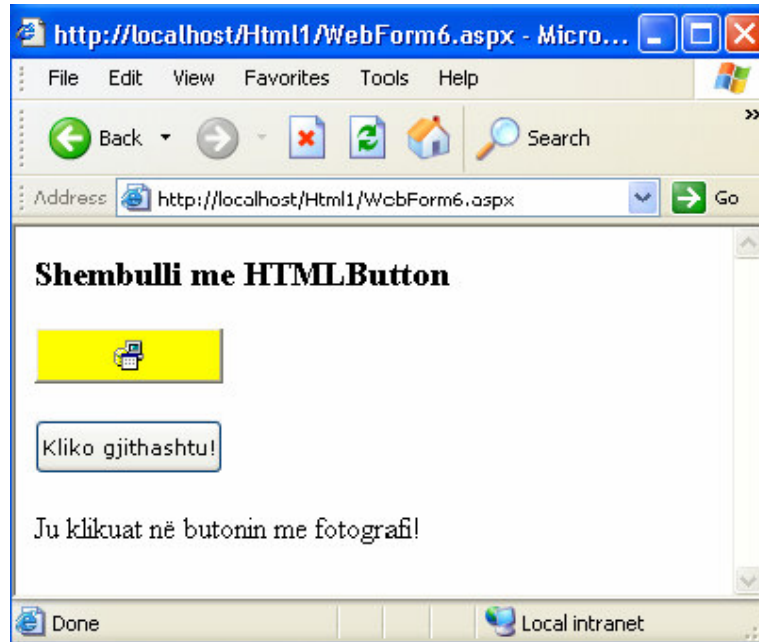


Figura 5. Pas shtypjes së butonit të parë

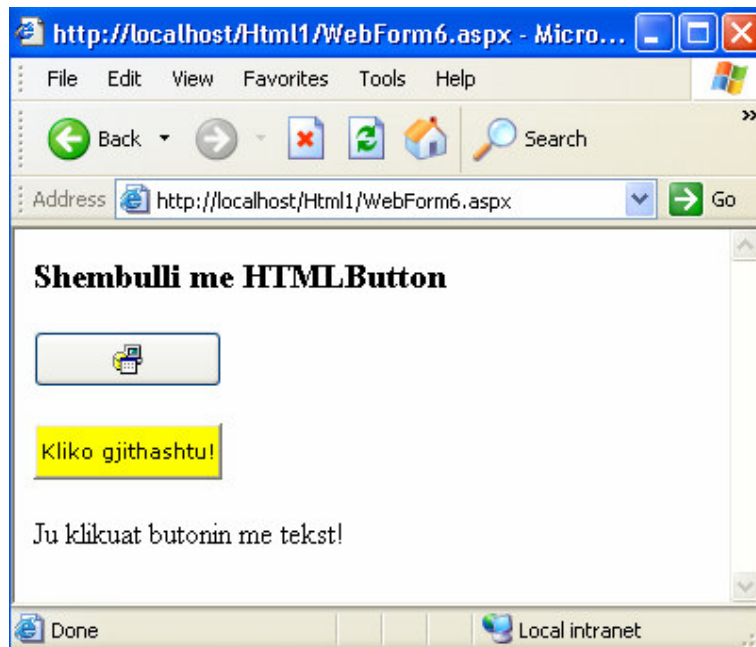


Figura 6. Pas klikimit mbi butonin e dytë

## Kontrolla HTMLForm

Kontrolla HTMLForm përdoret për të kontrolluar elementin **<form>**. Në HTML, elementi **<form>** përdoret për të krijuar një formë. Kontrolla HTMLForm përdoret për paraqitjen e të dhënave të një entiteti, ku mundësohet paraqitja, vendosja, thirrja dhe ndryshimi i të dhënave. Kjo kontrollë ka dy attribute: *Method* dhe *Action*.

Atributit *Method* i bashkangjesim dy vetitë **POST** dhe **GET**. Përmes vetisë **POST** ne mund të dërgojmë parametra në bazë të të dhënave ndërsa përmes vetisë **GET** kërkojmë ndonjë prej parametrave të vendosur në bazë të të dhënave. Të cekim se ASP.NET si të nënkuptueshme e ka vetinë **POST**.

Atributi *Action* specifikon faqen që do të thirret për dërgimin e të dhënave. Ky atribut nuk mund të ndryshohet, si pasojë, ne mund të thërrasim vetëm faqen tonë. Ky atribut paraqet një **URL** që definon ku të dërgohen të dhënat kur forma “submitohet” (“submit”).

### Shembull 3:

Në këtë shembull kemi përdorë një `HTMLInputText` kontroll, një `HTMLInputButton` kontroll, dhe një `HTMLGeneric` kontroll në fajll të `.aspx`. Të gjitha kontrollat janë të futura brenda kontrollës `HTMLForm`. Kur butoni “submit” shtypet, do të ekzekutohet subrutina. Submit subrutina e shkruan mesazhin për mirëseardhje dhe paraqet vlerën e shënuar në tekst boks përmes elementit `p`. Kodi për këtë shembull do të jetë:

```
<script runat="server">
Sub submitimi(sender As Object, e as EventArgs)
    If emri.value<>"" then
        p1.InnerHtml="Mirëseardhët " & name.value & "!" <!--mesazhi që do të
shfaqet pas shtypjes së butonit si dhe vlera që shënohet në tekst boks-->
    End if
End Sub
</script>
<html>
<body>
    <form id="Form1" action=WebForm1.aspx runat="server">
        Shënoni emrin tuaj:
        <input id="emri" type="text" size="30" runat="server"/>
<!--kontrolla input me tip tekstual për futjen e tekstit nga shfrytëzuesi-->
        <br /><br />
<input type="submit" value="Submit" OnServerClick="submitimi"
runat="server"/> <!--Kontrolla input me tipin submitues, si dhe me ngjarjen
"onserverclick" për ta lidhur me subrutinen "submitimi". Vlera që do të paraqitet
mbi këtë buton do të ketë vlerën "submit"-->
<p id="p1" runat="server"/> <!--elementi përmes së cilës subrutina e shkruan
mesazhin -->
    </form>
</body>
</html>
```

Pas ekzekutimit të programit fitohet rezultati si në figurën 7.

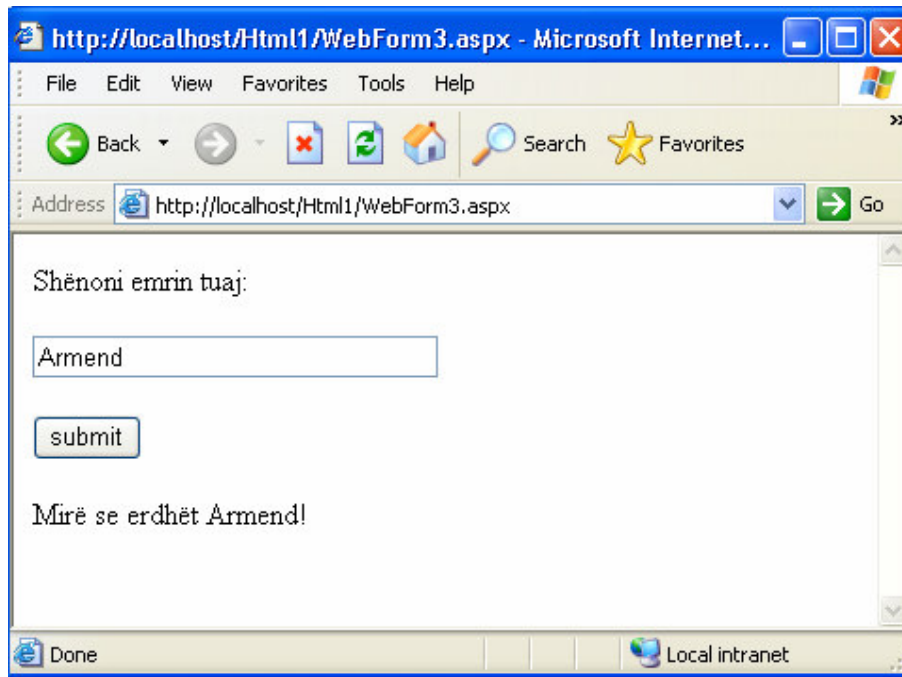


Figura 7. Pas shkruarjes së emrit dhe shtypjes së butonit

## Kontrolla HTMLGeneric

Kontrolla HTMLGeneric është përdorur për të kontrolluar elementet tjera të HTML-së, të cilat nuk janë të specifikuara nga HTML kontrollat e serverit, siç janë **<body>**, **<span>**, **<div>**, **<font>**, **<p>** etj. Kontrollat duhet futur brenda kontrollës HTMLForm. Përmes këtij elementi subrutina e ekzekuton p.sh mesazhin për ta paraqitur në ekran. Nga shembulli i mësipërm shihet rasti i këtyre kontrollave ku ishte përdorur elementi **<p>**.

## Kontrolla HTMLSelect

Kontrolla HTMLSelect përdoret për të kontrolluar elementin **<select>**. Ajo fillon me tagun **<select>** dhe duhet të mbyllet me tagun **</select>**. Për të përcaktuar vlerën e zgjedhur nga një selektim i veçantë, së pari e shfrytëzojmë vetinë *SelectItem*

për të marrë një indeks të entitetit të zgjedhur. Në mes të tagjeve `<select>` dhe `</select>`, mund të jenë disa tagje `<option>`. Vetia **“id”** e identifikon në mënyrë unike kontrollën HTMLSelect, ndërsa vetia **“value”** brenda tagjeve `<option>` e identifikon në mënyrë unike vlerën e zgjedhur në HTMLSelect kontrollë.

#### Shembull 4:

Në këtë shembull ne kemi deklaruar një HTMLImage dhe një HTMLSelect kontrollë, edhe këtu duhet të kemi parasysh që këto kontrollat duhet futur brenda HTMLForm kontrollës. Ne do të modifikojmë një imazh përmes vetisë **SRC** e cila do të varet nga zgjedhja e shfrytëzuesit. Vlera e zgjedhur në HTMLSelect kontrollë përcakton cili imazh të paraqitet. Kodi për këtë shembull do të jetë:

```
<script runat="server">
  Sub zgjedh_imazhin(Sender As Object, e As EventArgs) imazhi1.Src =
"/images/" & zgjedhja1.Value
End Sub
</script>
<html>
<body>
<form runat="server">
<select id="zgjedhja1" runat="server"> <!--elementi select me vetin unike "id"
zgjedhja1 që do të thirret tek subrutina për ta shfaqur një të një prej opsioneve -->
  <option value="smiley.gif">Qesharake</option> <!--opsioni 1 gjatë
selektimit-->
  <option value="angry.gif">Zemruar</option> <!--opsioni 2 -->
  <option value="stickman.gif">Njerishkopar</option><!--opsioni 3 -->
</select>
<input type="submit" runat="server" value="Shfaq imazhin"
OnServerClick="zgjedh_imazhin"> <br /><br />
 <!--kontrolla HTMLImage që ka vlerën unike "id" me
qëllim që të thirret imazhi gjatë ekzekutimit të subrutinës nga kontrollat
HTMLButton-->
</form></body>
</html>
```

## Kontrolla HTMLTextArea

Kontrolla HTMLTextArea përdoret për të kontrolluar elementin `<textarea>`. Ajo ka tagun e hapjes `<textarea>` dhe tagun e mbylljes `</textarea>`. Në HTML, elementi `<textarea>` përdoret për të krijuar një hapësirë (vend, sipërfaqe) për shënim të tekstit apo thënë më mirë ky element mundëson një tekst boks shumë linjesh (vijësh). Dimensionet e këtij tekstboksi janë të kontrolluara nga rreshtat **“rows”** dhe numri i kolonave **“cols”**. Me **“cols”** përcaktohet gjerësia e kontrollës, përderisa me **“rows”** përcaktohet gjatësia (lartësia) e kontrollës. Kjo kontrollë përmban ngjarjen **“Serverchange”** e cila shkaktohet kur ndryshon përmbajtja e `“textare_s”`, me këtë rast do të ekzekutohet funksioni me emër të njëjtë.

## Shembull 5:

Në këtë shembull ne kemi deklaruar një kontrollë HTMLTextArea, një HTMLInputButton kontrollë dhe një HTMLGeneric kontrollë në një fajll të .aspx. Gjithashtu këtu duhet të kemi parasysh se këto kontolla duhet të shënohen brenda kontrollës HTMLForm. Kur butoni “submit” shtypet atëherë do të ekzekutohet “subrutina”, dhe do të shfaqë tekstin “Ju shkruat:” dhe tekstin që keni shënuar në teksthapsirë përmes elementit p .Kodi për këtë shembull do të jetë:

```
<script runat="server">
Sub submitimi(sender As Object, e As EventArgs)
    p1.InnerHtml = "<b>Ju shkruajtët:</b> " & teksthapsira1.Value <!--ky
mesazh "Ju shkruajtët" së bashku me vlerën e marrur përmes tekst hapsirës do
të paraqitet përmes elementit <p> në ekranin tuaj-->
End Sub
</script>
<html>
<body>
<form runat="server">
<br>Shkruaj ca tekst:<br/>
<textarea id="teksthapsira1" cols="35" rows="6" runat="server"/> <!--kontrolla
textarea me vlerë unike "id" dhe me 35 kolona dhe 6 rreshta, kuptohet kontrollë
e serverit-->
<input type="submit" value="Shtyp" OnServerClick="submitimi"
runat="server"/> <!--kontrolla HTMLInputButon për ta paraqitur një buton hyrës
(submitues) e cila do ta ekzekutoj "subrutinen" me një klik mbi të-->
<p id="p1" runat="server"/> <!--kontrolla HTMLGeneric-->
</form>
</body>
</html>
```

## Kontrolla HTMLImage

Kontrolla HTMLImage përdoret për të kontrolluar elementin **<img>**. Kjo kontrollë nuk ka tag mbyllës dhe gjithashtu duhet të shënohet në mes tagjeve **<form>** dhe **</form>**. Brenda kësaj kontrole duhet të vendoset atributi **runat="server"** për të treguar se është kontrollë e serverit. Në HTML, elementi **<img>** përdoret për të paraqitur një imazh. Kjo kontrollë mundëson (lejon) që dinamikisht të vendosim dhe të rigjejmë burimin e imazhit, gjerësinë dhe lartësinë e tij, gjerësinë e skajeve, përshkrimet e shkurtëra të imazhit dhe pozitën se ku do të gjendet imazhi, përmes vetive **src**, **width**, **height**, **border**, **alt** dhe **align**.

Çdo imazh i vendosur duhet të ketë prapashitesën: .jpg, .gif, .bmp etj. Atributi “id” duhet të ketë vlerë unike që t’i referohet kodit në “subrutinë”.

Përmes atributit **Align** ne vendosim imazhin në:

- Top (pjesën e epërme)
- Middle (në mes-qendër)
- Bottom (pjesën e fundme)
- Left (pjesën e majtë)
- Right (pjesën e djathtë)

Atributi **Alt** jep një përshkrim të shkurtër të imazhit kur vendoset kursori mbi imazh.

Atributi **Border** tregon gjerësinë e skajeve për rreth imazhit.

Vetia **Height** tregon gjatësinë përderisa ajo **Width** tregon gjerësinë.

Vetia **SRC** është shumë e rëndësishme dhe tregon URL e imazhit që duhet të paraqitet, pra burimin e fajllit të imazhit.

**Shembulli 4\*:** Një shembull ku është përdorur kontrolla `HTMLImage` është shembulli 4.

## Kontrolla `HTMLInputButton`

Kontrolla `HTMLInputButton` përdoret për të kontrolluar elementet `<input type="button">`, `<input type="submit">` dhe `<input type="reset">`. Në HTML, këto elemente janë përdorur për të krijuar një buton për komandë, një buton për dërgim dhe një buton për fshirje. Kur shfrytëzuesi përdorë një klik mbi kontrollën `HTMLInputButton` ai dërgon informacionin deri te serveri për përpunim. Pastaj përgjigja kthehet prapa deri te “browseri” juaj nga serveri. Duhet theksuar se me klikimin e butonit “reset” nuk fshihen të gjitha kontrollat hyrëse në faqe por kthehen në gjendjen origjinale, si në momentin kur faqja është hapur. Kur përdoret lidhja në mes kontrollave `HTMLInputText` dhe `HTMLTextArea`, mund të krijohet një faqe me hyrje të vërtetuar (password) ose hyrje për vërtetim të shfrytëzuesit (user name), e cila do të përpunohet nga serveri. Edhe kjo kontrollë nuk ka tag mbyllës. Përmes vetisë *type* deklarohet se çfarë tipi është kontrolla “button”, “submit” ose “reset”, ndërsa përmes vetisë *value* shënohet vlera që dëshirohet të paraqitet mbi buton.

### Shembull 6:

Në këtë shembull po e deklarojmë një `HTMLInputText` kontrollë, dy kontrollat `HTMLInputButton` dhe një `HTMLGeneric` kontrollë. Kur butoni “submit” shtypet atëherë do të ekzekutohet subrutina, ndërsa kur shtypet butoni “reset” do të pastrohet teksti i shkruar në kontrollën *InputText*. Submit subrutina do të shkruaj mesazhin e mirëseardhjes përmes elementit p. Kodi për këtë shembull do të jetë:

```
<script runat="server">
Sub submit(sender As Object, e as EventArgs) <!--submit subrutina-->
if emri.value<>"" then <!--ketu kemi një degëzim që thotë se nëse në kontrollën
InputText ka një vlerë, paraqite mesazhin "Mirësevjen" së bashku me atë vlerë
në ekran-->
    p1.InnerHtml="Miresevjen " & emri.value & "!"
end if
End Sub
</script>
<html>
<body>
<form runat="server">
Shenoni emrin tuaj: <input id="emri" type="text" size="30" runat="server" />
<br /><br />
<input type="submit" value="Submit" OnServerClick="submit" runat="server"
/>
<!--kontrolla InputButton me tipin "submit", me një klik mbi këtë buton do të
ekzekutohet subrutina me emrin submit-->
<input type="reset" value="Pastro" runat="server"/> <!--kontrolla
HTMLInputButton me tipin "reset", e cila do ta pastroj fushën me tekst dhe do ta
kthej në gjendjen e nënkuptueshme-->
<p id="p1" runat="server" /> <!--kontrolla HTMLGeneric-->
</form>
```

```
</body>  
</html>
```

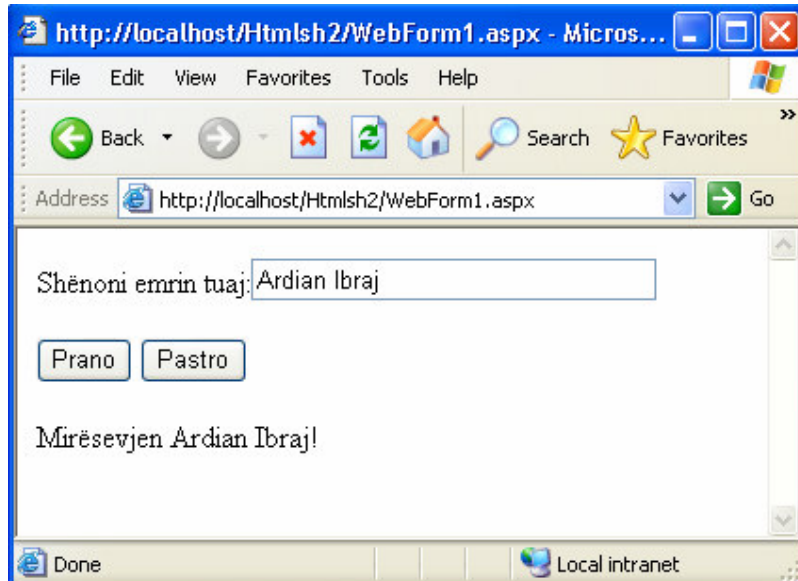


Figura 8. Kontrolla HtmlInputButton

## Kontrolla HTMLInputCheckBox

Kontrolla HTMLInputCheckBox përdoret për të kontrolluar elementin **<input type="checkbox">**. Në HTML, ky element përdoret për të krijuar një buton kontrollues. Kjo kontrollë mundëson zgjedhjen e disa opsioneve njëkohësisht. Mirëpo, kur klikojmë mbi një **"checkbox"** kontrollë, e dhëna nuk postohet ("dërgohet") deri tek serveri, gjendja e "checkbox-it" dërgohet tek serveri për përpunim atëherë kur përdorim kontrollën siç është "HTMLInputButton". Edhe kjo kontrollë nuk ka tag mbyllës. Përmes vetisë **"name"** që posedon kjo kontrollë, futen vlerat identike për elementin, ndërsa përmes vetisë **"type"** specifikohet se kemi të bëjmë me vetinë **"checkbox"**. Kjo kontrollë përfshin ngjarjen **"serverchange"** e cila shfaqet kur gjendja e kontrollës ndryshohet.

### Shembull:

Shembull për këtë kontrollë do ta paraqesim më vonë së bashku me disa kontrollat hyrëse ("input").

## Kontrolla HTMLInputRadioButton

Kontrolla HTMLRadioButton përdoret për të kontrolluar elementin **<input type="radio">**. Në HTML, ky element përdoret për të krijuar një buton kontrollues. Mund të krijohet grup i kontrollave duke futur në vetinë **"name"** vlerën e

njejtë për të gjithë, ndërsa për dallim nga kontrolla HTMLCheckBox këtu ka mundësi të zgjedhet vetëm një opcion. Gjithashtu këtu nuk mund të postohen të dhënat në server vetëm nëse zgjedhet një opcion, por kjo mund të bëhet përmes kontrollave HTMLButton, HTMLInputButton ose HTMLInputImage. Edhe këtu mund të shfrytëzohet ngjarja “serverchange” e cila shfaqet kur ndryshon gjendja e “radiobutonit”.

### Shembulli 7:

Në këtë shembull kemi paraqitur tri kontrollat HTMLInputRadioButton, një HTMLGeneric dhe një HTMLInputButton për zgjedhjen e njërës nga ngjyrat që e preferoni dhe keni mundësi të zgjidhni vetëm njërën nga ngjyrat. Pas shtypjes së butonit “submit” tek atëherë do të submitohen të dhënat (dërgohen të dhënat tek serveri) sepse atëherë do të ekzekutohet subrutina. Kodi për këtë shembull do të jetë:

```
<script runat="server">
Sub submit(Source As Object, e As EventArgs)
if r1.Checked=True then <!--nëse zgjedhet vlera e parë që ka vlerën unike "r1"
do të ekzekutohet kodi tek elementi "p1"-->
    p1.InnerHtml=" Ngjyra e preferuar është e kuqe"
else
    if r2.Checked=True then <!--nëse zgjedhet vlera e dytë që ka vlerën unike "r2"
do të ekzekutohet kodi me elementi "p1"-->
        p1.InnerHtml="Ngjyra e preferuar është e gjelbërt"
    else
        if r3.Checked=True then <!--nëse zgjedhet vlera e tretë që ka vlerën unike
"r3" do të ekzekutohet kodi tek elementi "p1" si në vazhdim-->
            p1.InnerHtml="Ngjyra e preferuar është e kaltërta"
        end if
    end if
end if
End Sub
</script>
<html>
<body>
<form runat="server">
<p>Select your favorite color:
<br />
<input id="r1" name="col" type="radio" runat="server"> Kuqe</input>
<br />
<input id="r2" name="col" type="radio" runat="server"> Gjelbërt</input>
<br />
<input id="r3" name="col" type="radio" runat="server"> Kaltër</input>
<br /> <!--kontrollat input që kanë vlerë të njejtë në vetinë name ndërsa vlerë
unike në vetinë "id" dhe janë deklaruar se janë kontrollat të serverit-->
<input type="button" value="Submit" OnServerClick="submit" runat="server"/>
<p id="p1" runat="server" />
</form>
</body>
</html>
```

## Kontrolla HTMLInputText

Kontrolla HTMLInputText përdoret për të kontrolluar elementet **<input type="text">** dhe **<input type="password">**. Në HTML, këto kontrollat janë përdorë për të krijuar një fushë me tekst dhe me fjalëkalim (password). Këto kontrollat lejojnë për të krijuar një vijë të veçantë tekstboksit për të marrë një tekst hyrës nga shfrytëzuesi,

ndërsa kur vetisë **"type"** i ndajmë vlerën "password", atëherë tekstin që e fut shfrytëzuesi do të jetë i maskuar. Mund të kontrollohet numri i karakterëve duke e futur vetinë **"maxlength"** (numri maksimal i karaktereve). Me vetinë **"size"** kontrollohet gjerësia e tekstboks-it ndërsa vetia **"value"** i jep një vlerë të nënkuptueshme tekst boks-it. Atributi **"id"** paraqet vlerën unike për elementin ndërsa ai **runat="server"** specifikon se elementi është një kontrollë e serverit.

### Shembulli:

Shembull për këtë kontrollë kemi paraqitur në shumë shembuj të mëhershëm dhe do të paraqesim edhe në shembujt e më vonshëm.

## Kontrolla HTMLInputImage

Kontrolla "HTMLInputImage" përdoret për të kontrolluar elementin **<input type="image">**. Në HTML, ky element përdoret për të krijuar një buton hyrës duke përdorur një imazh, në vend të butonit të zakonshëm shtypës. Edhe kjo kontrollë nuk ka tagun mbyllës. Përmes vetisë **"Align"** tregohet rreshtimi i imazhit, përmes vetisë **"Alt"** ipet përshkrim i shkurtër kur vendoset mausi mbi imazh, përmes vetisë **"borders"** caktohet gjerësia e skajeve të imazhit. Ngjarja që përdorë kjo kontrollë është **"OnServerClick"**, me çka ekzekutohet funksioni që posedon atë emër kur klikohet mbi imazh. Vetia **"SRC"** tregon vendin (burimin) ku gjendet imazhi.

### Shembull:

Shembull për këtë kontrollë do ta paraqesim më vonë së bashku me disa kontrollat hyrëse ("input").

## Kontrolla HTMLInputFile

Kontrolla HTMLInputFile përdoret për të kontrolluar elementin **<input type="file">**. Në HTML, ky element përdoret për të bërë një **"upload"** fajllit në server. Vlen të ceket se nuk ka tag mbyllës kjo kontrollë. Kjo kontrollë shfrytëzohet për të dizajnuar lehtë faqen e cila lejon shfrytëzuesin të "uplojoj" fajlle tekstuale ose binare nga browseri te direktoriumi që i caktohet Web Server-it.

### Shembulli 8:

Në këtë shembull kemi deklaruar një kontrollë `HTMLInputFile`, një `HTMLInputButton` dhe tri kontrollat `HTMLGeneric` në një fajll `.aspx`. Kur butoni “dërgues” shtypet atëherë do të ekzekutohet subrutina. Emri i fajllit dhe tipi i fajllit do të shfaqen në faqe, përderisa fajlli vetveten e “uploadon” në direktoriumin `C` të serverit. Kodi për këtë shembull do të jetë:

```
<script runat="server">
Sub submit(Sender as Object, e as EventArgs) <!--kur ekzekutohet kjo subrutinë
do të shfaqet emri i fajllit dhe kapaciteti i tij -->
    emrif.InnerHtml=MyFile.PostedFile.FileName
    gjatsiK.InnerHtml=MyFile.PostedFile.ContentLength
    Fajlliim.PostedFile.SaveAs("c:\uploadfile.txt")
End Sub
</script>
<html>
<body>
<form method="post"
enctype="multipart/form-data" runat="server"> <!--Ne kete rastë shihet që
është përdorur vetia "method" me vlerën post që do të thotë fajlli të postohet deri
te serveri-->
<p>
Zgjedhë fajllin për të uploaduar në server:
<input id="Fajlliim" type="file" size="40" runat="server"> <!--kontrolla
HTMLInputFile për të treguar serverit që kemi të bëjmë me fajll-->
</p>
<p>
<input type="submit" value="Uplad!" OnServerclick="submit"
runat="server">
</p>
<p>
<div runat="server"> <!--kontrollë HTMLGeneric-->
    Emri I fajllit: <span id="emrif" runat="server"/><br />
    Kapaciteti I fajllit: <span id="gjatsiK" runat="server"/> bajta
</div>
</p>
</form>
</body>
</html>
```

## Kontrolla `HTMLInputHidden`

Kontrolla `HTMLInputHidden` përdoret për të kontrolluar elementin **`<input type="hidden">`**. Në HTML, kjo kontrollë përdoret për të krijuar një fushë hyrëse të fshehur. Kjo kontrollë është pjesë e formës, por ajo asnjëherë nuk shfaqet në formë. Gjithashtu kjo kontrollë nuk përmban tag mbyllës. Atributi “**id**” është unik dhe atributi **`runat="server"`** tregon se kontrollat është e serverit. Kjo kontrollë e ka ngjarjen “**serverchange**” kur përmbajtja e elementit të ndryshohet.

### Shembulli 9:

Në këtë shembull ne kemi deklaruar disa kontrollera por çka është me rëndësi në këtë shembull kemi deklaruar kontrollën `HTMLInputHidden`. Në këtë rast kur shtypet butoni `submit`ues do të ekzekutohet subrutina. Subrutina me emrin “`submit`” vlerën e fajllit të fshehur e barazon me vlerën e marrur nga fusha, dhe kur futet vlera për herë të dytë në fushë ajo e shfaq vlerën e parë të fshehur përmes elementit `p`. Kodi për këtë shembull do të jetë:

```
<script runat="server">
Sub submit(Source As Object, e As EventArgs) <!--subrutina me emrin submit që
do të ekzekutohet pasi të klikohet mbi butonin "submitues"-->
    hidden1.Value=string1.Value <!--deklarimi i variablës hidden1 si string
variabël-

p1.InnerHtml="Vlera e fshehur= " + hidden1.Value <!--do të shfaqet mesazhi
"vlera e fshehur = " plus edhe vlera e parë e fshehur-->
End Sub
</script>
<html>
<body>
<form runat="server">
Shkruaj ca tekst: <input id="string1" type="text" size="25" runat="server" />
<input type="submit" value="Submit" OnServerClick="submit" runat="server"
/>
<input id="hidden1" type="hidden" runat="server" /> <!--kontrollera me tipin
"hidden" e cila në këtë rast e fshehë vlerën e shkruar në kontrollën
HTMLInputText->
<p id="p1" runat="server" />
</form>
</body>
</html>
```

## Kontrollera HTMLTable

Kontrollera `HTMLTable` përdoret për të kontrolluar elementin `<table>`. Në HTML, ky element përdoret për të krijuar tabela. Një kontrollë e tillë është e ndërtuar nga “`rows`” (rreshtat) dhe “`cells`” (kolonat). Kontrollera `HTMLTable` lejon gjithashtu të modifikohet pamja e tabelës. Me vetinë “`BGColor`” mund të specifikohet ngjyra e tabelës, me vetinë “`Border`” gjerësia e skajeve të tabelës, me vetinë “`BorderColor`” ngjyra e skajeve si dhe gjerësinë dhe gjatësinë e tabelës me vetinë “`width`” dhe “`height`”.

Duke përdorur vetinë “`Cellspacing`” dhe “`Cellpadding`” mund të kontrollohet hapësira në mes kolonave dhe hapësira në mes përmbajtjes së kolonës dhe skajeve të kolonës

### Shembull:

Shembull për këtë kontrollë do ta paraqesim më vonë së bashku me disa kontrollera tjera.

## Kontrolla HTMLTableRow

Kontrolla HTMLTableRow përdoret për të kontrolluar elementin `<tr>`. Në HTML, ky element përdoret për të krijuar rreshta të tabelës. Duke përdorur vetinë `"BgColor"` kontrollohet ngjyra e një rreshti, me vetinë `"BorderColor"` kontrollohet ngjyra e skajeve të rreshtit si dhe gjatësia dhe gjerësia përmes vetive `"Height"` dhe `"Width"`. Këtu vlen të ceket se kontrolla ka edhe tagun mbyllës `</tr>`. Rreshtimi vertikal dhe horizontal i përmbajtjes së kolonës në rresht kontrollohet me përdorimin e vetive `"Align"` dhe `"VAlign"`.

### Shembull:

Shembull për këtë kontrollë do ta paraqesim më vonë së bashku me disa kontrolla tjera.

## Kontrolla HTMLTableCell

Kontrolla HTMLTableCell përdoret për të kontrolluar elementet `<td>` dhe `<th>`. Në HTML, këto elemente përdoren për të krijuar kolonën (ose celulata) e tabelës dhe kokën e celuleve të tabelës.

Pra elementi `<td>` prezanton të dhënat e kolonës, përderisa elementi `<th>` prezanton kokat e kolonave. Përmbajtja e kolonës `<th>` kur shfaqet në tabelë gjithëherë është **"bold"** (me ngjyrë të plotë të trashë) dhe vlera e saj gjendet në qendër. Përmes vetisë `"BgColor"` caktohet ngjyra e kolonës, ndërsa ngjyra e skajeve të kolonës caktohet përmes vetisë `"BorderColor"`. Gjatësia dhe gjerësia e kolonave caktohet përmes vetive `"height"` dhe `"width"`. Me vetinë `"Colspan"` paraqitet numri i kolonave që do t'i zë hapësira e një kolone, ndërsa me vetinë `"Rowspan"` paraqitet numri i rreshtave që do t'i zë hapësira e një kolone.

### Shembulli 10:

Në këtë shembull kemi paraqitur dy HTMLSelect kontrollat dhe kontrollat HTMLInputTabel, HTMLInputRow dhe HTMLInputCell. Me anë të kontrollës **select** kemi mundësi të zgjedhim numrin e kolonave dhe rreshtave, që pas shtypjes së butonit me vlerën **"Paraqite tabelen"**, do të ekzekutohet subrutina me emrin **"submit"** dhe do të paraqitet tabela. Kodi për këtë shembull do të jetë:

```
<script runat="server">
Sub submit(sender As Object, e As EventArgs)
Dim row,numrows,numcells,j,i <!--deklarimi i variablave-->
row=0 <!--variablës "row" i ndahet vlera 0-->
numrows=rows1.Value
numcells=cells1.Value
for j=1 to numrows
Dim r As New HtmlTableRow()
row=row+1
for i=1 to numcells
Dim c As New HtmlTableCell()
c.Controls.Add(New LiteralControl("rreshti " & j & ", kolona " & i))
```

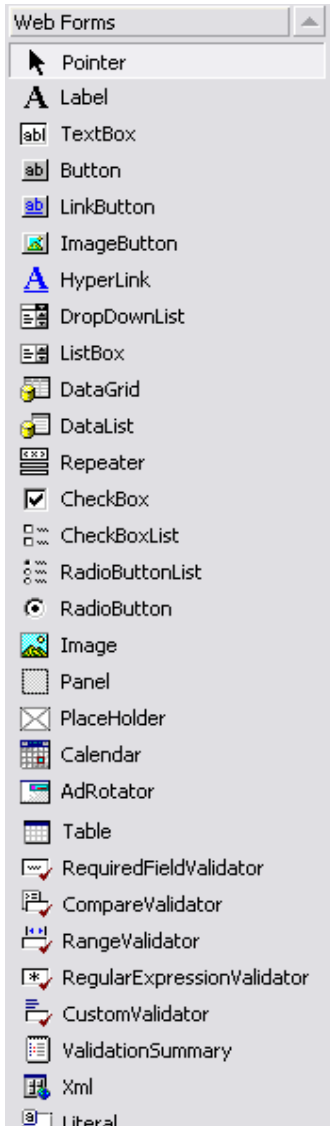
```

r.Cells.Add(c)
next
t1.Rows.Add(r)
t1.Visible=true
next
End Sub
</script>
<HTML>
<body>
<form runat="server"><p>
Rreshtat e tabelës:
<select id="rows1" runat="server"> <!--kontrolla select, që pas selektimit dhe
submitimit nga butoni do të paraqet numrin e rreshtave-->
    <option value="1" >1</option> <!--opcioni i parë zgjedhës-->
    <option value="2">2</option> <!--opcioni i dytë zgjedhës-->
    <option value="3">3</option> <!--opcioni i tretë zgjedhës-->
</select>
<br>
Kolonat e tabelës:
<select id="cells1" runat="server">
    <option value="1">1</option> <!--opcioni i parë zgjedhës-->
    <option value="2">2</option> <!--opcioni i dytë zgjedhës-->
    <option value="3">3</option> <!--opcioni i tretë zgjedhës-->
    <option value="4">4</option> <!--opcioni i katërt zgjedhës-->
</select>
<br><br>
<input type="submit" value="Paraqite tabelën" runat="server"
OnServerClick="submit"></p> <!--kontrolla HTMLInputButton, për submitim të të
dhënave-->
<table id="t1" border="1" runat="server" visible="false">
</table>
</form>
</body>
</HTML>

```

## Pjesa e dytë

### ASP Server Kontrollat



ASP Server kontrollat njihen edhe si Web kontrollat apo edhe si Web Form kontrollat. Prandaj, në vazhdim të kësaj “përmbledhjeje” do të përdoret termi “Web kontrollat”, për të nënkuptuar se fjala është për Server kontrollat. Në vazhdim, në figurën 1, janë paraqitur pothuajse të gjitha Web kontrollat e mundshme. Në këtë lëmë, Microsofti ka futur një numër të madh Web kontrollash në kuadër të nejmshpejsit (ang. *Namespace*) “**System.Web.UI.WebControls**”.

Këto kontrollat janë përmbledhur në tri kategori kryesore:

- ◆ **Web kontrollat themelore (bazike), (ang. Basic Web Controls)**
- ◆ **Kontrollat e validimit (ang. Validation Controls)** dhe
- ◆ **ListKontrollat për të dhënat e kufizuara (Databound ListControls)**

Parimisht, të gjitha Web kontrollat e kanë prejardhjen nga një klasë e përgjithshme e quajtur **WebControl**. Kështuqë, Web kontrollat e trashëgojnë të njëjtën bashkësi të anëtarëve të klasës. Disa nga anëtarët e kësaj klase që më së shumti përdoren janë:

*BackColor, BorderColor, BorderStyle, BorderWidth, DataBind Enabled, Font, ForeColor, Height, Page, Parent, Site, TabIndex, ToolTip, Visible, Init, Load, Unload, Dispose, ToString, OnInit, OnLoad dhe OnDataBinding.*

Figura 1. Web Form kontrollat (Server kontrollat)

## Web kontrollat themelore (Basic Web Controls)

Në tabelën në vijim (tabela 1), në pika të shkurtëra janë shpjeguar disa nga funksionet e Web kontrollave më fundamentale (bazike). Disa nga këto kontrolla, sillen pothuajse ngjashëm në funksionet që kryejnë. P.sh. përdorimi dhe karakteristikat e **CheckBoxList** kontrollës janë pothuajse identike me ato të **RadioButtonList** kontrollës.

Për këtë arsye, në tabelën vijuese këto kontrolla janë grumbulluar në këtë mënyrë:

**Tabela 1. Web kontrollat themelore**

Server Kontrolla	Përshkrimi i funksionit
<b>Label</b>	<b>Label</b> kontrolla përdoret për shkruarjen e teksteve. Mirëpo, nëse dëshirojmë të paraqesim një tekst statik, atëherë nuk është e domosdoshme të përdorim <b>Label</b> kontrollën, por në vend të saj mund të përdorim gjuhën HTML. Përndryshe, <b>Label</b> kontrollën e përdorim kryesisht kur dëshirojmë që të bëjmë modifikime nëpërmjet kodit që ekzekutohet në anën e serverit.
<b>TextBox</b>	<b>TextBox</b> kontrolla i mundëson shfrytëzuesit që në fushën tekstuale të shkruajë tekstin e dëshiruar. Në rastin e përgjithshëm, atributi <i>TextMode</i> i <b>TextBox-it</b> mund të mirret si <i>SingleLine</i> (d.m.th në fushën tekstuale të paraqitet vetëm një rresht tekstual), mirëpo, kjo veti mundet gjithashtu edhe të programohet si <i>MultiLine</i> (me shumë se një rresht në fushën tekstuale) apo edhe <i>Password</i> (d.m.th, në këtë fushë mund të shkruhet edhe fjalëkalimi). Në rastin kur përdoret <b>TextBox-i</b> <i>MultiLine</i> , atëherë atributi <i>Rows</i> përcakton lartësinë e asaj fushe tekstuale. Në qoftë se atributi <i>AutoPostBack</i> është zgjedhur të jetë “True”, atëherë ai gjeneron një <i>PostBack</i> në ngjarjen <i>Text_Changed( )</i> të saj.
<b>Buttons</b>	Të tre llojet e butonëve, funksionet e të cilëve janë shpjeguar më poshtë, shkaktojnë <i>PostBacks</i> -a kur shfrytëzuesi të klikojë mbi to.

- **Button**

**Button** kontrollat mund të vendosen kudo përbrenda ndonjë kontrolle poseduese siq janë *DataList*-i, *DataGrid*-i dhe *Repeater*-i.

- **LinkButton**

**LinkButton** kontrollat bënë renderimin e hiperlinqeve në faqe.

- **ImageButton**

**ImageButton** kontrollat bënë të mundur paraqitjen e imazheve që i përgjigjen klikimeve të miut. Gjithashtu, kjo kontrollë mund të përdoret si një hartë imazhi. Kjo do të thotë se ne do të jemi në gjendje që saktësisht të lokalizojmë vendin në të cilin ka klikuar shfrytëzuesi.

## CheckBox

Kjo kontrollë i mundëson shfrytëzuesit të fut të dhëna të tipit **Boolean**: *True* ose *False* apo *Yes* ose *No*.

Atributi *Checked* (i zgjedhur) i kësaj kontrolle mund të lidhet me një fushë të *datasource*-it. Nga ana tjetër, ngjarja *CheckedChanged* mund të përdoret për *AutoPostBack*.

## ListControls

Këto kontrollat rrjedhin prej klasës abstrakte *ListControl*.

- **CheckBoxList**

Këto kontrollat zakonisht shpjegohen në kuadër të kontrollave të validimit dhe si të tilla mbeten për shpjegime në kuadër të këtij lëmi !!

- **DropDownList**

- **ListBox**

- **RadioButtonList**

## HyperLink

Kryesisht paraqitet në formë teksti, tekst ky i cili mund të modifikohet në kuadër të atributit *Text* (ang. *Text* property). Poashtu, hiperlinku mundet të paraqitet edhe në formë të imazhit i cili caktohet duke përdorur atributin *ImageUrl* (ang. *ImageUrl* property). Nga ana tjetër, nëse në të njëjtën kohë merren në përdorim të dyja nga këtotribute (veti), atëherë, si veti e nënkuptueshme (default), do të paraqitet vetia *ImageUrl*. Mirëpo, nëse imazhi nuk ekziston (mund të ndodhë për shkak se është marrë

gabimisht lokacioni ku ndodhet fajlli imazh apo për ndonjë arsye tjetër!), atëherë, në vend të imazhit do të paraqitet vetëm pjesa e tekstit e caktuar me atributin *Text*.

## Image

**Image** kontrolla përdoret zakonisht për vendosjen e imazheve në web faqe. Atributi *ImageUrl* tregon shtekun (lokacionin) e vendndodhjes së fajllit imazh. Në rastin kur fajlli imazh mungon në atë lokacion, ka mundësi që të caktohet vetëm teksti i cili do të paraqitet në vend të tij, tekst ky i cili modifikohet duke përdorur atributin *AlternateText* (ose shkurt *Alt*). Pra, **Image** kontrolla bën paraqitjen vetëm të imazheve në faqet e dëshiruara. Nëse dëshirohet që të aktivizohen edhe klikimet e miut mbi këto imazhe, atëherë, në vend të kësaj kontrole përdoret kontrolla **ImageButton**.

## Panel

Kjo kontrollë mund të përdoret në kuadër të kontrollave tjera, d.m.th. si kontrollë e “brendshme” e tyre. Kjo kontrollë automatikisht shndërrohet në HTML kod, nëpërmjet tagut `<div>` të HTML-it.

## RadioButton

Kjo kontrollë përdoret për të krijuar radio-butona në web faqe. Këta radio-butona mund të grupohen për të mundësuar zgjedhje të ndryshme aty ku ka mundësi të zgjedhim njëri nga veçoritë e ndryshme, p.sh. në rastin e gjinisë (M apo F).

## Table

Kjo kontrollë mundëson përdorimin e HTML tabelave në faqe. Tabelat mund të ndërtohen ashtu që në kuadër të përmbajtjes së tyre të futen të dhëna statike, mirëpo nëpërmjet kontrollës **Table** zakonisht bëhet futja e të dhënave me përmbajtje dinamike (d.m.th. me përmbajtje në të cilën do të mund të bëhen modifikime apo ndryshime në të ardhmen!). Mirëpo, këto ndryshime apo modifikime programore shtesë në rreshtat apo kolonat e tabelës, nuk ndikojnë drejtpërsëdrejti në kodin që është ruajtur në server. Ndryshimet që bëhen në rreshtat apo kolonat e tabelës duhet të rikonstruktohen pas çdo postimi (ang. post) në server. Në raste të tilla, si alternativa më të mira

për të kryer një funksion të tillë (d.m.th. për të kryer modifikimet e nevojshme), preferohet të përdoren kontrollat si **DataList-i** ose **DataGrid-i**.

## **Xml**

Kjo kontrollë shërben për transformimin e fajllave me prapashtesa të ndryshme, në fajlla me prapashtesën XML.

---

Shumica e këtyre Web kontrollave themelore të përshkruara më lartë funksionojnë pothuajse ngjajshëm me HTML Server kontrollat homologe të tyre. Të gjitha Server kontrollat (Web kontrollat) fillojnë me prefiksin *asp*: në tagjet e tyre. P.sh. tagu për Web kontrollën *Label* shkruhet në formën: `<asp: Label>`. Edhe përdorimet e tyre janë pothuajse intuitive. Nga ana tjetër, të gjithë shembujt e ilustruar në kuadër të HTML Server kontrollave, mundën që me efikasitet të zgjerohen dhe modifikohen me futjen në përdorim të Web kontrollave. Në vazhdim, me shembuj konkret do të ilustruhet përdorimi i Web kontrollave.

### **Përdorimi i ngjarjeve të kontrollave**

Deri para disa kohësh, kur .NET-i akoma nuk ishte paraqitur në “skenë”, vizitorit të ndonjë Web-faqeje me shembuj, i është dashur të klikojë në butona të ndryshëm për të parë rezultatin për zgjedhjen e ndonjë liste të caktuar produktesh që ai ka vendosur ta paraqes para vetes.

Por, këtu parashtrohet pyetja se, çfarë nëse do të kishte mundësi të paraqitjes së kësaj liste produktesh menjëherë pasi vizitori të ketë bërë zgjedhjen e dëshiruar në kuadër të ndonjë kategorie të caktuar produktesh në atë listë, duke mos patur nevojë që ai të klikojë në ndonjë buton?. Natyrisht se një gjë e tillë, për fat të mirë është e mundur!

Për të parë këtë mundësi, në vazhdim kemi një shembull ilustrues, fajllin *Products-Web2.aspx*, i cili është treguar në figurën 2. Në këtë faqe nuk shihet të ketë ndonjë buton mirëpo sa më parë që shfrytëzuesi të zgjedhë një kategori të produkteve, menjëherë paraqitet një tabelë me produktet që hyjnë në kuadër të asaj kategorie (fig.2).

Kodi i mëposhtëm do të ekzekutohet kur zgjedhet një kategori në DropDownList (nga figura e mëposhtme).

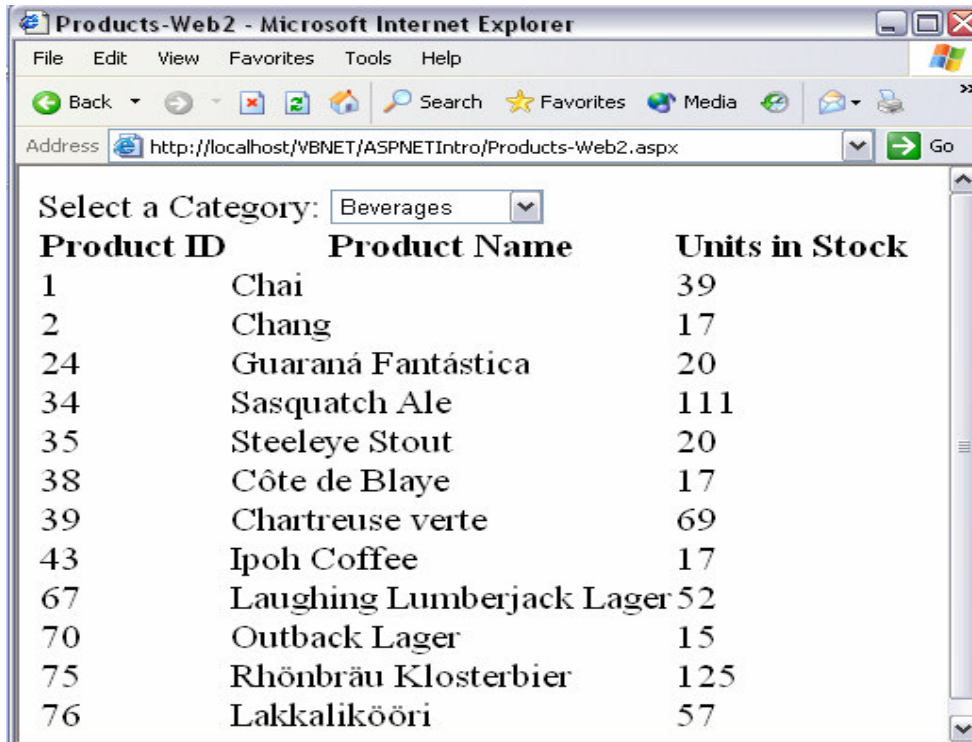


Figura 2. Përdorimi i DropDownList Box-it për gjenerimin e ngjarjes “postback”

```

Private Sub cboCategory_SelectedIndexChanged(ByVal sender As Object,
ByVal e As System.EventArgs) Handles cboCategory.SelectedIndexChanged
    Dim cnn As New SqlConnection( _
        "Data Source=AMD;Initial Catalog=Northwind;Integrated
Security=SSPI")
    Dim cmd As SqlCommand
    Dim rdr As SqlDataReader

    grdProducts.Visible = True
    cmd = New SqlCommand( _
        "SELECT ProductID [Product ID], ProductName [Product
Name], UnitsInStock [Units in Stock] FROM Products" _
        & " WHERE CategoryID = " &
cboCategory.SelectedItem.Value, _
        cnn)
    cnn.Open()
    rdr = cmd.ExecuteReader(CommandBehavior.CloseConnection)

    grdProducts.DataSource = rdr
    grdProducts.DataBind()
    rdr.Close()
    cnn.Close()
End Sub

```

Nga kodi shihet se tabela e cila në anën e serverit është një **DataGrid**, do të mbushet me produktet që i përkasin kategorisë së selektuar në **DropDownList**. Është me rëndësi, të shihet HTML kodi që gjenerohet në anën e klientit dhe që e bënë të

mundur realizimin e kësaj ngjarjeje. Do të shohim se ASP.NET ka gjeneruar kodin në javascript në anën e klientit për ta shkaktuar (ang. to trigger) ngjarjen, dhe njëra kontrollë hyrëse e cila është e padukshme e quajtur “\_\_EVENTTARGET” , përdoret për të treguar se cila kontrollë e ka shkaktuar ngjarjen. Ndërsa, kontrolla e dytë “\_\_EVENTARGUMENT”, e cila poashtu është e padukshme, përdoret për ruajtjen e ndonjë argumenti të ngjarjes, e cila në rastin tonë është vetëm një string i zbrazët. Në vazhdim është paraqitur HTML kodi i cili gjeneron një **DropDownListBox** dhe shkakton ngjarjen “postback” në rastin kur vlera e indeksit të tij të ndërrohet:

---

```
...
<form name="Form1" method="post" action="Products-Web2.aspx"
id="Form1">
<input type="hidden" name="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" value="" />

<script language="javascript">
<!--
    function __doPostBack(eventTarget, eventArgument) {
        var theform;
        if (window.navigator.appName.toLowerCase().indexOf("netscape")
> -1) {
            theform = document.forms["Form1"];
        }
        else {
            theform = document.Form1;
        }
        theform.__EVENTTARGET.value = eventTarget.split("$").join(":");
        theform.__EVENTARGUMENT.value = eventArgument;
        theform.submit();
    }
// -->
</script>

        <span id="Label1">Select a Category:</span>
        <select name="cboCategory"
onchange="__doPostBack('cboCategory','')" language="javascript"
id="cboCategory">
            <option value="1">Beverages</option>
            <option value="2">Condiments</option>
            <option value="3">Confections</option>
            <option value="4">Dairy Products</option>
            <option value="5">Grains/Cereals</option>
            <option value="6">Meat/Poultry</option>
            <option value="7">Produce</option>
            <option value="8">Seafood</option>
        </select>
<br> </form>
```

---

## Pjesa e tretë

# Kontrollat Validuese

## Kontrollat Validuese si Web Form Kontrolla

Kontrollat Validuese (**Validation Controls**) së bashku me Kontrollat Bazike të Webit (**Basic Web Controls**) dhe List-Kontrollat Databound (**Databound ListControls**), janë të njohura si **Web Form Kontrolla**. Të gjitha këto Microsofti i ka përfshirë në një *Namespace* **System.Web.UI.WebControls**, kështu që Web Kontrollat trashëgojnë një bashkësi të përbashkët të antarëve të klasës dhe e trashëgojnë klasën me emrin **WebControls**. Këto kontrolla janë zhvilluar në mënyrë ekskluzive për validimin e kontrollave hyrëse. Këto kontrolla mundësojnë të kontrollohet një fushe apo kontrollë hyrëse dhe të paraqitet mesazh gabimi nëse është e nevojshme. Shfrytëzimi i tyre ka një rëndësi shumë të madhe për aplikacionin, duke e bërë atë shumë më efikas, më të lehtë dhe gjithashtu me të shpejtë për ta programuar.

## Kontrollat Validuese

Kontrollat validuese mundësojnë që të validohet apo të kontrollohet një fushë apo kontrollë hyrëse dhe të paraqitet mesazh gabimi nëse është e nevojshme. Këto kontrolla kanë ngjashmëri të madhe me server kontrollat tjera, me disa metoda dhe veti shtesë. Fillimisht serveri i trajton ato si kontrolla të padukshme, mirëpo me futjen e ndonjë të dhëne të gabuar ato bëhen të dukshme. Shfrytëzimi i këtyre kontrollave validuese bën që aplikacioni të zhvillohet shumë më shpejt dhe të jetë më efikas.

Me qëllim që të kuptohen këto kontrolla më mirë, janë shqyrtuar metodat dhe vetitë e tyre në mënyrë individuale me disa shembuj mjaft të thjeshtë.

Tipet e ndryshme të kontrollave validuese janë:

- ◆ **RequiredFieldValidator** - kontrollon nëse kontrolla hyrëse ka shënim apo vlerë.
- ◆ **RegularExpressionValidator** - përdoret për përputhje me ndonjë model apo mostër të caktuar.
- ◆ **CompareValidator** - kontrollon nëse vlera është e pranueshme, duke e krahasuar me ndonjë vlerë të caktuar ose me vlerën e përmbajtjes së ndonjë kontrolle tjetër.
- ◆ **RangeValidator** - kontrollon se vlera e kontrollës përkatëse është brenda një rangui të caktuar.
- ◆ **CustomValidator** - përdoret për validim në raste më komplekse
- ◆ **ValidationSummary** - mundëson paraqitjen e të gjitha gabimeve në një lokacion të caktuar.

Zakonisht secila kontrollë validuese e kryen validimin apo kontrollimin, si në anën e klientit ashtu edhe në anën e serverit. Pëveq kontrollës *RequiredFieldValidator*, të gjitha kontrollat tjera validuese e trajtojnë një fushë të

zbrazët si fushë valide, pra kontrolla *RequiredFieldValidator* duhet të përdoret për secilën kontrollë hyrëse që të mos

ketë fushë të zbrazët. Mund të përdoren më shumë se një kontrollë validuese, për një kontrollë hyrëse, si p.sh. përdorimi i *RequiredFieldValidator* dhe *RangeValidator* sigurojnë që hyrja nuk është e zbrazët dhe është brenda një rangu të specifikuar.

Këto kontrolla kanë shumë attribute apo veti, por do ti cekim vetëm disa që janë më të rëndësishme:

- **ErrorMessage** - Në rast të ndonjë gabimi, sistemi e paraqet këtë mesazh ne lokacionin e kontrollës.
- **Display** – Kontrolla validuese është e padukshme, derisa të futet një e dhënë e padëshirueshme apo e gabueshme. Në rast të dhënies së gabueshme sistemi do ta paraqet një mesazh gabimi. Mekanizmi i paraqitjes së mesazhit mund të trajtohet në njëren nga këto tri mënyra: *static*, *dinamic* dhe *none*.

## Kontrolla *RequiredFieldValidator*

Kjo kontrollë përdoret kur kërkohet që ndonjë kontrollë hyrëse të mos ketë fushë të zbrazët, pra kur kërkohet që fusha të plotësohet patjeter.

Në shembullin e mëposhtëm, shfrytëzuesi pritët që ti fus dy vlera. Nëse ai nuk e plotëson njëren nga këto dhe klikon butonin submit, sistemi do ta paraqes mesazhin e gabimit. Për zhvillimin e këtij aplikacioni dhe për kontrollimin e fushës nuk duhet kod shtesë, gjë që lehtëson dhe shpejton mjaft programimin. Kur të klikohet butoni submit, forma do të dërgohet tek serveri dhe ai do ta bëjë validimin automatikisht. Pamja “run-time” e aplikacionit është paraqitur më poshtë në figurë.

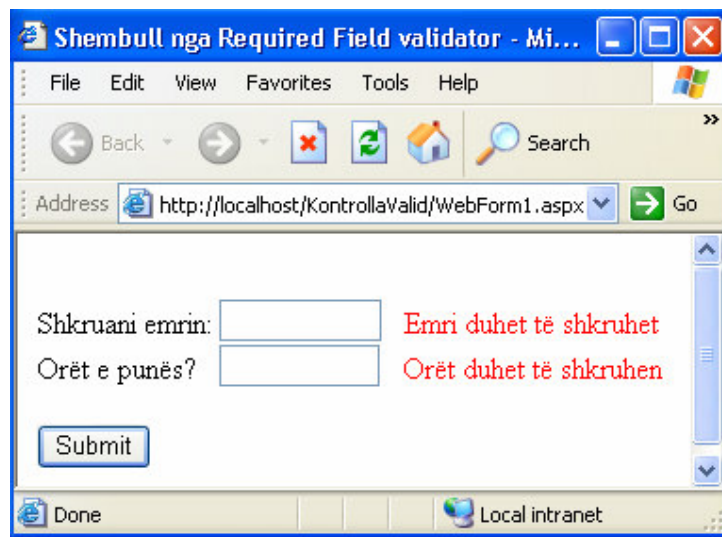


Figura 1.

Poashtu edhe kodi i aplikacionit është paraqitur më poshtë dhe shihet se është mjaft vetëpërshkrues.

```

<!-- Required Field Validator -->
<html><head</head>
<title>Shembull nga Required Field
validator</title><body>
<form runat="server"><br>Shkruani emrin:
<asp:TextBox id="txtName" rows="1 " width="50"
runat="server"/>
<asp:RequiredFieldValidator id="validTxtName"
runat="server" controlToValidate="txtName"
errorMessage="Emri duhet të shkruhet" display="static">
</asp:RequiredFieldValidator></br>
Orët e punës?
<asp:TextBox id="txtH" width="30" runat="server"/>
<asp:RequiredFieldValidator id="validTxtH" runat="server"
controlToValidate="txtH" errorMessage="Orët duhet të
shkruhen"
display="static">
</asp:RequiredFieldValidator></br>
<asp:Button id="btnSubmit" runat="server" text="Submit"
/>
</form></body></html>

```

## Kontrolla *RegularExpressionValidator*

Kontrolla *RegularExpressionValidator* përdoret për përputhje me ndonjë model.

Si shembull, supozojmë se vlera e fushës orë-pune duhet ti ketë një deri në tri shifra. Në këtë rast do ta vendosim kontrollën *RegularExpressionValidator* pas *txtH* kontrollës, dhe në atributin *RegularExpression* do ta specifikojmë mostren `/d{1,3}`. Kjo do të detyron sistemin që ta paraqes një mesazh gabimi nëse hyrja është më shumë se 3 shifra. Kodi i këtij aplikacioni është paraqitur në vijim:

```

<%@ Page Language="VB" Debug="true" %>
<html><head</head><body>
<form runat="server"><br>
Shkruaj emrin:
<asp:TextBox id="txtName" rows="1 " width="60"
runat="server"/>
<asp:RequiredFieldValidator id="validTxtName"
runat="server"
controlToValidate="txtName" errorMessage="Emri duhet të
shkruhet" display="static">
</asp:RequiredFieldValidator></br>
Orët e punës?
<asp:TextBox id="txtH" width="40" runat="server" />
<asp:RegularExpressionValidator id="validTxtH" runat="server"
controlToValidate="txtH" errorMessage="Orët duhet të
shkruhen" display="static">
</asp:RegularExpressionValidator>
<asp:RegularExpressionValidator id="regvH"
runat="server" display="static" controlToValidate="txtH"
errorMessage="orët duhet të jenë mes 1-3 numrash"
validationExpression="\d{1,3}">

```

```

</asp:RegularExpressionValidator></br>
<asp:Button id="btnSubmit" runat="server" text="Submit"
/>
</form></body></html>

```

## Kontrolla *CompareValidator*

Kontrolla **CompareValidator** krahason një hyrje me një vlerë të specifikuar ose me vlerën e një kontrole tjetër. Mundet gjithashtu të përdoret për të kontrolluar se e dhëna hyrëse është e ndonjë tipi të veçantë të të dhënave. Në shembullin vijues do të shtojmë edhe një textbox me emrin txtR, ku shfrytëzuesi do ta shënojë çmimin për orë. Dhe të supozojmë se dëshirojmë që tipi i të dhënave të kësaj fushe të jetë double, ku këtë e bëjmë me kontrollën validuese *CompareValidator*. Nëse vlera e fushës është double, nuk do të na paraqitet mesazhi i gabimit, por në të kundërtën do të paraqitet. Pamja e aplikacionit në kohën e ekzekutimit është treguar në figurën 2.

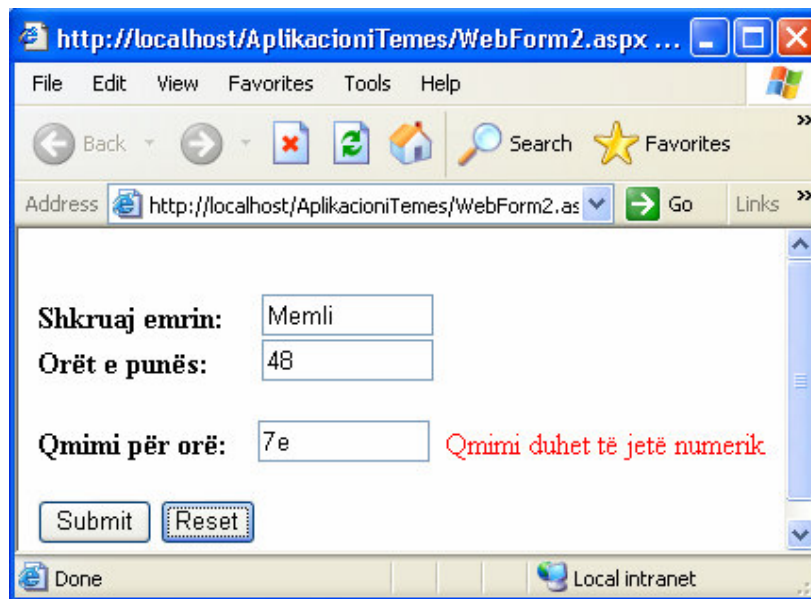


Figura 2.

Kodi i këtij aplikacioni është i njëjtë me kodin e katërt por vetëm duhet ti shtohet edhe kjo pjesë e kodit

```

<asp:CompareValidator id="comvR" runat="server"
display="static"
controlToValidate="txtR" errorMessage="Rate must be
numeric"
type="Double" operator="DataTypeCheck">
</asp:CompareValidator></br>

```

## Kontrolla *RangeValidator*

Kontrolla **RangeValidator** kontrollon nëse shënimi në kontrollën hyrëse është brenda një rangu të specifikuar. Supozojmë se e përdorim një textboks që

paraqet numrin e anëtarëve të familjes dhe ne dëshirojmë që vlera e kësaj fushe të jetë brenda një rangu të caktuar p.sh. 0 deri 10. Në këtë rast e kemi përdorur atributet *minValue* dhe *maxValue*, që i përkasin kësaj kontrolli, për të përcaktuar intervalin e vlerës së fushës. Në figurën 3 është paraqitur rezultati i aplikacionit, ndërsa më poshtë është paraqitur kodi shtesë.

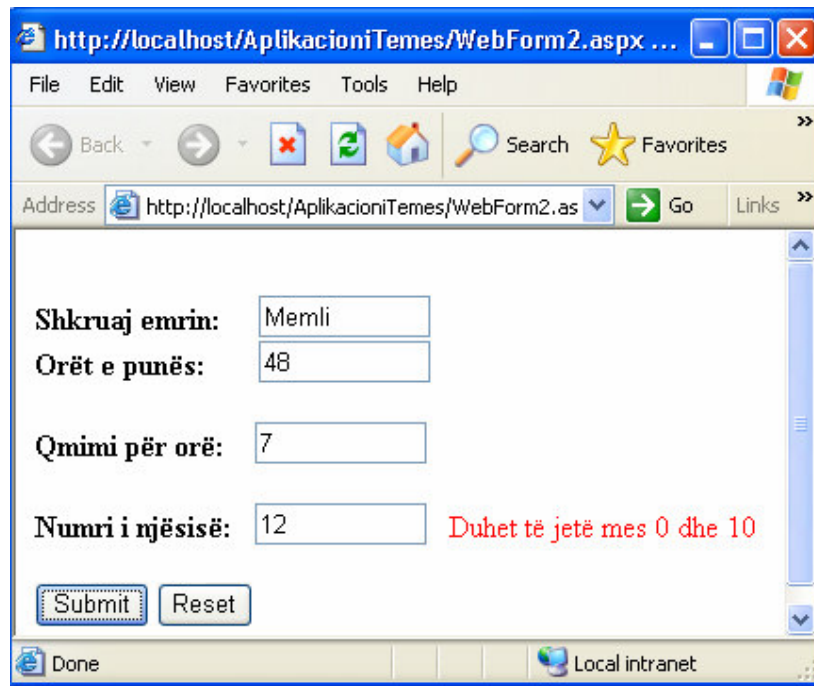


Figura 3

```
<asp:RangeValidator id="ranvDependents" runat="server"
display="static" controlToValidate="txtDependents"
errorMessage="Must be from 0 to 10"
type="Integer" minimumValue=0 maximumValue=10>
</asp:RangeValidator></br>
```

## Kontrolla *CustomValidator*

Në raste më komplekse, nuk mund të përdoren disa kontrollat validuese për validim por përdoret kontrollat **CustomValidator**. Kur përdoret kjo kontrollë, shfrytëzuesi mund të vendos funksionet e veta me qëllim që të kryej ndonjë detyrë të caktuar. Supozojmë që shfrytëzuesi do të vendos të dhëna në fushën e numrit të departamentit dhe ky numër duhet të jetë i plotpjestueshëm me 10. Me anë të kontrollës **CustomValidator** detyrohet sistemi që të paraqes një mesazh gabimi nëse vlera e dhënë nuk është e pranueshme. Kodi shtesë, në krahasim me kodin e mësipërm të aplikacionit është paraqitur më poshtë:

```
Cili është numri i departamentit tuaj?
<asp:TextBox id="txtDeptNum" width="40" runat="server"
/>
<asp:CustomValidator id="cusvDeptNum" runat="server"
display="static" controlToValidate="txtDeptNum"
onServerValidate="validateDeptNum"
errorMessage="Duhet të jetë i plotpjestueshëm me 10" >
</asp:CustomValidator></br>
<asp:Button id="btnSubmit" runat="server" text="Submit"
/>
</form></body></html>
<script language="VB" runat="server">
Sub validateDeptNum(source As Object, s as
ServerValidateEventArgs)
If (CInt(s.Value) Mod 10)=0 Then
s.IsValid= True
Else
s.IsValid=False
End If
End Sub
End Sub
</script>
```

## Kontrolla *ValidationSummary*

Kjo kontrollë mundëson që të paraqiten të gjitha gabimet në një lokacion të caktuar. Kodi për shtimin e kësaj kontrollë është paraqitur në vijim, ndërsa rezultati i aplikacionit është paraqitur në figurën 4.

```
<asp:ValidationSummary id="valSummary" runat="server"
headerText="Please correct the following errors"
display="static" showSummary="True"/>
```

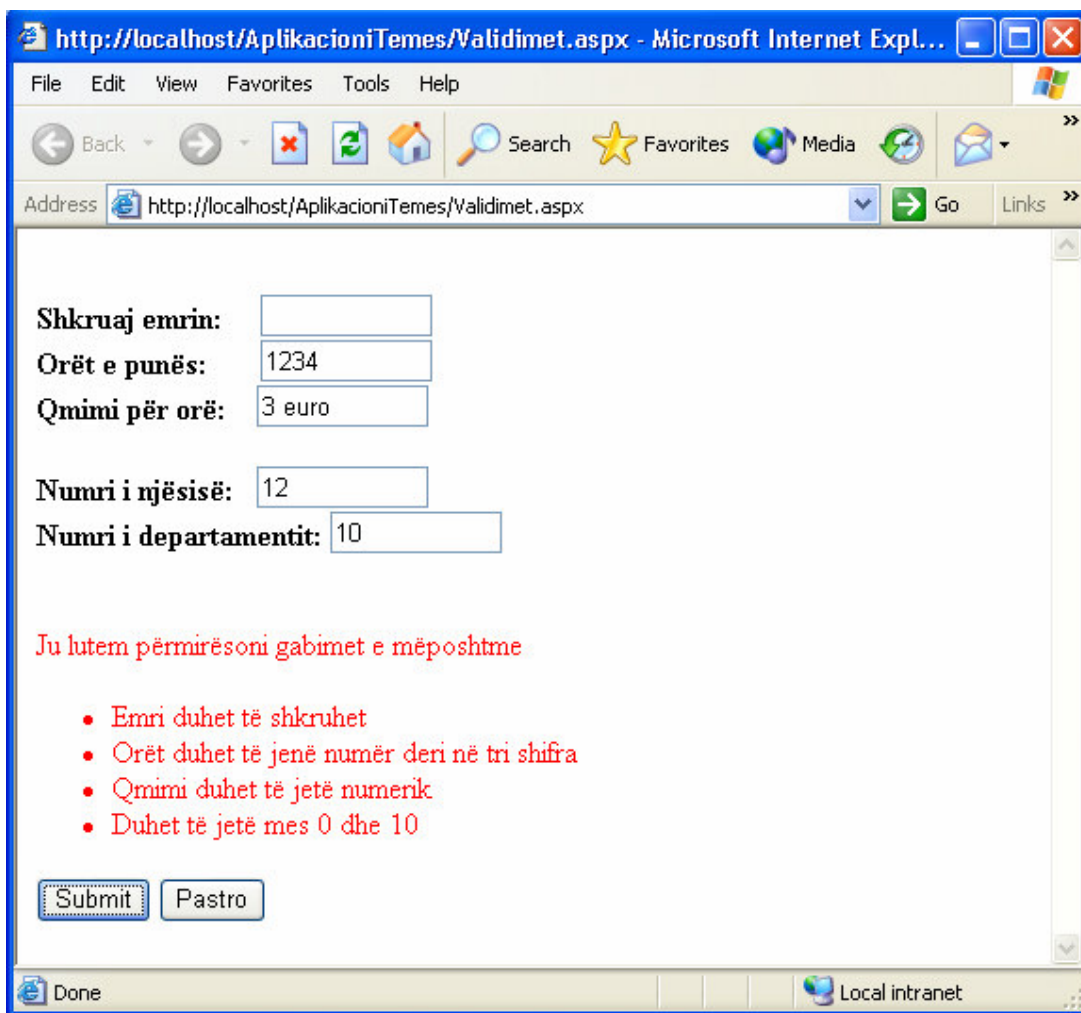


Figura 4.

## Custom Kontrollat

**Custom Kontrollat** u mundësojnë programerëve të krijojnë kontrollat të reja sipas funksionalitetit që atyre ju nevojiten. Këto **Custom kontrollat** mund të realizohen nga fillimi, të realizohen duke trashëguar klasë të kontrollave të caktuara dhe pastaj të modifikohen për t’iu përshtatur nevojave të programerit ose ato mund të krijojnë edhe duke gërshetuar disa kontrollat ekzistuese brenda një **Custom kontrollat** të re. Këto **Custom kontrollat** do të mund të përdoren kudo brenda web formës apo aplikacionit. **Custom kontrollat** mund të vendosen në dritaren **Toolbox** dhe prej aty të përdoren njëjloj sikur të gjitha kontrollat tjera ekzistuese.

Përveq krijimit të “user” kontrollave të cilat në të vërtetë paraqesin web faqe të vogla të ripërdorshme, mund të krijojnë edhe **Custom kontrollat** të kompajluara. Siq cekëm më lartë **Custom kontrollat** mund të krijojnë në tri mënyra:

- Krijimi i një **Custom kontrollat** të derivuar e cila trashëgon një kontrollat ekzistuese
- Krijimi i një kontrollat të përbërë (ang. **Composite Control**) duke grumbulluar kontrollat ekzistuese në një kontrollat të kompajluar të përbashkët
- Krijimi i një **Custom kontrollat** të përbërë duke e trashëguar nga *namespace*-i **Web.UI.WebControl**.

Kontrollat e përbëra janë të afërta me **user kontrollat**. Dallimi kyç ndërmjet tyre qëndron në atë se kontrollat e përbëra kompajlohen në fajllat me prapashtesën *dll* dhe pastaj përdoren njëjloj siq përdoren të gjitha server kontrollat.

Në vijim do të krijojmë një Librari të Web Kontrollave (ang. **Web Control Library**) në të cilën do të krijojmë **Custom kontrollat** që do të zbatohen gjatë këtij punimi. Procedura e krijimit të kësaj librerie është dhënë në vijim: Hapim programin **Visual Studio .NET** dhe në dritaren **File** zgjedhim opcionin **New Project**. Në dritaren **New Project Window**, zgjedhim opcionin **Visual Basic Projects** dhe në dritaren **Templates** zgjedhim opcionin **Web Control Library** të cilën do të emërojmë **Custom\_Kontrollat**

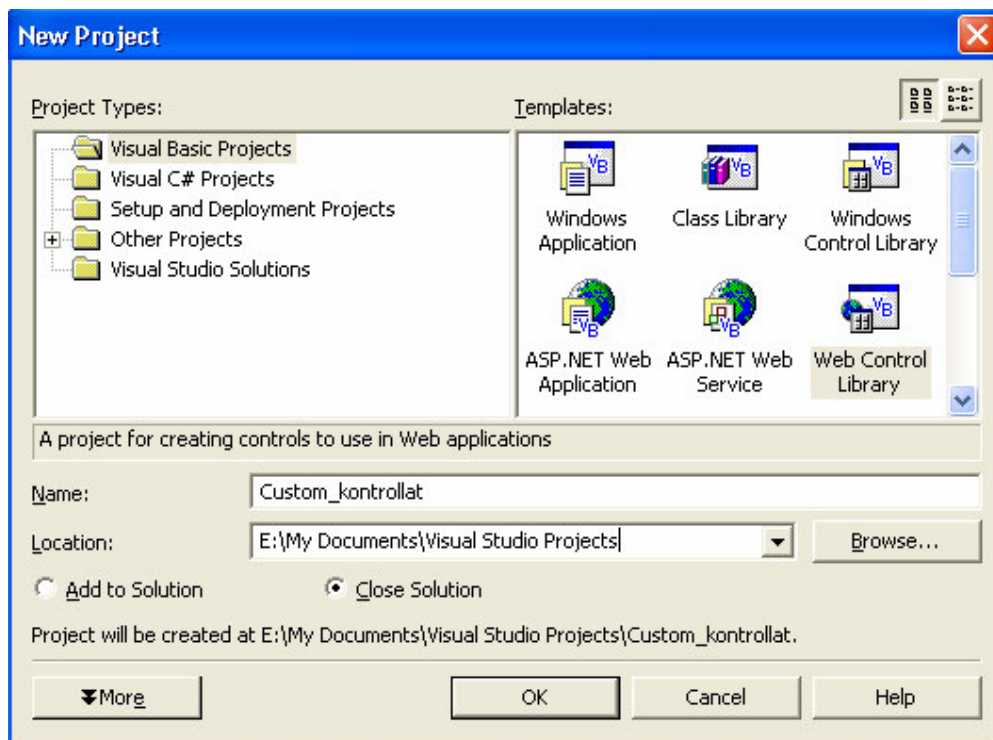


Figura 1.

Pas krijimit të projektit do të vërehet që **Visual Studio** vetvetiu ka krijuar një **Custom kontrollë** të kompletuar të emëruar **WebCustomControl**. Para se të shikojmë këtë kontrollë të krijuar ne do të krijojmë një Web Aplikacion për ta testuar atë.

Në menynë **File** zgjedhim sërish opcionin **New Project** dhe krijojmë një Web Aplikacion të ri duke klikuar në opcionin **ASP .NET Web Application** të cilin do ta emërojmë **Web\_Faqja\_e\_Custom\_Kontrollave**. Para se të mbarojmë me krijimin e projektit të ri duhet të zgjedhim opcionin **Add to Solution** në mënyrë që të dy këto projekte të përfshihen brenda një Solucioni.

Gjatë këtij punimi do të krijojmë disa **Custom kontrollera** të cilat do t'i testojmë nga ky aplikacion. Zakonisht gjatë kompajlimit të **Custom kontrollave** kopjojmë fajllin dll në direktoriumin *bin* i faqes ku do të testohet kontrolla. Mënyrë tjetër është që si shteg dalës (ang. output path) të caktojmë *bin* direktoriumin e faqes testuese, në këtë rast s'kemi nevojë të kopjojmë dll fajllin dhe do të mundemi të testojmë kontrollën më shpejtë. Procedura e caktimit të shtegut dalës është dhënë në vijim:

Vendosim kursorin mbi ikonën e projektit **CustomControls** dhe klikojmë tastin e djathtë të miut me ç'rast do të hapet dritarja e treguar në figurën e mëposhtme. Pastaj klikojmë në opcionin **Properties**, me ç'rast do të hapet dritarja e dhënë në figurën 3.

Zgjedhim opcionin **Configuration Properties** dhe në textboks-in me mbishkrimin **Output Path** gjejmë (përmes butonit **Browse**) direktoriumin e faqes testuese, siç është treguar në figurën 4.

Të tri figurat janë dhënë më poshtë.

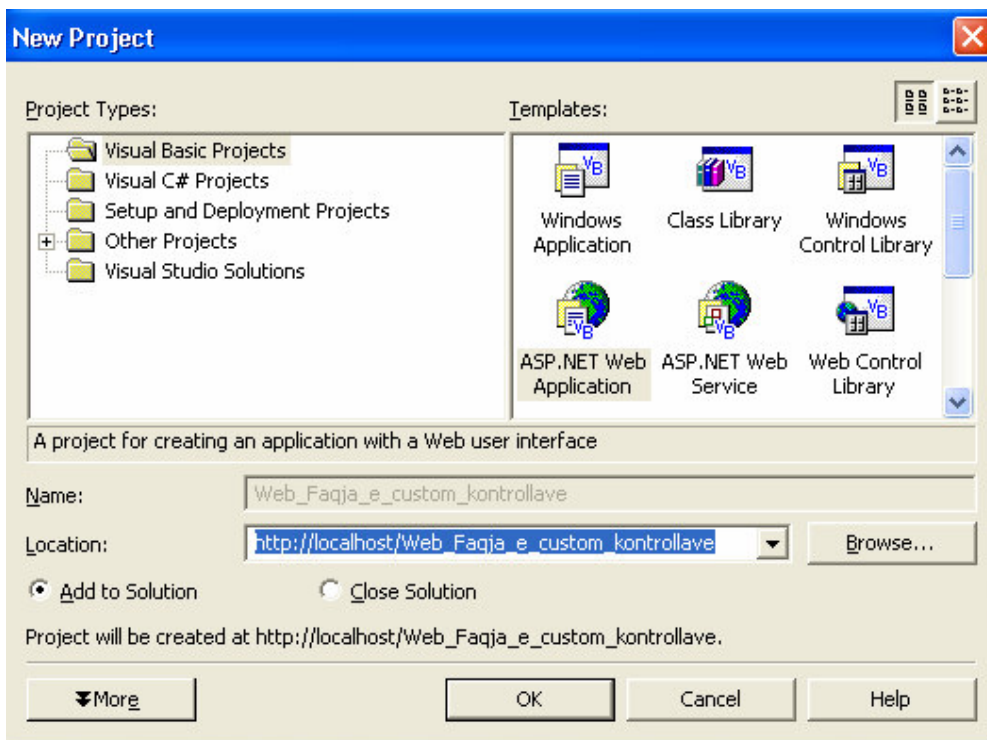


Figura 2.



Figura 3.

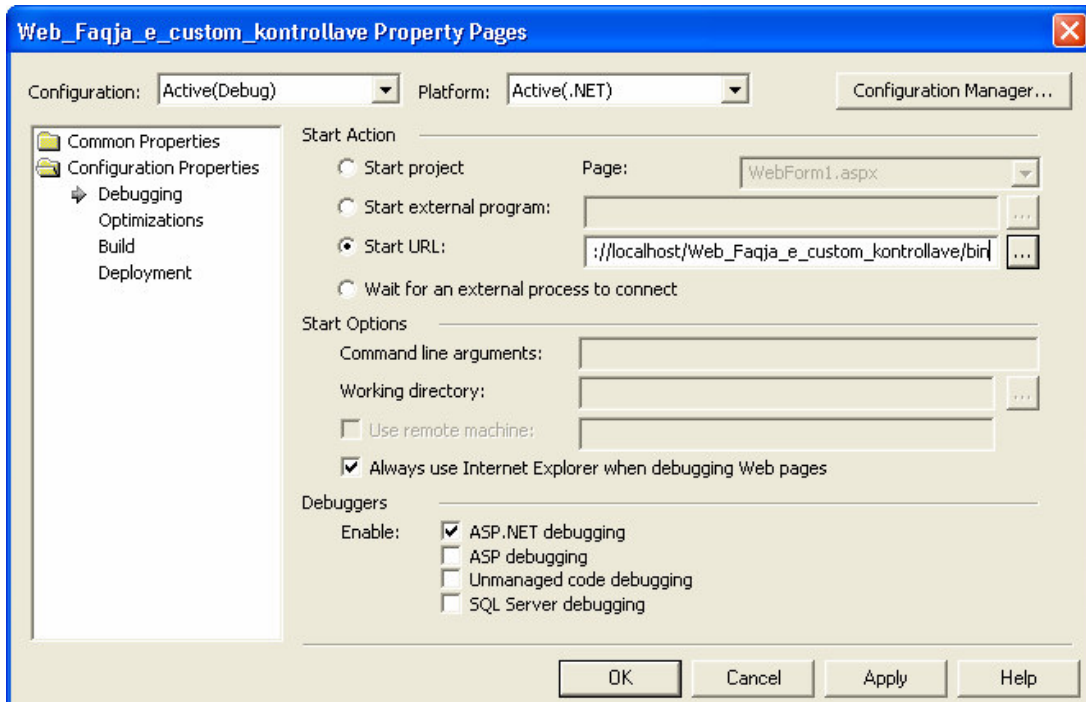


Figura 4.

## Custom Kontrolla ( e nënkuptuar ) e plotë

Siç pamë më lart gjatë krijimit të Librisë me **Web Kontrolla**, Visual Studio krijon një **Custom Kontrollë** të emëruar **WebCustomControl**. Kjo është një **Custom Kontrollë** e plotë, e trashëguar nga klasa **System.Web.UI.WebControls.WebControl**.

Ne mund të testojmë këtë kontrollë në web faqen testuese që krijuam, si në vijim: Hapim fajllin **WebForm1.aspx** në projektin **Custom\_Kontrolla\_e\_Plote\_WebFaqja** dhe shtojmë kodin për të regjistruar kontrollën e re:

```
<%@ Register TagPrefix="CustomKontrolla"
Namespace="Custom_Kontrolla_e_Plote"
Assembly="Custom_Kontrolla_e_Plote" %>
```

Ky kod regjistron **Custom kontrollën** në web faqe përmes tagut **@register** dhe dhënies së prefiksit të tagut (**CustomKontrolla**) si dhe tagjeve **Namespace** dhe **Assembly** të cilat identifikojnë kontrollën dhe fajllin dll që kjo faqe duhet të përdorë.

Tash faqes testuese mund t'i shtojmë kontrollën. Kontrollës duhet t'i caktojmë edhe atributin **Runat** që nevojitet për të gjitha server kontrollat dhe atributin **Text** i cili përmban tekstin që kontrolla do të shfaq gjatë ekzekutimit. Kontrollës së shtuar duhet t'i caktojmë edhe atributin **Id**. Tagu do të duhej të duhej si në vijim:

```
<CustomKontrolla:WebCustomControl1 Runat="Server" Text="Tung!"
id="WC1"/>
```

Pas ekzekutimit të programit ose pas hapjes së url-së [http://localhost/Custom\\_Kontrolla\\_e\\_Plote\\_WebFaqja/webform1.aspx](http://localhost/Custom_Kontrolla_e_Plote_WebFaqja/webform1.aspx) në web faqe do të paraqitet teksti “*Tung*” të cilin më herët ja caktuam atributit **Text**.

I tërë kodi i **Custom kontrollës** së plotë të gjeneruar nga **Visual Studio** është dhënë në vijim

```
Imports System.ComponentModel
Imports System.Web.UI
Imports System.Drawing

<DefaultProperty("Text"),
ToolboxData("<{0}: WebCustomControll1

runat=server></{0}:WebC
ustomControll1>")>

Public Class WebCustomControll1
    Inherits System.Web.UI.WebControls.WebControl

    Dim _text As String

    Public Sub WebCustomControll1()
        ViewState("Size") = "1"
    End Sub

    <Bindable(True), Category("Appearance"),
DefaultValue("")> _
    Property [Text]() As String
        Get
            Return _text
        End Get
        Set(ByVal Value As String)
            _text = Value
        End Set
    End Property End Class
```

Siç shihet kjo kontrollë përmban vetëm atributin **Text** i cili si variabël të fushës së të dhënave përmban stringun text.

## Atributet e Custom kontrollave

Atributeve të **Custom Kontrollave** mund t’iu qasemi njëllë siç ju qasemi attributeve të çdo klase tjetër. Mund t’iu qasemi në mënyrë programore (brenda kodit) apo pas deklarimit të Custom kontrollës duke iu dhënë vlera konkrete attributeve të kontrollës siç vepruam edhe me atributin **Text** në shembullin e faqes testuese, me ç’rast i vendosem vlerën “*Tung!*”.

Pra atributit **Text** të kontrollës iu qasëm përmes atributit **Text** në web faqe. Me këtë rast ndërlihdja ndërmjet atributit Text të implementuar brenda kontrollës dhe atributit **Text** në web faqen testuese është e thjeshtë pasi që të dy atributet i takojnë

tipit string, sidoqoftë, në raste tjera ASP .NET do të kryej shndërrimin inteligjent të tipeve të ndryshme të attributeve. Për shembull, nëse gjatë realizimit të kontrollës, një atributi të saj i caktojmë tipin integer, atëherë vlera që ne i caktojmë atributit gjatë instancimit të objektit do të konvertohet nga ASP .NET në tipin e të dhënave integer.

## Metoda e Renderimit (ang. Render)

Metoda Render paraqet metodën kyçe brenda **Custom kontrollave**. Kjo metodë deklarohet brenda klasës bazë dhe duhet të shkilet (ang. override) në klasën e derivuar nëse duam që faqes ti mundësojmë kontrollin e renderimit.

Në shembullin e mëposhtëm **Custom kontrollës** do t'i shtojmë një atribut të ri të quajtur **Madhesia**, ndërkaq faqes testuese që krijuam më herët do t'i shtojmë një buton përmes së cilit do të rrisim madhësinë e tekstit duke shfrytëzuar atributin **Madhesia** të kontrollës. Gjatë këtij shembulli gjithashtu do të shfrytëzojmë objektin **HtmlTextWriter** përmes të cilit do shkruajmë në *browser* stringun e dhënë në atributin **Text**. Klasa **HtmlTextWriter** trashëgon klasën **TextWriter** dhe ofron mundësi të shumta të formatizimit të tekstit.

Për të rritur madhësinë e tekstit deklarojmë një atribut të ri të emëruar **Madhesia** brenda **Custom kontrollës**. Kodi brenda metodës për renderim duket si në vijim:

```
Protected Overrides Sub Render(ByVal output As
System.Web.UI.HtmlTextWriter)
    output.Write("<font size = " & Madhesia & ">" &
[Text] & "</font>")
End Sub
```

Ky atribut **Madhesia**, duhet të ruaj gjendjen e tij përmes direktivës **postback** e cila aktivizohet pasi të klikojmë mbi buton. Ky operacion kryhet me anë të leximit dhe regjistrimit në koleksionin **ViewState** që mirëmbahet nga objekti faqe (ang. page).

Brenda kodit të **Custom Kontrollës** definojmë atributin **Madhesia** si në vijim:

```
Public Property Madhesia() As Integer
    Get
        Return Convert.ToInt32(ViewState("Size"))
    End Get
    Set(ByVal Value As Integer)
        ViewState("Size") = Value.ToString()
    End Set
End Property
```

Metoda **Get** e atributit e merr vlerën që i caktohet atributit përmes **ViewState** atributit dhe atë e kthen në string, dhe pastaj këtë string e konverton në ekuivalentin e tij integer.

Metoda **Set** e atributit e merr stringun që përfaqëson madhësinë në atributin **ViewState** të kontrollës.

Për tu siguruar që në fillim se atributi **ViewState** do të përmbaj një vlerë të arsyeshme, thirret konstruktori përmes të cilit inicializohet atributi **ViewState** me një vlerë të arsyeshme

```
Public Sub New()  
    ViewState("Size") = "1"  
End Sub
```

Konstruktori i mësipërm e inicializon vlerën që ruhet në atributin **ViewState** në vlerën 1. Më pastaj, siç do të tregohet në vijim çdo klikim i butonit do të azhuroj atributin **Madhësia** (Size).

Në vijim do të deklarojmë edhe një buton në faqen testuese përmes të cilit do të ekzekutojmë kodin brenda kontrollës.

```
<asp:Button Runat = "Server" Text="Zmadho shkronjat" OnClick="Button1_Click"  
id="Button1" />
```

Ndër atributet e butonit kemi deklaruar edhe një trajtues të ngjarjes që do të thirret me rastin e klikimit të butonit. Kodi për trajtuesin e ngjarjes vendoset në fajllin me prapashtesë *.aspx.vb* të quajtur “code-behind page” i cili do të hapet nëse klikojmë dy herë mbi kontrollën e butonit në pjesën dizajnuese të faqes testuese.

Web faqes testuese duhet ti shtojmë edhe një referim për në fajllin dll të projektit **CustomControls** në mënyrë që të pranoj objektin e instancuar. Fajlli dll krijohet gjithmonë pasi të jetë kompajluar projekti dhe gjendet në direktoriumin `\bin`. Referimin e shtojmë ashtu që në dritaren **Solution Explorer** pozicionohemi mbi ikonën **References** dhe klikojmë në tastin e djathtë të miut, më ç’rast do të hapet dritarja **Add Reference**.



Në këtë dritare zgjedhim opcionin **Projects** dhe pastaj përmes butonit **Browse** gjejmë shtegun deri tek `/bin` direktoriumi i projektit **Custom Controls** ku gjendet fajlli dll. Kodi në *.aspx.vb* fajllin do të do të duket si në vijim

```
Imports Custom_Kontrolla_e_Plote.WebCustomControl1  
Public Class WebForm1  
    Inherits System.Web.UI.Page  
    Protected WithEvents Button1 As  
System.Web.UI.WebControls.Button  
    Protected WC1 As  
Custom_Kontrolla_e_Plote.WebCustomControl1  
  
+ Web Form Designer Generated Code  
  
    Public Sub Button1_Click(ByVal sender As Object, ByVal e As  
System.EventArgs)_
```

```
Handles Button1.Click
    WC1.Madhësia += 1
End Sub
End Class
```

Në kodin e mësipërm kemi krijuar një instancë të re të **Custom kontrollës WebCustomControl1** nga projekti **Custom\_Kontrolla\_e\_Plote**. Instancën e krijuar e kemi emëruar **WC1**. Më poshtë kemi deklaruar trajtuesin e ngjarjes me rastin e klikimit të butonit **Button1**, me ç'rast atributi **Madhësia** i kësaj kontrollë do të rritet për një.

Pas ekzekutimit të projektit do të vërejmë se çdo herë kur klikohet butoni do të rritet për një vlera e variablës së gjendjes **Size** e me këtë edhe madhësia e tekstit.

Për të ekzekutuar shembullin në **InternetExplorer** japim shtegun:

*http://localhost/Custom\_Kontrolla\_e\_Plote\_WebFaqja/webform1.aspx*

Pasi të klikojmë 3 herë në butonin me tekst Zmadho Shkronjat madhësia e shkronjave do të rritet nga vlera fillestare 1 në vlerën 4, pas çdo klikimi madhësia e shkronjave të tekstit "Tung!" do të rritet për një. Pas tri klikimeve madhësia e tekstit në browser do të duket si më poshtë:



Figura 5.

## Krijimi i Custom kontrollave të derivuara

Ka raste kur nuk është e nevojshme të krijojmë një kontrollë nga e para. Ne mund të zgjedhim vetitë dhe metodat e një kontrollë ekzistuese. Kontrollat ekzistuese mund të trashëgohen në të njëjtën mënyrë siç trashëgohen klasët.

Si shembull, mund të kodojmë një buton që të tregoj se sa herë është shtypur (klikuar). Ka aplikacione të caktuara ku një buton i tillë do të jetë i dobishëm. Web kontrollat **Button** nuk e ka të implementuar këtë mundësi.

Për të tejkalluar këtë kufizim të klasës së butonit, ne do të trashëgojmë një **Custom kontrollë** të re nga klasa **System.Web.UI.WebControls.Button**, siç është treguar në kodin e mëposhtëm:

```

Imports System.ComponentModel
Imports System.Web.UI
Imports System.Web.UI.WebControls

' Custom kontrollë trashëgon kontrollën button
Public Class ButoniNumrues
    Inherits System.Web.UI.WebControls.Button

    ' Ne konstruktor inicializojme vlerat fillestare
    Public Sub New()
        Me.Text = "Më kliko"
        ViewState("Numrimi") = 0
    End Sub

    ' numrimimi si atribut që ruhet atributin ViewState
    Public Property Numrimi() As Integer
        Get
            Return CInt(ViewState("Count"))
        End Get
        Set(ByVal Value As Integer)
            ViewState("Count") = Value
        End Set
    End Property

    ' e shkelim metoden OnClick per te inkrementuar numerimin,
    ' e azhurojme tekstin e butonit dhe pastaj therrasim
    OnClick metoden e klases baze

    Protected Overrides Sub OnClick(ByVal e As EventArgs)
        ViewState("Count") = CInt(ViewState("Count")) + 1
        Me.Text = ViewState("Count") & " clicks"
        MyBase.OnClick(e)
    End Sub
End Class

```

Në fillim e trashëgojmë klasën e re *ButoniNumrues* nga klasa ekzistuese **Button**

```

Public Class ButoniNumrues
    Inherits System.Web.UI.WebControls.Button

```

Kjo klasë e krijuar e ka për detyrë të ruaj gjendjen e saj: të tregoj sa herë është klikuar butoni. Brenda klasës do të deklarojmë atributin **Numrimi** i cili si variabël interne do të ketë vlerën e ruajtur në atributin **ViewState**. Përdorimi i atributit **ViewState** është i nevojshëm sepse butoni do ta dërgoj (ang. post) web faqen dhe po të mos përdorej atributi **ViewState** gjendja paraprake do të humbej. Definimi i atributit **Numrimi** është dhënë në vijim:

```

Public Property Numrimi() As Integer
    Get
        Return CInt(ViewState("Count"))
    End Get
    Set(ByVal Value As Integer)
        ViewState("Count") = Value

```

```
End Set
End Property
```

Për të nxjerrë vlerën "Count" nga atributi **ViewState**, e përdorim stringun **Count** si një kompenzim në koleksionin **ViewState**. Ajo që kthehet paraqet një objekt të cilin mund ta kthejmë në integer.

Për tu siguruar që atributi Numrimi do të kthej një vlerë të arsyeshme atë e inicializojmë brenda një konstruktorit. Brenda konstruktorit do të inicializojmë edhe tekstin fillestar që do të përmbaj butoni. Konstruktori do të deklarohet si në vijim:

```
Public Sub New()
Me.Text = "Më kliko"
ViewState("Numërimi") = 0
End Sub
```

Pasi që **ButoniNumrues** trashëgon klasën **Button** nuk është vështirë të shkelim (ang. override) implementimin e ngjarjes **Click**. Në këtë shembull, kur shfrytëzuesi klikon butonin, duhet të rrisim vlerën **Count** të ruajtur në atributin **ViewState** si dhe të azhuroj tekstin e mbishkruar në buton ashtu që të tregoj numrin e klikimeve. Pastaj thërrasim metodën **OnClick** të klasës bazë për të kryer procesimin e zakonshëm të ngjarjes **Click**.

Kodi burimor për trajtuesin e ngjarjes **Click** është dhënë në vijim:

```
Protected Overrides Sub OnClick(ByVal e As EventArgs)
    ViewState("Count") = CInt(ViewState("Count")) + 1
    Me.Text = ViewState("Count") & " clicks"
    MyBase.OnClick(e)
End Sub
End Class
```

Në fund HTML kodit të formës .aspx ia shtojmë këtë kontrollë të krijuar siç do t'i shtonim kontrollat e përbëra:

```
<CustomKontrolla:ButoniNumrues Runat="Server" id="CB1" />
```

Sikur edhe tek shembulli i mëparshëm edhe këtu duhet shtuar deklarinimin me tagun Register.

```
<%@Register TagPrefix="CustomKontrolla"
Namespace="Custom_Kontrolla_e_Trasheguar"
Assembly="Custom_Kontrolla_e_Trasheguar" %>
```

Për të ekzekutuar shembullin në **InternetExplorer** japim shtegun:

[http://localhost/Custom\\_Kontrolla\\_e\\_Trasheguar\\_WebFaqja/webform1.aspx](http://localhost/Custom_Kontrolla_e_Trasheguar_WebFaqja/webform1.aspx)

Pasi të klikojmë 5 herë në butonin e paraqitur, pamja në browser do të duket si më poshtë:



Figura 6.

## Krijimi i Custom Kontrollave të përbëra

Mënyra e tretë e krijimit të Custom kontrollave është me anë të kombinimit të dy apo më shumë kontrollave ekzistuese. Në shembullin e mëposhtëm do të formojmë një kontrollë pak më komplekse e cila do të mund të përdoret për të regjistruar numrin e kërkesave për libra të caktuara në ndonjë librari.

Kontrolla përmes së cilës zgjedhim një apo më shumë libra dhe sa herë që klikojmë në një libër kontrolla do të regjistroj vlerën se për të satën herë është klikuar ai libër.

Në HTML fajllin e faqes testuese do të shtojmë kodin e pastaj do ta komentojmë atë:

```
<%@ Register TagPrefix="CustomKontrolla"
Namespace="Custom_Kontrolla_e_Perbere"
Assembly="Custom_Kontrolla_e_Perbere"%>

<%@ Page Language="vb" AutoEventWireup="false"
Codebehind="WebForm1.aspx.vb"
Inherits="Custom_Kontrolla_e_Perbere_WebFaqja.WebForm1"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
    <meta name="GENERATOR" content="Microsoft Visual
Studio.NET 7.0">
    <meta name="CODE_LANGUAGE" content="Visual Basic 7.0">
    <meta name="vs_defaultClientScript" content="JavaScript">
    <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
  </HEAD>
  <body MS_POSITIONING="GridLayout">
    <form id="Form2" method="post" runat="server">

      <CustomKontrolla: Lista_e_Kerkesave_te_Librave Runat="Server"
id="bookInquiry1">
```

```

<CustomKontrolla:Numruesi_i_Librave
    Runat="server" Emri_i_Librit="Programming ASP.NET"
ID="Numruesi_i_Librave1"> </CustomKontrolla:Numruesi_i_Librave>

<CustomKontrolla:Numruesi_i_Librave
    Runat="server" Emri_i_Librit="Programming C#"
ID="Numruesi_i_Librave2"> </CustomKontrolla:Numruesi_i_Librave>

<CustomKontrolla:Numruesi_i_Librave
    Runat="server" Emri_i_Librit="Teach Yourself C++ 21 Days"
ID="Numruesi_i_Librave3">
</CustomKontrolla:Numruesi_i_Librave>

...

<CustomKontrolla:Numruesi_i_Librave
    Runat="server" Emri_i_Librit="XML Web Documents From
Srcatch"ID="Numruesi_i_Librave8">
</CustomKontrolla:Numruesi_i_Librave>

</CustomKontrolla:Lista_e_Kerkesave_te_Librave>
</form>
</body>
</HTML>

```

Siç shihet komponenta **Lista\_e\_Kerkesave\_te\_Librave** përmban një numër të elementeve **Numruesi\_i\_Librave** . Për secilin libër “tracking” të dhënë ekziston nga një element **Numruesi\_i\_Librave** . Kontrolla është mjaft fleksibile. Mund të ruajmë gjendjen e një deri në tetë librave (siç tregohet në shembull), ose çfarëdo numri tjetër librash. Secili element **Numruesi\_i\_Librave** ka atributin **Emri\_i\_Librit** i cili shërben për të shfaqur emrin e librit që ruhet.

Gjendja e secilit libër ruhet me anë të **Custom kontrollës Butoni\_Numrues**.

Deklarimi i **ButoniNumrues** nuk ndodhet brenda *.aspx* fajllit, ai është implementuar brenda **Custom kontrollës Numruesi\_i\_Librave**.

E tërë arkitektura është kontrollës është dhënë në vijim:

1. Custom kontrollja e përbërë **Lista\_e\_Kerkesave\_te\_Librave** trashëgon klasën **WebControl** dhe implementon **INamingContainer** siç do të tregohet më vonë.

2. Kontrollja **Lista\_e\_Kerkesave\_te\_Librave** ka atributin **Controls** të cilin e trashëgon nga klasa **Control** (përmes **WebControl**) i cili kthen një koleksion të kontrollave fëmijë.

3. Brenda koleksionit **Controls** ndodhen një numër i kontrollave **Numruesi\_i\_Librave**

4. Edhe kontrollja **Numruesi\_i\_Librave** është kontrollë e përbërë, ajo trashëgon klasën **WebControl** që ndodhet në *namespace*-in **System.Web.UI.WebControls**. Edhe kontrollja **Numruesi\_i\_Librave** implementon interfejsin **INamingContainer**.

- a) Çdo instancë e klasës **Numruesi\_i\_Librave** ka dy attribute, **Emri\_Librit** dhe **Numero**.
- b) Atributi **Emri\_Librit** implementohet përmes atributit **ViewState** dhe inicializohet përmes atributit **Emri\_i\_Librit** në fajllin *.aspx*.
- c) Atributi **Numrues\_i\_Librave** i delegohet objektit **ButoniNumrues**, i cili instancohet në **BookContainer.CreateChildControls()**.

Objekti **Lista\_e\_Kerkesave\_te\_Librave** ka dy qëllime: shërben si kontejner për objektet **Numrues\_i\_Librave** dhe përkujdeset që të renderoj veten dhe të siguroj që objektet **Numrues\_i\_Librave** që përfshihen brenda tij të renderohen sipas nevojës.

Për të kuptuar se si funksionon kodi do ta analizojmë pjesë pjesë. Objekti që më së shumti përsëritet është objekti **ButoniNumrues**. Këtë objekt e kemi implementuar edhe në shembullin për **Custom kontrollën** e trashëguar por tash kemi bërë disa ndryshime në të.

### ' Custom kontrolla trashëgohet nga klasa Button derives

```
Public Class ButoniNumrues
    Inherits System.Web.UI.WebControls.Button

    Private Stringu_i_shfaqur As String

    ' brenda konstruktorit inicializojmë një vlerë fillestare
    Public Sub New()
        Stringu_i_shfaqur = " klikime"
        Init()
    End Sub

    ' e mbingarkojmë konstruktorin duke i dhënë si parametër
    stringun që duhet shfaqur (psh., 5 libra)
    Public Sub New(ByVal Stringu_i_shfaqur As String)
        Me.Stringu_i_shfaqur = Stringu_i_shfaqur
        Init()
    End Sub

    ' Sub procedura e mëposhtme thirret brenda konstruktorëve
    për të inicializuar vlerat fillestare
    Private Shadows Sub Init()
        If ViewState("Count") Is Nothing Then
            ViewState("Count") = 0
            Me.Text = "Më Kliko"
        End If
    End Sub

    ' Atributi Numero që ruhet në atributin ViewState
    Public Property Numero() As Integer
        Get
            Return CInt(ViewState("Numero"))
        End Get
    End Property
End Class
```

```

        Set (ByVal Value As Integer)

        ViewState("Numero") = Value
        End Set
    End Property

    ' e shkelim metoden OnClick për të inkrementuar numrimin
    (count),
    ' e axhurojmë tekstin që shfaqet në buton dhe pastaj
    thërrasim metodën invoke të klasës bazë
    Protected Overrides Sub OnClick(ByVal e As EventArgs)
        ViewState("Numero") = CInt(ViewState("Numero")) + 1
        Me.Text = CStr(ViewState("Numero") & " " &
Stringu_i_shfaqur)
        MyBase.OnClick(e)
    End Sub
End Class

```

Pasi që në shembullin e tanishëm dëshirojmë që butoni të tregoj tekstin 5 kërkesa e jo 5 klikime, duhet të shkelim funksionin e ngjarjes OnClick që e shfaq tekstin e butonit.

```

Me.Text = CStr(ViewState("Numero") & " " & Stringu_i_shfaqur)

```

Në këtë rast kemi përdorur variablën private **Stringu\_i\_shfaqur** për të ruajtur vlerën fillestare që do të caktohet në konstruktor.

```

Private Stringu_i_shfaqur As String

```

Këtë string duhet ta inicializojmë brenda konstruktorit. Për ta mbrojtur kodin e klientit i cili veçse përdorë konstruktorin e nënkuptuar (pa parametra), ne do të mbingarkojmë konstruktorin, duke i dhënë një version që merr një string.

```

Public Sub New(ByVal Stringu_i_shfaqur As String)
    Me.Stringu_i_shfaqur = Stringu_i_shfaqur
    Init()
End Sub

```

Ju mund të modifikoni konstruktorin e nënkuptuar duke i dhënë varaiblës interne **Stringu\_i\_shfaqur** një vlerë tjetër fillestare.

```

Public Sub New()
    Stringu_i_shfaqur = " klikime"
    Init()
End Sub

```

Brenda konstruktorit kemi thirrur metodën Init ( ) e cila e inicializon atributin **Count** me vlerën 0 dhe vendos tekstin fillestar që do të shfaqet në buton.

```

Private Shadows Sub Init()
    If ViewState("Count") Is Nothing Then
        ViewState("Count") = 0
        Me.Text = "Click me"
    End If
End Sub

```

Pas këtyre modifikimeve, kontrollla **ButoniNumrues** është e gatshme të përdoret në kontrollën e parë të përbërë, **Numrues\_i\_Librave** .

## Krijimi i kontrollës së përbërë Numrues\_i\_Librave

Kontrolla e përbërë **Numrues\_i\_Librave** duhet të ruaj dhe të shfaq numrin e kërkesave të bëra një libri të caktuar. Më poshtë do të japim kodin e plotë të kësaj kontrole:

```
Public Class Numruesi_i_Librave
    Inherits System.Web.UI.WebControls.WebControl
    Implements INamingContainer

    ' e inicializojme anetarin ButoniNumrues
    Public btn As ButoniNumrues = New ButoniNumrues("kërkesa")

    Public Property Emri_i_Librit() As String
        Get
            Return CStr(ViewState("Emri_i_Librit"))
        End Get
        Set(ByVal Value As String)
            ViewState("Emri_i_Librit") = Value
        End Set
    End Property

    Public Property Numero() As Integer
        Get
            Return btn.Numero
        End Get
        Set(ByVal Value As Integer)
            btn.Numero = Value
        End Set
    End Property

    Public Sub Reset()
        btn.Numero = 0
    End Sub

    Protected Overrides Sub CreateChildControls()
        Controls.Add(btn)
    End Sub

End Class
```

## Interfejsi INamingContainer

Gjëja e parë që bie në sy tek klasa **Numrues\_i\_Librave** është implementimi i interfejsit **INamingContainer**. Ky është një interfejs i zbrazët (ang. Marker Interface) i cili nuk ka asnjë metodë. Ky interfejs ka për qëllim të identifikoj një kontejner kontrollë (ang. **ContainerControl**) e cila do të krijoj një *namespace* identifikues të ri, duke garantuar që atributi ID i të gjitha kontrollave fëmijë do të jetë i njëjtë për të gjitha kontrollat brenda aplikacionit.

Klasa **Numrues\_i\_Librave** përfshin një instancë të **ButoniNumrues** :

```
Public btn As CountedButton = New  
CountedButton("inquiries")
```

Anëtari **btn** është instancuar në metodën **CreateChildControls** të trashëguar nga klasa **System.Control**:

```
Protected Overrides Sub CreateChildControls()  
    Controls.Add(btn)  
End Sub
```

Metoda **CreateChildControls** thirret në prag të renderimit dhe klasës **Numrues\_i\_Librave** i ofron mundësinë që të shtojë objektin **btn** si një kontrollë të përmbajtur (ang. Contained).

Klasa **Numrues\_i\_Librave** nuk ka nevojë të shkelë metodën **Render**; ajo mjafton të renderoj klasën **ButoniNumrues**, i cili mund ta renderoj vetën. Sjellja e nënkuptuar e metodës **Render** është të renderoj të gjitha kontrollat fëmijë, andaj nuk kemi nevojë të shtojmë kod për renderimin e kontrollave fëmijë.

Klasa **Numrues\_i\_Librave** ka edhe dy attribute tjera: **Emri\_i\_Librit** dhe **Numero**. **Emri\_i\_Librit** paraqet stringun që do të shfaqet në kontrollë dhe menaxhohet përmes atributit **ViewState**. Kodi i këtij atributi është dhënë në vijim:

```
Public Property Emri_i_Librit() As String  
    Get  
        Return CStr(ViewState("Emri_i_Librit"))  
    End Get  
    Set(ByVal Value As String)  
        ViewState("Emri_i_Librit") = Value  
    End Set  
End Property
```

**Numero** paraqet numrimin e kërkesave të bëra për një libër të caktuar; ruajtja e kësaj vlere i lihet klasës **ButoniNumrues**. Kodi i këtij atributi është dhënë në vijim:

```
Public Property Numero() As Integer
```

```

    Get
        Return btn.Numero
    End Get
    Set(ByVal Value As Integer)
        btn.Numero = Value
    End Set
End Property

```

Nuk ka nevojë të vendosim ndonjë vlerë në atributin ViewState, pasi që vetë butoni përkujdeset për shfaqjen e tekstit që përmban.

## Krijimi i kontrollës së përbërë Lista\_e\_Kërkesave\_të\_Librave

Secila prej kontrollave **Numrues\_i\_Librave** është përfshirë brenda koleksionit të Kontrollave të **Lista\_e\_Kerkesave\_te\_Librave**. Kjo kontrollë nuk ka attribute e as gjendje. E vetmja metodë që përmban është metoda Render, siç është treguar më poshtë:

```

<ControlBuilder(GetType(Ndërtuesi_i_Numruesit_te_Librave)),
ParseChildren(False)> _
Public Class Lista_e_Kerkesave_te_Librave
    Inherits System.Web.UI.WebControls.WebControl
    Implements INamingContainer

    Protected Overrides Sub Render(ByVal output As
HtmlTextWriter)

        Dim gjithsej_Kerkesa As Integer = 0

        ' Formulojmë hederin e tabelës
        output.Write("<Table border='1' width='90%'
cellpadding='1'" & _
            "cellspacing='1' align = 'center' >")
        output.Write("<TR><TD colspan = '2' align='center'>")
        output.Write("<B> Kërkesat </B></TD></TR>")

        ' nëse nuk kemi asnjë kontroll të përfshirë (ang
        contained, paraqit mesazhin e nënkuptuar.
        If Controls.Count = 0 Then
            output.Write("<TR><TD colspan = '2'>
align='center'")

            output.Write("<B> Nuk ka libra në listë </B></TD></TR>")
            ' përndryshe, nëse ka kontrolla të përfshira (ang.
            contained controls) renderojë secilën prej tyre
        Else
            ' ciklo nëpër secilën prej kontrollave në
            koleksionin e kontrollave dhe
            ' shfaqe emrin e librit për secilën kontrollë
            ' pastaj thuaji secilës kontroll të renderoj veten

```

```

        Dim current As Numruesi_i_Librave
        Dim kontrolla_e_Tanishme As Numruesi_i_Librave

        For Each kontrolla_e_Tanishme In Controls
            gjithsej_Kerkesa +=
kontrolla_e_Tanishme.Numero
            output.Write("<TR><TD align='left'>" & _
                kontrolla_e_Tanishme.Emri_i_Librit +
" </TD>")

            output.RenderBeginTag("TD")
            kontrolla_e_Tanishme.RenderControl(output)
            output.RenderEndTag()                ' mbyllim
tagun <TD>

            output.Write("</TR>")
        Next
        Dim str_gjithsej_Kerkesa As String
        str_gjithsej_Kerkesa = gjithsej_Kerkesa.ToString

        output.Write("<TR><TD colspan='2' align='center'> " & _
            " Gjithsej Kërkesa: " & _
            CStr(str_gjithsej_Kerkesa) & "</TD></TR>")
        End If
        output.Write("</TABLE>")
    End Sub

End Class

```

## Atributet ControlBuilder dhe ParseChildren

Klasa **Numrues\_i\_Librave** ndërlidhet me klasën **Lista\_e\_Kerkesave\_te\_Librave** në mënyrë që ASP .NET të përkthej elementet e fajllit *.aspx* në kodin përkatës.

Kjo arrihet përmes atributit **ControlBuilder**:

```
<ControlBuilder(GetType(Ndërtuesi_i_Numruesit_te_Librave)),
ParseChildren(False)>
```

Argumenti i atributit **ControlBuilder** është një objekt **Type** të cilin e marrim përmes klasës në **Ndërtuesi\_i\_Numruesit\_te\_Librave**, klasë kjo që do ta definojmë për të kthyer tipin e klasës **Numrues\_i\_Librave** përmes dhënies së tagut të emëruar

**BookCounter** në *.aspx* fajllin. Kodi për **Ndërtuesi\_i\_Numruesit\_te\_Librave** është treguar më poshtë:

```
Friend Class Ndërtuesi_i_Numruesit_te_Librave
    Inherits ControlBuilder

```

```

    Public Overrides Function GetChildControlType( _
        ByVal emri_i_Tagut As String, ByVal attributes As
IDictionary) As Type
        If emri_i_Tagut = "Numruesi_i_Librave" Then
            Dim x As Numruesi_i_Librave
            Return x.GetType
        Else
            Return Nothing
        End If
    End Function

    Public Overrides Sub AppendLiteralString(ByVal s As
String)
    End Sub

End Class

```

ASP.NET do ta përdorë këtë **Ndërtuesi\_i\_Numruesit\_te\_Librave**, i cili trashëgon klasën **Numrues\_i\_Librave**, për të përcaktuar tipin e objektit që tregon tagu **Numruesi\_i\_Librave**. Përmes kësaj ndërlidhjeje, secili prej objekteve **Numruesi\_i\_Librave** do të instancohet dhe do t'i shtohet koleksionit kontrollave të klasës **Lista\_e\_Kerkesave\_te\_Librave**.

Atributit të dytë, **ParseChildren** duhet ti caktohet vlera *false* për t'i bërë me dije ASP.NET-it që i kemi trajtuar atributet fëmijë dhe se nuk nevojitet trajtim i mëtejshëm. Vlera *false* tregon që atributet fëmijë të përfshira nuk janë attribute të objektit të jashtëm, por kontrollat fëmijë.

## Renderimi

Metoda e vetme që **Lista\_e\_Kerkesave\_te\_Librave** përmban është metoda e cila shkelë funksionin **Render**. Kjo metodë ka për qëllim vizatimin e tabelës, që si rezultat do të paraqitet në browser, me anë të dhënave të menaxhuara nga secila prej kontrollave fëmijë **Numruesi\_i\_Librave**.

**Lista\_e\_Kerkesave\_te\_Librave** ofron numërimin e të gjitha kërkesave të bëra. Kodi numëron kërkesat duke inicializuar një variabël integer, **gjithsej\_Kerkesa** në vlerën 0 dhe pastaj përsëritet nëpër secilën kontrollë me rradhë, duke i kërkuar secilës kontrollë atributin **Numero** siç tregohet më poshtë:

```
gjithsej_Kerkesa += current.Numero
```

Atributi **Numero** i kontrollës i delegohet atributit **Numero** të **ButoniNumrues**, gjë që mund të vërehet gjatë debugimit të kodit. I njëjti kod renderon secilën prej kontrollave fëmijë duke përsëritur të njëjtin cikël nëpër secilën prej kontrollave.

```
For Each kontrollat_e_Tanishme In Controls
```

```

        gjithsej_Kerkesa +=
kontrolla_e_Tanishme.Numero
        output.Write("<TR><TD align='left'>" & _
            kontrolla_e_Tanishme.Emri_i_Librit +
"</TD>")
        output.RenderBeginTag("TD")
        kontrolla_e_Tanishme.RenderControl(output)
        output.RenderEndTag()           ' mbyllim tagun
<TD>
        output.Write("</TR>")

```

Objekti lokal i **Numrues\_i\_Librave**, **kontrolla\_e\_Tanishme**, i jepet secilit objekt në koleksionin **Controls** me rradhë:

```
For Each kontrolla_e_Tanishme In Controls
```

Me këtë objekt mund të marrim vlerën e atributit **Numero**, siç u sqarua më parë

```
gjithsej_Kerkesa += kontrolla_e_Tanishme.Numero
```

dhe pastaj vazhdohet me renderimin e objektit. **HtmlTextWriter** përdoret së pari për të krijuar një rresht dhe për të shfaqur emrin e librit, përmes atributit **Emri\_i\_Librit** të objektit të tanishëm (**kontrolla\_e\_Tanishme**) të **Numrues\_i\_Librave**

```

        output.Write("<TR><TD align='left'>" & _
            kontrolla_e_Tanishme.Emri_i_Librit + "</TD>")

```

Pastaj renderojme një tag TD, dhe brenda këtij tagu ne i tregojmë objektit **Numrues\_i\_Librave** të renderoj veten. Më në fund e renderojmë një tag TD mbyllës duke përdorur shprehjen e rezervuar **RenderEndTag**, dhe një tag të rreshtit përfundues duke përdorur metodën **Write** të **HtmlTextWriter**.

```

        output.RenderBeginTag("TD")
        kontrolla_e_Tanishme.RenderControl(output)
        output.RenderEndTag()           ' mbyllim tagun <TD>
        output.Write("</TR>")

```

Pastaj, kur i themi kontrollës së përfshirë ( ang. contained) që të renderoj veten:

```
kontrolla_e_Tanishme.RenderControl(output)
```

ajo do të thërras metodën **Render** të klasës **Numrues\_i\_Librave**. Meqë këtë metodë nuk e kemi shkelur, gjatë renderimit do të thirret metoda **Render** e klasës bazë e cila i thotë secilit object të renderoj veten. I vetmi objekt i përfshirë (ang. contained) është **ButoniNumrues**. Pasi që nuk e kemi shkelur metodën **Render** tek klasa **ButoniNumrues**, do të thirret metoda e renderimit të klasës bazë **Button** për të renderuar butonin.

Për të ekzekutuar shembullin në InternetExplorer japim shtegun:

[http://localhost/Custom\\_Kontrolla\\_e\\_Perbere\\_WebFaqja/webform1.aspx](http://localhost/Custom_Kontrolla_e_Perbere_WebFaqja/webform1.aspx)

Pasi të klikojmë disa herë në butonat për kërkimin e librave të ndryshëm, pamja e web faqes do të duket si në vijim:

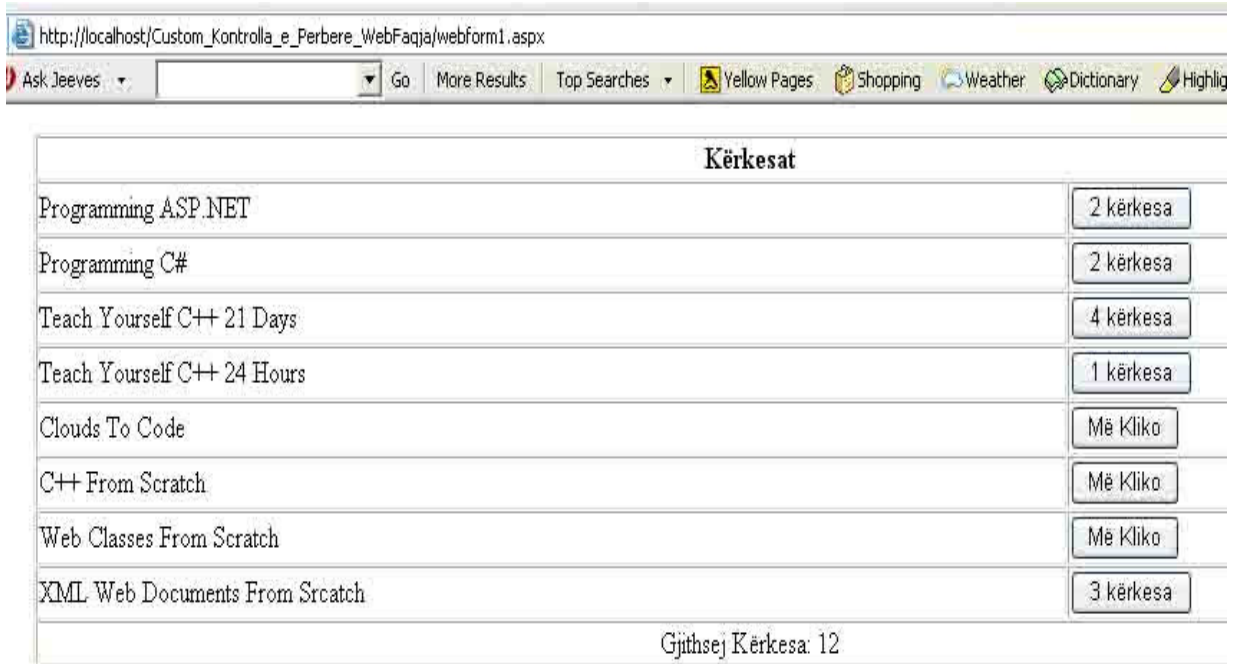


Figura 7.

## Përmbyllje

Teknologjia informative në përgjithësi dhe Interneti në veqanti vazhdojnë të kenë ndikim çdo herë e më shumë në jetën tonë të përditshme. Kështu Interneti ka gjetur zbatim pothuajse në të gjitha sferat e jetës si për shembull: informim, edukim, argëtim, komunikim, marketing etj. Agjensitë informative bëjnë publikimin e informatave, institucionet e ndryshme publikojnë aktivitetin e tyre, kompanitë afariste bëjnë promovimin dhe reklamimin e veprimtarive por edhe shitjen e produkteve të tyre, universitetet publikojnë programet arsimore, shpallin konkurset por gjithashtu mundësojnë shkollimin në distancë. Të gjitha këto mundësi kanë ardhë në shprehje përmes teknologjive të ndryshme për zhvillimin e web aplikacioneve. Një ndër këto teknologji është edhe ASP.NET-i si pasardhëse e ASP-së e zhvilluar nga Microsofti për krijimin e web aplikacioneve të lehta, të shpejta, të sigurta dhe interaktive. Përderisa verzionet para .Net-it paraqitnin vështirësi në realizimin e faqeve interaktive si dhe ndjekja dhe përmirësimi i gabimeve në kod nga zhvilluesit ishte shumë e papërshtatshme, ASP.NET-i ka eliminuar këto mangësi duke mundësuar përdorimin e dy gjuhëve të reja: VB.NET dhe C#, të cilat janë komplet të orientuara në objekte. Gjatë zhvillimit të web formave në ASP.NET mund të përdoren këto lloje të kontrollave: HTML Server Kontrollat, ASP Server Kontrollat, Kontrollat Validuese dhe Custom Kontrollat. Secila prej tyre i ka përparësitë e veta varësisht prej natyrës së aplikimit.

- HTML Server Kontrollat paraqesin HTML elemente të cilat përmes attributeve të tyre bëhen të programueshme në server,
- ASP Server Kontrollat paraqesin gjeneratën e re të kontrollave, të cilat janë më të pasura me veqori, dhe më të lehta për t'u programuar e për t'u kontrolluar nga zhvilluesi, gjithashtu në këtë grup ekzistojnë kontrollat të specializuara si për shembull AdRotator apo Calendar.
- Kontrollat për Validim janë zhvilluar në mënyrë ekskluzive për validimin e të dhënave hyrëse. Këto kontrollat mundësojnë të kontrollohet (validohet) vlera e një fushe apo kontrollë hyrëse dhe të paraqitet mesazhi i gabimit. Shfrytëzimi i tyre ka një rëndësi shumë të madhe për aplikacionin, duke e bërë atë shumë më efikas, më të lehtë dhe gjithashtu me të shpejtë për ta programuar.
- Custom kontrollat u mundësojnë programerëve të krijojnë kontrollat të reja që posedojnë funksionalitete shtesë që atyre ju nevojiten. Këto Custom kontrollat mund të realizohen duke trashëguar klasë të kontrollave të caktuara dhe pastaj të modifikohen për t'iu përshtatur nevojave të programerit ose ato mund të krijojnë edhe duke gërrshetur disa kontrollat ekzistuese brenda një Custom kontrollat të re.

Pra krejt në fund mund të themi se këto kontrollat kanë ndihmuar që të krijojnë aplikacione të mëdha me lehtësi të madhe për zhvilluesit, por edhe të afërta me shfrytëzuesin.

## Summary

Information technology in general and Internet in particular, continue to influence our every day's life. So, Internet is used almost in every field of life such as: Information, Education, Entertainment, Communication, Marketing etc. News Agencies communicate their information; business companies promote their activities and even sell their products; universities publish their educational programs, their announcements and even organize lessons and exams in distance. These are only a few from so many possibilities which were offered by several technologies for developing Web applications. One of these technologies is ASP.NET, which follows ASP, built by Microsoft, for developing interactive, safe and robust web applications. Because the versions prior to .NET had their difficulties in creating interactive pages, and also tracking and eliminating code errors by developers was very uncomfortable to them. ASP.NET has eliminated these difficulties by using two new programming languages: VB.NET and C#, which are completely object oriented. During developing web forms in ASP.NET different type of controls can be used: HTML Server Controls, ASP Server Controls, Validation Controls and Custom Controls. Each type of these controls has its own advantages depending on the nature of the application.

- HTML server controls are HTML elements which through their attributes are programmable in server.
- ASP server controls are the new generation of web forms controls which possess much more properties that make them easier to program and control by programmer. There are specialized controls in this group like AdRotator or Calendar.
- Validation controls have been developed exclusively for user input validation. These controls make possible to validate input values entered by user and display warning messages. Using these controls is very important to applications because it makes them very easy to develop, more efficient and less error prone.
- Custom controls allow programmers to create new controls which possess additional functionalities. They can be created by extending an existing control or a group of controls to provide additional functionalities.

We can conclude that these variety of controls have helped developers to create major and user friendly applications.

## SHTOJCA A

### Fjalorthi

<b>ASP.NET</b>	Active Server Pages, është teknologji për të zhvilluar web aplikacione.
<b>BROWSER</b>	Aplikacioni për kërkim në internet ( p.sh. internet explorer)
<b>CUSTOM KONTROLLA</b>	Kontrollat “me porosi”, nga përdoruesi
<b>DATASOURCE</b>	Burim i të dhënave për kontrollën
<b>FAJLLI DLL</b>	Dynamic Linked Library, librari ku gjinden klasat dhe funksionet e caktuara
<b>HTML</b>	HyperText Marked Language, është një gjuhë që përdoret nga klienti dhe serveri për të komunikuar në mes veti
<b>INTERFEJS</b>	Ndërfaqe
<b>NAMESPACE</b>	Hapësira e emërtimeve
<b>STRING</b>	Varg me karaktere, shkronja ose numra
<b>SUBMIT</b>	Buton i cili të dhënat e futura nëpër kontrolla i dorëzon në server
<b>SUBRUTINA</b>	Pjesë e kodit e cila mund të thirret nga pjesët e tjera të programit
<b>TAGU</b>	Etiketa të cilat identifikojnë elementet në HTML
<b>UPLOAD</b>	Dërgim i fajllit në server
<b>URL</b>	URL është shkurtesë nga <i>Uniform Resource Locator</i> e cila është adresë globale e dokumenteve dhe e resurseve të tjera në World Wide Web.
<b>WEB SERVERI</b>	Server i specializuar për web aplikacione

## SHTOJCA B

### Ekzekutimi i shembujve

Në mënyrë që shembujt të cilët gjinden në këtë shtojcë të mund të ekzekutohen dhe të shihen rezultatet e tyre, duhet që të plotësohen disa kushte paraprake.

Kompjuteri të jetë së paku *Pentium III*,

Të jetë i instaluar *Microsoft Visual Studio .NET 2002*,

Shembujt nuk mund të ekzekutohen, me versione më të vjetra se *Microsoft Visual Studio*

*.NET 2002*,

Shfrytëzuesi i kyqur në sistemin operativ të Windowsit të ketë autorizimet e *Administratorit*.

Në vijim janë dhënë disa udhëzime gjenerale për ekzekutimin e shembujve, e të cilat në

masë të madhe do ti ndihmojnë lexuesit që t'i përdorë këta shembuj në mënyrë të drejtë

dhe të lehtë.

Filimisht shembulli i cili dëshirohet që të ekzekutohet duhet të kopjohet nga CD-ja ku gjindet, në ndonjë folder në disk. Pastaj nga folderi në fjalë zgjedhet fajlli i selektuar si në figurën në vijim.

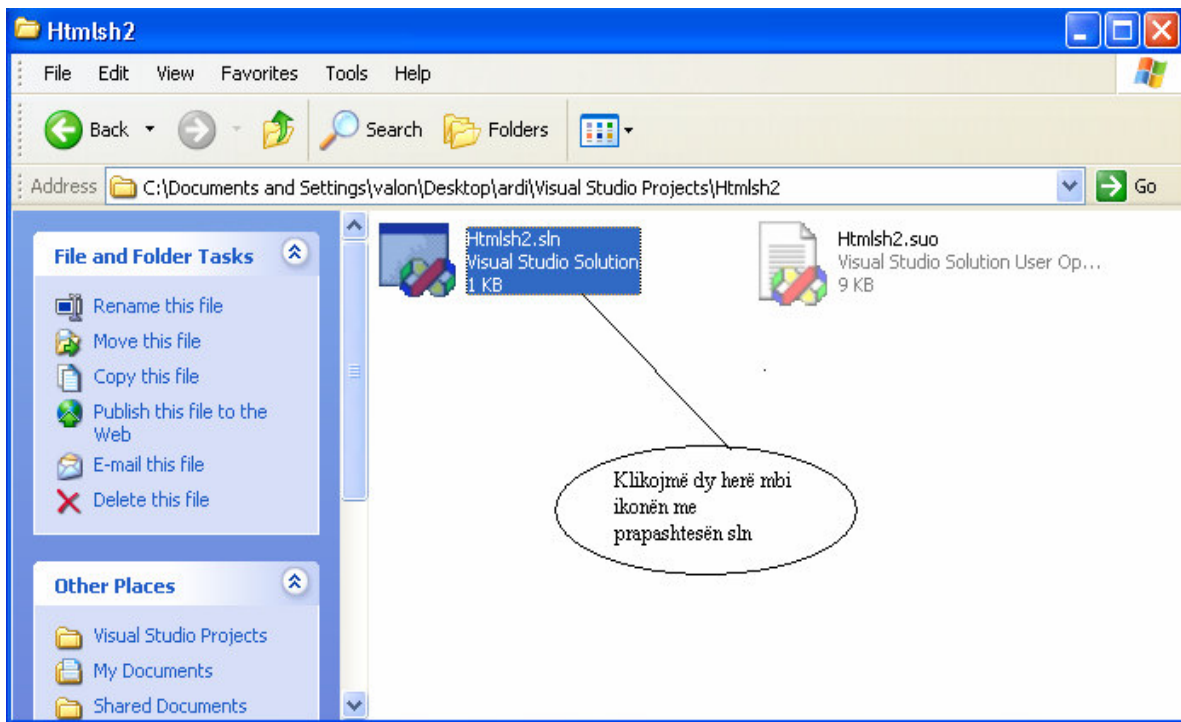


Figura 1. Zgjedhja e fajllit për ekzekutim

Në vazhdim do të hapet një dritare tjetër, por tani e Visual Studio .NET-it dhe shembulli do të jetë i gatshëm për ekzekutim. Ekzekutimi do të ndodh në momentin kur në opcionet e komandave në *Menybar*, zgjedhet opcioni *Debug* dhe nga menyja rënëse që do të shfaqet me këtë rast zgjedhet opcioni *Start (Debug – Start)*.

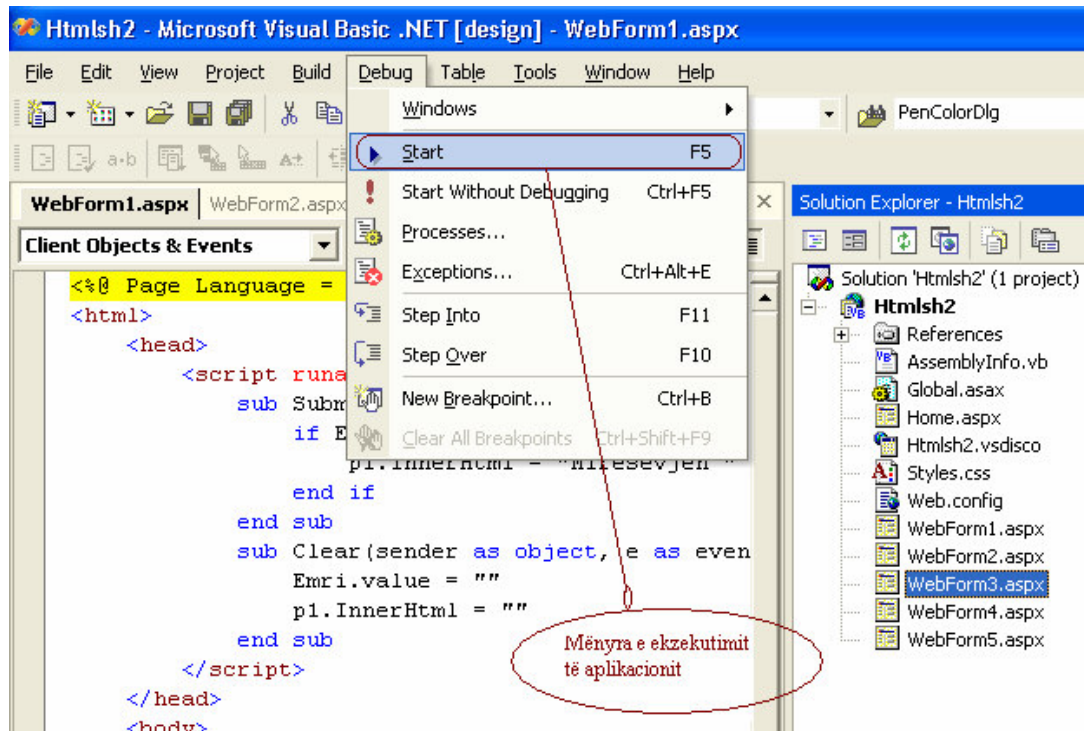


Figura. 2 Mënyrat e ekzekutimit të aplikacionit

Mënyrë tjetër e ekzekutimit është edhe duke shtypur *F5* në tastierën e kompjuterit dhe rezultati do të jetë i njëjtë.

## Literatura:

- **ASP.NET**  
G.Andrew Duthie & Matthew MacDonald  
Syngress Publishing, Inc. © 2002
- **ASP.NET Web applications on the .NET Framework**  
Danny Ryan and Tommy Ryan  
Hungry minds Inc. 909 Avenue New York © 2002
- **ASP.NET Bible**  
Mridula Parihar.  
Hungry minds Inc. 909 Avenue New York © 2002
- **ASP.NET – Database programming**  
Jason Butler and Tony Caudill  
Hungry minds Inc. 909 Avenue New York ©2002
- **ASP.NET Validation Controls**  
Jeff Prosisë's forthcoming .NET  
All material contained herein is copyrighted © 2001 by Jeff Prosisë.
- **ASP.NET Web Developer's Guide**  
Mesbah Ahmed, Garrett, Jeremy Faircloth, Chris Payne.  
Syngress Publishing, Inc. 800 Hingham Street Rockland © 2002
- **Programming ASP .NET**  
Jesse Liberty, Dan Hurwitz O'Reilly.  
Microsoft Press © 2002
- **Designing Microsoft ASP.NET Applications**  
Douglas J. Reilly  
ISBN: 0735613486 Microsoft Press © 2002

### Referimet në Web:

1. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/cpconaspsyntaxforhtmlcontrols.asp>
2. [http://www.w3schools.com/aspnet/aspnet\\_refhtmlcontrols.asp](http://www.w3schools.com/aspnet/aspnet_refhtmlcontrols.asp)
3. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbtskaddinghtmlcontrolstowebformspage.asp>
4. <http://www.asp101.com/lessons/htmlcontrols.asp>

Vërejtje: Materiali kryesisht është marrë në referimin e parë dhe të dytë në Web deri me datën 08.09.2004