Podatkovna polja, Zbirke, ArrayList, Cast

Podatkovna polja

Podatkovna polja in zbirke

- **Podatkovna polja** (*ang. array*)
  - Zaporedje elementov enakega tipa
  - Dostop do elementov s pomočjo indeksa
    - 0 … <število elementov polja – 1>

- **Zbirke** (*ang. collection*)
  - Alternativa uporabi polj
  - Tip je načeloma vedno objekt
  - Pogosto shranjevanje sklicev na objekte
    - Lahko tudi v poljih

Podatkovna polja

```
// Različne definicije polj
int[] polje1 = new int[6];
int[,] polje2 = new int[3,6];

int[] polje3 = new int[4] { 1, 2, 2, 6 };


// Vnos vrednosti
polje1[3] = 15;
polje2[2,1] = 7;
```

```
// Primer št. 1
int[] polje1 = new int[4];
polje1[2] = 6;

Response.Write(polje1[2]);


// Primer št. 2
try
{
                int[] polje2 = { 9, 3, 7, 2 };
                Response.Write(polje2[4]);
}
catch (IndexOutOfRangeException caught)
{
                …
}
```

```
// Primer št. 3
int[] polje3 = { 9, 3, 7, 2 };
for (int i = 0; i != polje3.Length; i++)
{
   int element = polje3[i];
   Response.Write(element.ToString());

}


// Primer št. 4
int[] polje4 = { 9, 3, 7, 2 };
foreach (int element in polje4)
{
   Response.Write(element.ToString());

}
```

Zbirke (1)

- Brez vnaprej določene dolžine
- Najpogosteje uporabljene zbirke
  - *ArrayList*
  - *Queue*
  - *Stack*
  - *SortedList*
- Imenska prostora
  - System.Collections
  - System.Collections.Generic
    - Parametrizirani razredi
    - *List<T>*

Zbirke (2)

- Pogoste operacije
  - *Add*
  - *Clear*
  - *Remove*
  - *RemoveAt*
  - *IndexOf*
- Specifične operacije
  - Queue: *Enqueue*, *Dequeue*
  - Stack: *Push*, *Pull*

Zbirke – ArrayList

```
string niz = "Slavko Avsenik";

Krog krog = new Krog(5);



ArrayList al = new ArrayList();


// Dodajanje elementov
al.Add(niz);

al.Add(krog);


// Število elementov

int stElementov = al.Count;


// Dostop do elementov
foreach (object element in al)
            if (element.GetType() == niz.GetType())
                        Response.Write(element.ToString());
            else if (element.GetType() == krog.GetType())
                        Response.Write(((Krog)element).Povrsina().ToString());
```

Zbirke in gradniki

- Gradniki pogosto uporabljajo zbirke za shranjevanje vsebine
  - *DropDownList*
  - *ListBox*
- Lastnost *Items*
  - Izpeljana iz zbirk
  - Uporaba enakih operacij kot pri zbirkah

```
ListBox1.Items.Clear();
ListBox1.Items.Add("Informacijski sistemi");
```

Cast

- Pretvorba nekega podatkovnega tip v drugega
- Cast ≠ razred *Convert*
- Pretvorba splošnih objektov
  - Splošnemu objektu "določimo" konkreten razred

```
// Cast iz tipa "object" v "int"
object o = 42;

int i = (int)o;


// Cast v tip "Krog"
Response.Write(((Krog)element).Povrsina().ToString());
```

Dinamično dodajanje gradnikov (1)

- Uporaba gradnika *PlaceHolder*

```
// Kreiranje gradnikov
for (int i = 0; i < 10; i++)
{
    TextBox tb = new TextBox();
    tb.ID = "TB" + i.ToString();

    // Uporaba gradnika PlaceHolder
    PlaceHolder1.Controls.Add(tb);
}

// Dostop do gradnikov
for (int i = 0; i < 10; i++)
{
    Label1.Text += ((TextBox)PlaceHolder1.FindControl("TB" + i.ToString())).Text;
}
```

Dinamično dodajanje gradnikov (2)

- Uporaba objekta Page

```
// Dodajanje gradnika v obrazec (med <form> in </form>).
// ID obrazca najdete na strani .aspx pod značko <form id="form1" >

TextBox tb = new TextBox();
tb.Text = "Test";

// Deluje, če na strani .aspx ni programske kode
// Gradnik doda na začetek spletnega obrazca
FindControl("form1").Controls.AddAt(0,tb);
```

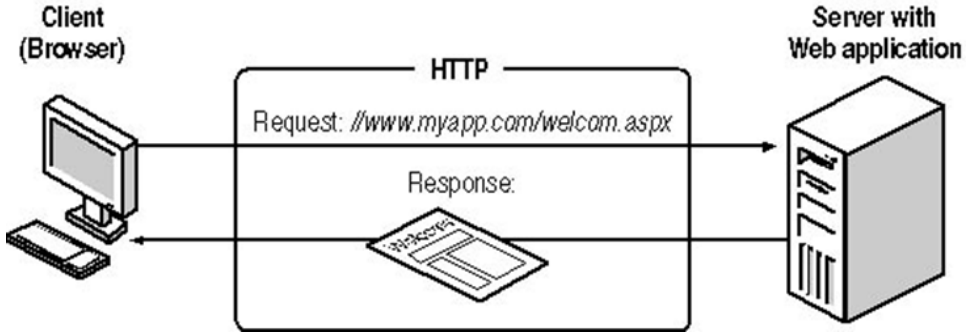
Vaja 4

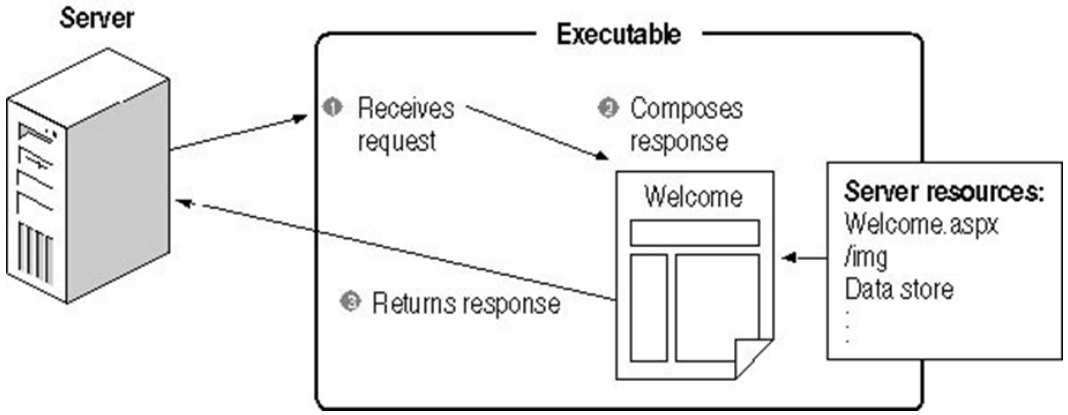
- Kalkulator razširite tako, da bo omogočal pregled zgodovine rezultatov
  - Zgodovina naj se shranjuje v gradnik *ListBox*

- Na Kalkulator dodajte še nov gumb, ki bo med delovanjem programa sprožil **dinamično dodajanje** gradnika *TextBox* na spletno formo. V dodanem gradniku se bo izpisalo število elementov v gradniku *ListBox.*
  - Programsko kreirajte nov objekt razreda *TextBox* in mu nastavite ustrezne lastnosti
  - Nov objekt dodajte na spletno formo

Delovanje spletnih aplikacij

Delovanje spletnih aplikacij (1)



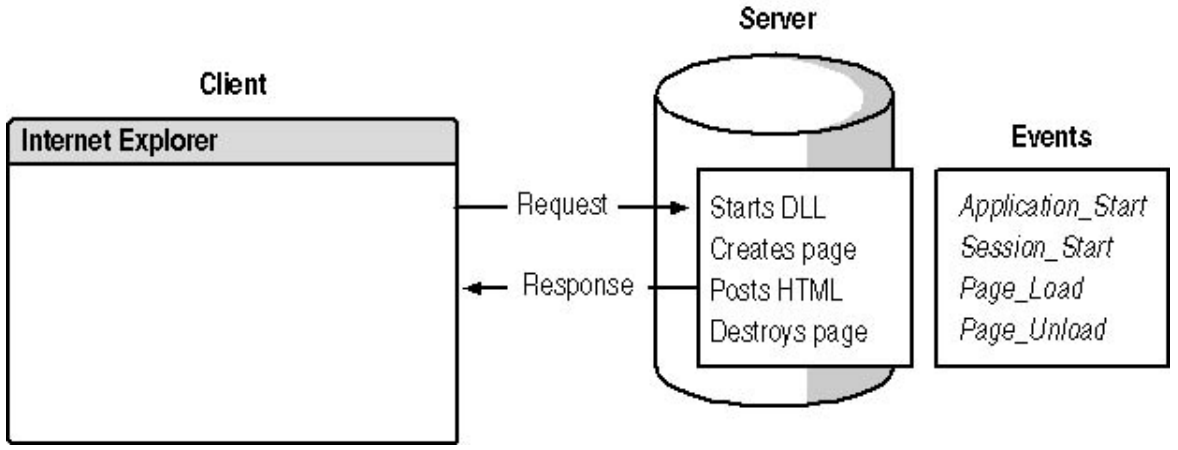


Delovanje spletnih aplikacij (2)

- **Spletna aplikacija**
  - Izvajanje na strežniku
  - Uničenje po poteku vseh sej
- **Seja**
  - Kreiranje ob novi povezavi
  - Uničenje po preteku določenega časa (*ang. timeout*) ali ob klicu *Session.Abandon()*
- **Spletna stran**
  - Kreiranje na strežniku
  - Se ne shranjuje, tj. uniči po odgovoru strežnika
  - Uničijo se tudi vse spremenljivke

Življenjski cikel (1)

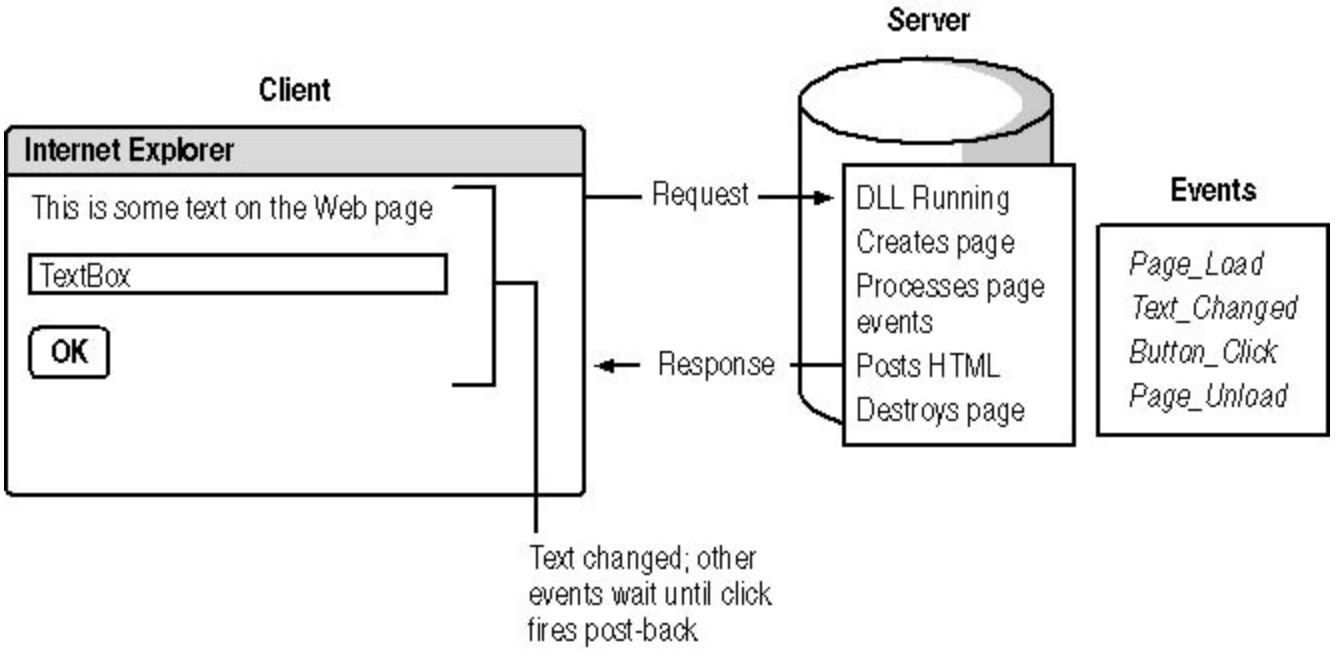
- **Zagon** spletne aplikacije
  - Datoteka .dll
  - Začetek nove seje
  - Izdelava, pošiljanje in uničenje spletne strani



Življenjski cikel (2)

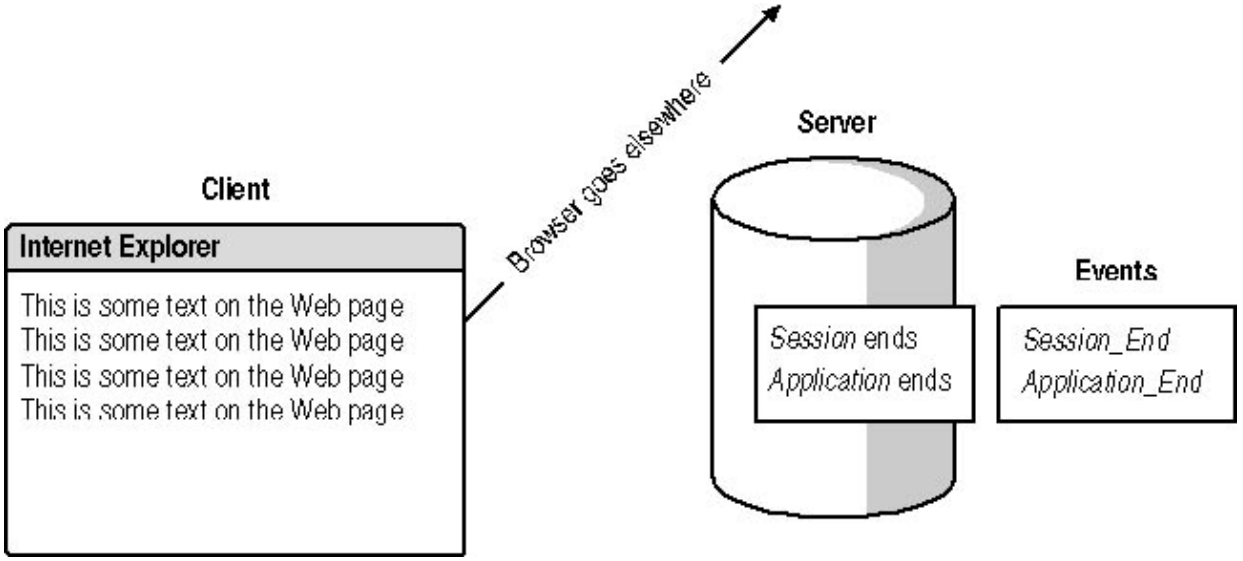
- **Delovanje** spletne aplikacije
  - Post-back
    - Sproži ga dogodek *Button_Click*
    - Pošiljanje nove zahteve
  - Vsi ostali dogodki "čakajo" na Post-back
    - Obravnava dogodkov na strani strežnika

Življenjski cikel (3)



Življenjski cikel (4)

- **Konec** delovanja
  - Seja se uniči po preteku določenega časa
  - Aplikacija se uniči po uničenju zadnje seje

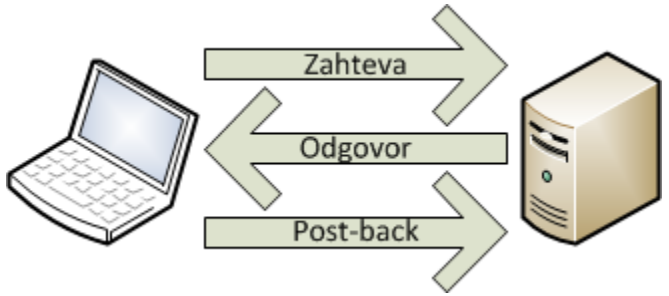


Dogodki spletnih gradnikov

- **Dogodki Post-back**
  - Neposredna zahteva za obdelavo strani
  - Obdelava na strežniku
  - Npr. *Button_Click*
- **Shranjeni dogodki**
  - Obdelajo se ob prvem dogodku Post-back
  - Obdelava na strežniku
  - Npr. *TextBox_TextChanged*
- **Dogodki za validacijo vnosa**
  - Obdelava na odjemalcu

Lastnost IsPostBack

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        // Programska koda, ki se izvede samo ob prvem prikazu strani
    }
    if (IsPostBack)
    {
        // Programska koda, ki se izvede samo ob ponovnem prikazu strani
    }
    // Programska koda, ki se izvede vedno
}
```



Preusmerjanje

- *Response.Redirect*
  - Brskalnik je preusmerjen na nov naslov
  - Katerikoli naslov na spletu
- *Server.Transfer*
  - Strežnik poda drugo spletno stran pod istim naslovom
  - Skrito brskalniku
  - Samo strani v domeni strežnika

```
Response.Redirect("http://www.google.si");

Server.Transfer("Vaja25.aspx");
```

Prenos stanj

Vaja 5

- Kreirajte novo spletno stran
  - Definirajte števec kot objektno spremenljivko tipa *int*
  - Dodajte en gradnik *TextBox* in dva gumba
    - Ob kliku na prvi gumb naj se spremenljivka števec poveča za 1 in njena nova vrednost zapiše v *TextBox*
    - Ob kliku na drugi gumb naj se spremenljivka števec zmanjša za 1 in njena nova vrednost zapiše v *TextBox*

Opazujte dogajanje
Razložite dogajanje
Predlagajte, kako bi problem odpravili

Hranjenje in prenos stanja

- Samodejen prenos stanja v gradnikih
- Načini prenosa stanja spremenljivk

Parametri v naslovu (*ang. Query String*)	Znakovni niz (String)
Piškotki (*ang. Cookies*)	Znakovni niz (String)
Lastnost ViewState	Znakovni niz (String)
Sejna spremenljivka (ang. *Session State*)	Objekt
Aplikacijska spremenljivka (ang. *Application State*)	Objekt

Parametri v naslovu *(ang. Query String)*

- Prenos vrednosti spremenljivke v naslovu
- Vrednost na voljo po preusmeritvi

```
protected void Button1_Click(object sender, EventArgs e)
{
    // Preusmeritev na naslov s podanim parametrom
    Response.Redirect("Default.aspx?Ime=Andrej&Priimek=Petrovič");
}



protected void Page_Load(object sender, EventArgs e)
{
    // Dostop do parametra v naslovu
    Label1.Text = Request.QueryString["Ime"] + " " +
        Request.QueryString["Priimek"];
}
```

Piškotki *(ang. Cookies)*

- Hranjenje podatkov pri odjemalcu/v brskalniku
- Vrednost na voljo po post-back
  - Zavračanje piškotkov

```
// Kreiranje piškotka
if (Request.Browser.Cookies)
{
    HttpCookie hc = new HttpCookie("Jezik");
    hc.Value = "Slovensko";
    Response.Cookies.Add(hc);

}


// Dostop do piškotka
if (Request.Browser.Cookies)
    if (Request.Cookies["Jezik"] != null)
        Label1.Text = Request.Cookies["Jezik"].Value;
```

Lastnost ViewState

- Prenos vrednosti spremenljivke v skritem polju
- Vrednost na voljo po post-back

```
// Pisanje v lastnost ViewState
ViewState.Add("Ime", "Krešimir");


// Dostop do lastnosti ViewState
Label1.Text = "";
foreach (StateItem si in ViewState.Values)
{
    Label1.Text += si.Value.ToString();
}
```

Sejna spremenljivka

- Hranjenje vrednosti na strežniku
- Veljavnost do konca seje
- Vrednost na voljo takoj

```
// Pisanje v sejno spremenljivko
Session["Naziv"] = "Grof";


// Dostop do sejne spremenljivke
if (Session["Naziv"] != null)
    Label1.Text = (String)Session["Naziv"];
```

Aplikacijska spremenljivka

- Hranjenje vrednosti na strežniku
- Veljavnost do konca zadnje seje
- Vrednost na voljo takoj
- Vrednost dostopna v vseh sejah

```
// Pisanje v aplikacijsko spremenljivko
Application["Naziv"] = "Grof";


// Dostop do aplikacijske spremenljivke
if (Application["Naziv"] != null)
    Label1.Text = (String)Application["Naziv"];
```

Vaja 6

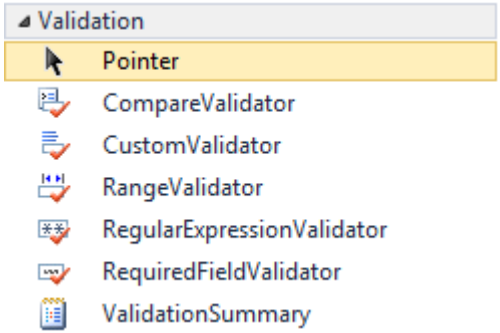
- Popravite spletno stran, ki ste jo izdelali v sklopu vaje 5
  - Uporabite sejno spremenljivko

```
// Pisanje v sejno spremenljivko
Session["Naziv"] = "Grof";

// Dostop do sejne spremenljivke
if (Session["Naziv"] != null)
    Label1.Text = (String)Session["Naziv"];
```

Validacija vnosa

Validacija vnosa (1)



- Validacija vnosa
  - Pri odjemalcu
  - Na strežniku
- Gradniki za validacijo – nastavitev lastnosti
  - *ControlToValidate*
  - *ErrorMessage*
  - *IsValid*
  - *Validate*
  - *Display*

Validacija vnosa (2)

- RequiredFieldValidator
  - Lastnosti
    - *ControlToValidate*

```
Required field:
<asp:textbox id="TextBox1" runat="server"/>


<asp:RequiredFieldValidator id="valRequired" runat="server"
ControlToValidate="TextBox1" ErrorMessage="* Please enter a value"
Display="dynamic">*
</asp:RequiredFieldValidator>
```

Validacija vnosa (3)

- CompareValidator
  - Lastnosti
    - *ControlToValidate*
    - *ControlToCompare*
    - *Operator*

```
Textbox 1: <asp:textbox id="textbox1" runat="server"/><br />
Textbox 2: <asp:textbox id="textbox2" runat="server"/><br />


<asp:CompareValidator id="valCompare" runat="server"
   ControlToValidate="textbox1" ControlToCompare="textbox2"
   Operator="Equals"
   ErrorMessage="* You must enter the same values into textbox 1 and textbox 2"
   Display="dynamic">*
</asp:CompareValidator>
```

Validacija vnosa (4)

- RangeValidator
  - Lastnosti
    - *ControlToValidate*
    - *MinValue*
    - *MaxValue*
    - *Type*

```
Enter a date from 1998:
<asp:textbox id="textbox1" runat="server"/>


<asp:RangeValidator id="valRange" runat="server"
   ControlToValidate="textbox1"
   MaximumValue="12/31/1998"
   MinimumValue="1/1/1998"
   Type="Date"
   ErrorMessage="* Date must be between 1/1/1998 and 12/13/1998" Display="static">
   *
</asp:RangeValidator>
```

Validacija vnosa (5)

- RegularExpressionValidator
  - Regularni izrazi
  - Lastnosti
    - *ControlToValidate*
    - *ValidationExpression*

```
E-mail:
<asp:textbox id="textbox1" runat="server"/>


<asp:RegularExpressionValidator id="valRegEx" runat="server"
   ControlToValidate="textbox1"
   ValidationExpression=".*@.*\\..*"
   ErrorMessage="* Your entry is not a valid e-mail address."
   display="dynamic">*
</asp:RegularExpressionValidator>
```

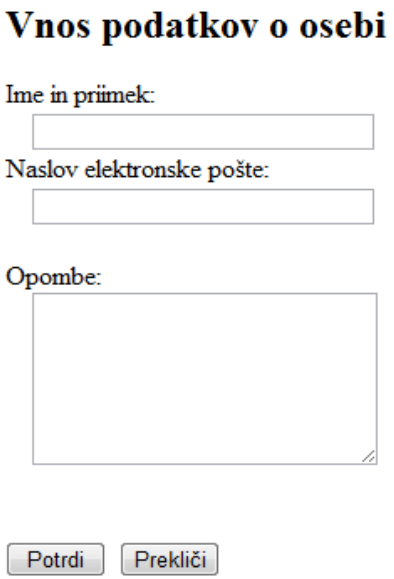
Validacija vnosa (6)

- Gradnik ValidationSummary
  - Prikaz vseh napak
    - *ShowSummary*
    - *ShowMessageBox*

```
<asp:ValidationSummary id="valSummary" runat="server"
   HeaderText="Errors:"
   ShowSummary="true" DisplayMode="List" />
```

Vaja 7

- Izdelajte obrazec s tremi gradniki *TextBox* in preverite pravilnost vnosa.
  - Ime in priimek (obvezen podatek, dve besedi, brez cifer)
  - Naslov elektronske pošte (pravilen format elektronskega naslova)
  - Opombe (brez preverjanja)



RegEx za dve besedi, brez cifer:
"^[A-Za-z]+ [A-Za-z]+$"

V primeru napake "*WebForms UnobtrusiveValidationMode requires a ScriptResourceMapping for 'jquery,…*" dodajte v datoteko **Web.config** ključ

```
<appSettings>
  <add key="ValidationSettings:UnobtrusiveValidationMode" value="None" />
</appSettings>
```