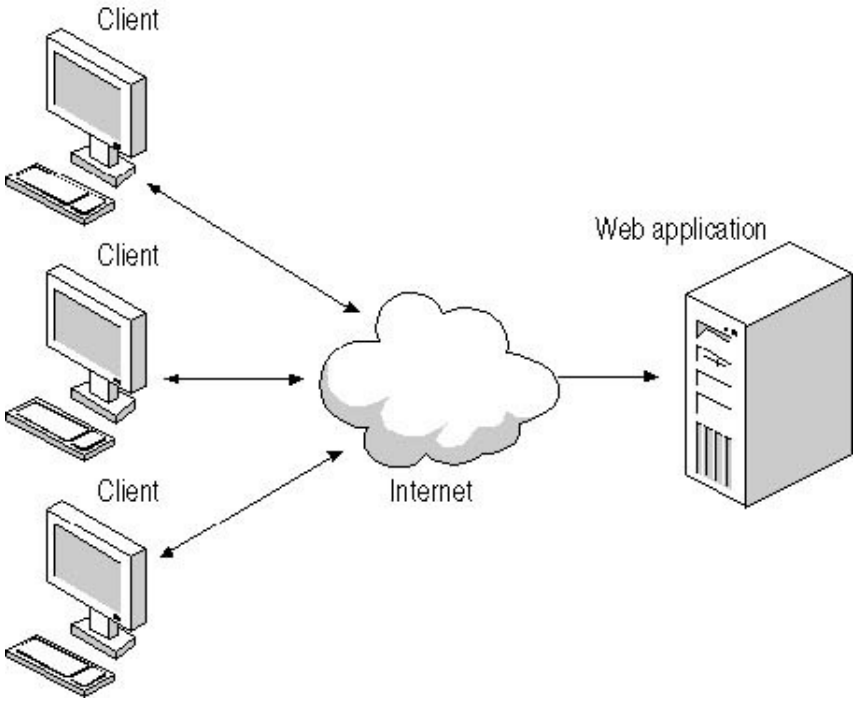
SPLETNE APLIKACIJE

Osnove, izdelava in zagon

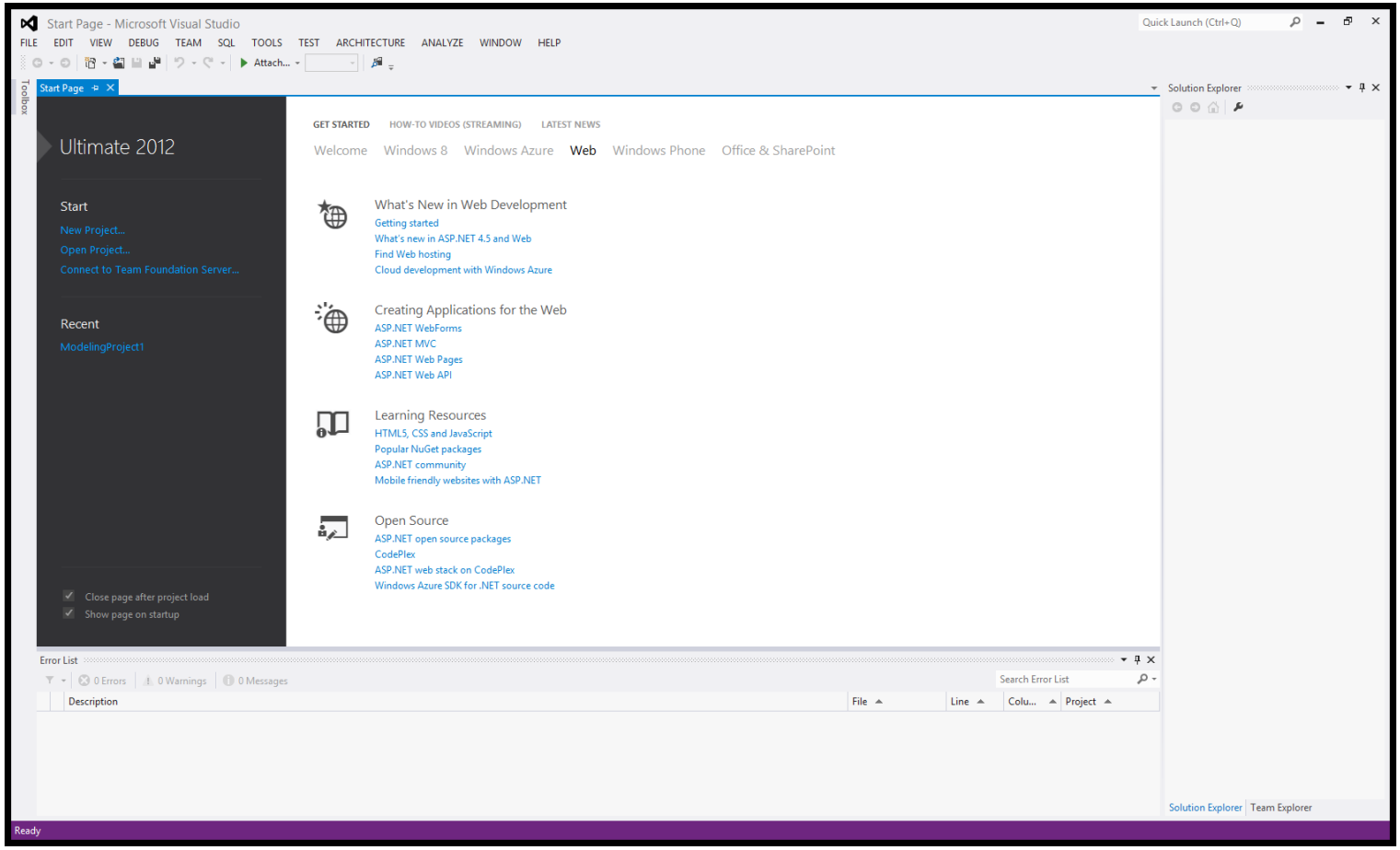
OSNOVE IZDELAVE SPLETNIH APLIKACIJ

Spletne aplikacije

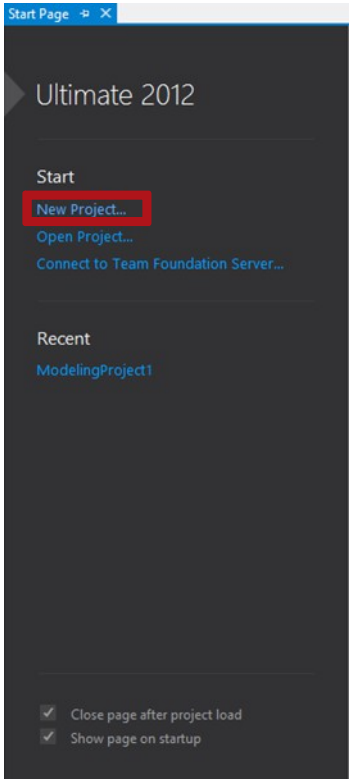
- **Spletni strežnik**
  - Izdelava vsebine na spletnem strežniku za odjemalce na internetu
- **Odjemalci**
  - Uporaba s pomočjo spletnega brskalnika

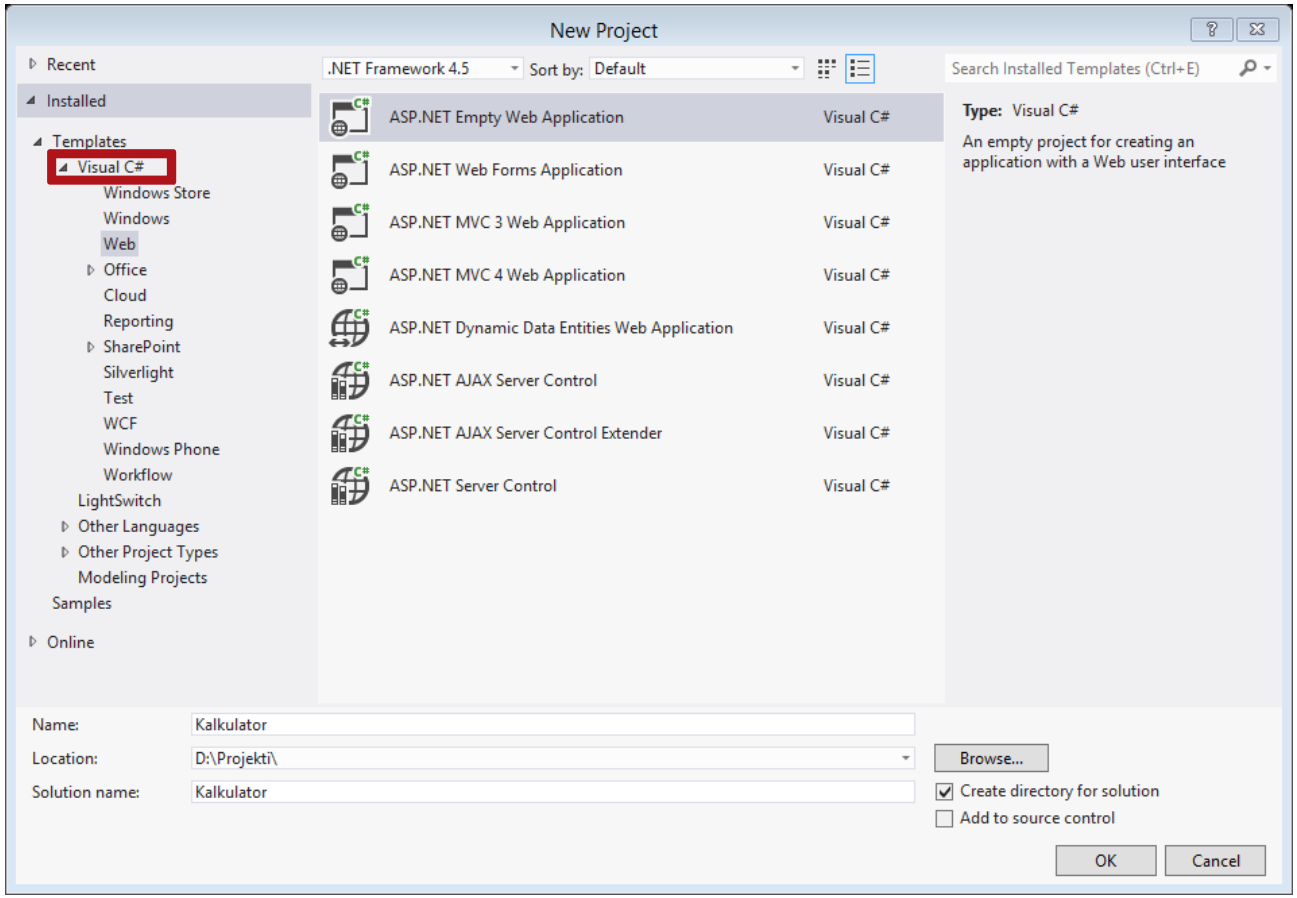


Microsoft Visual Studio 2012

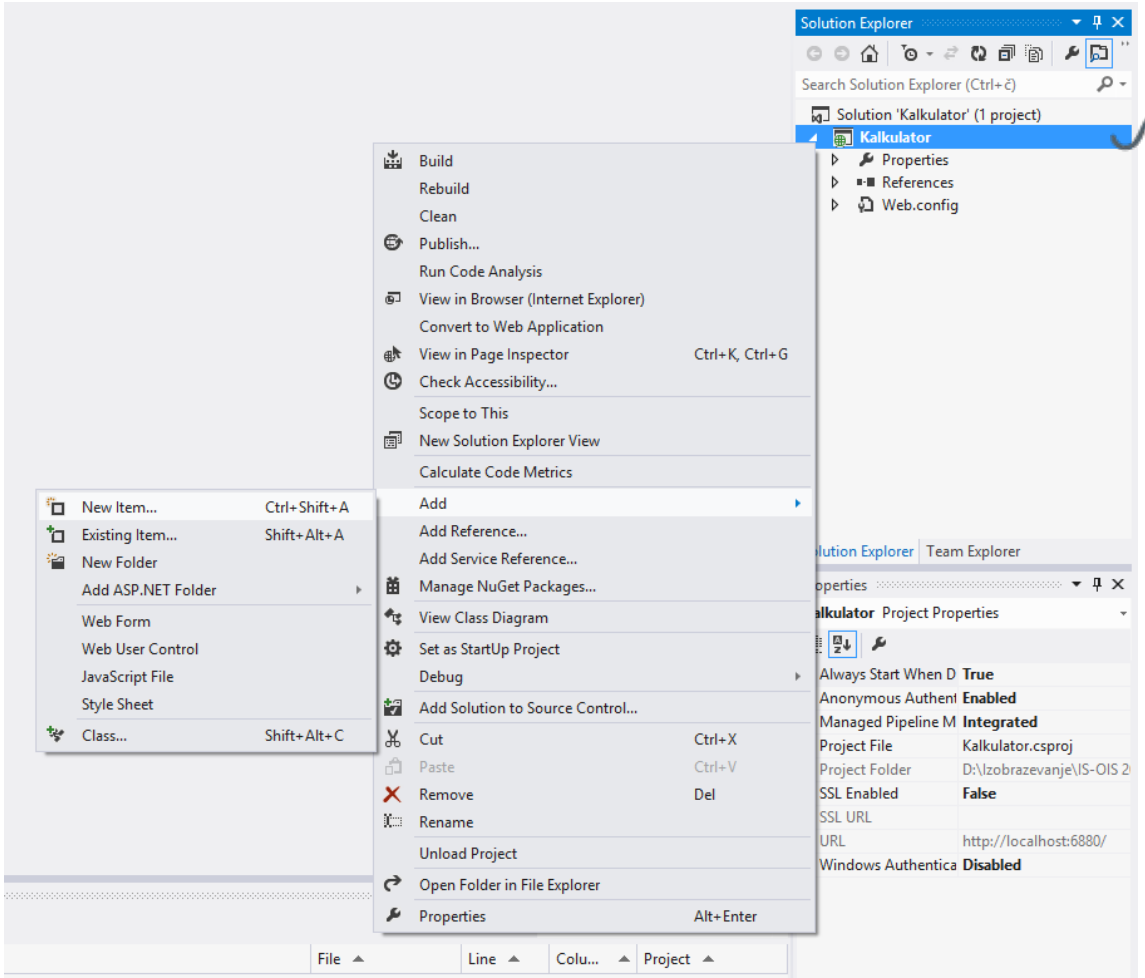


Izdelava spletne aplikacije (1)

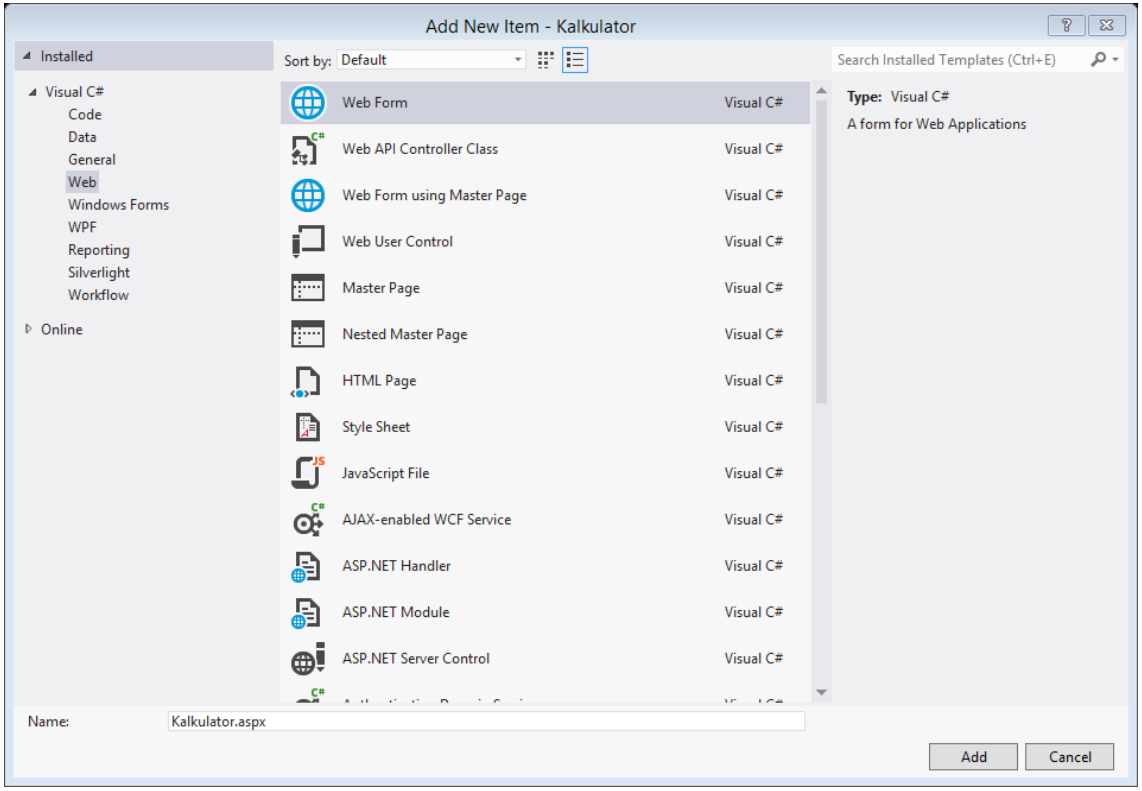




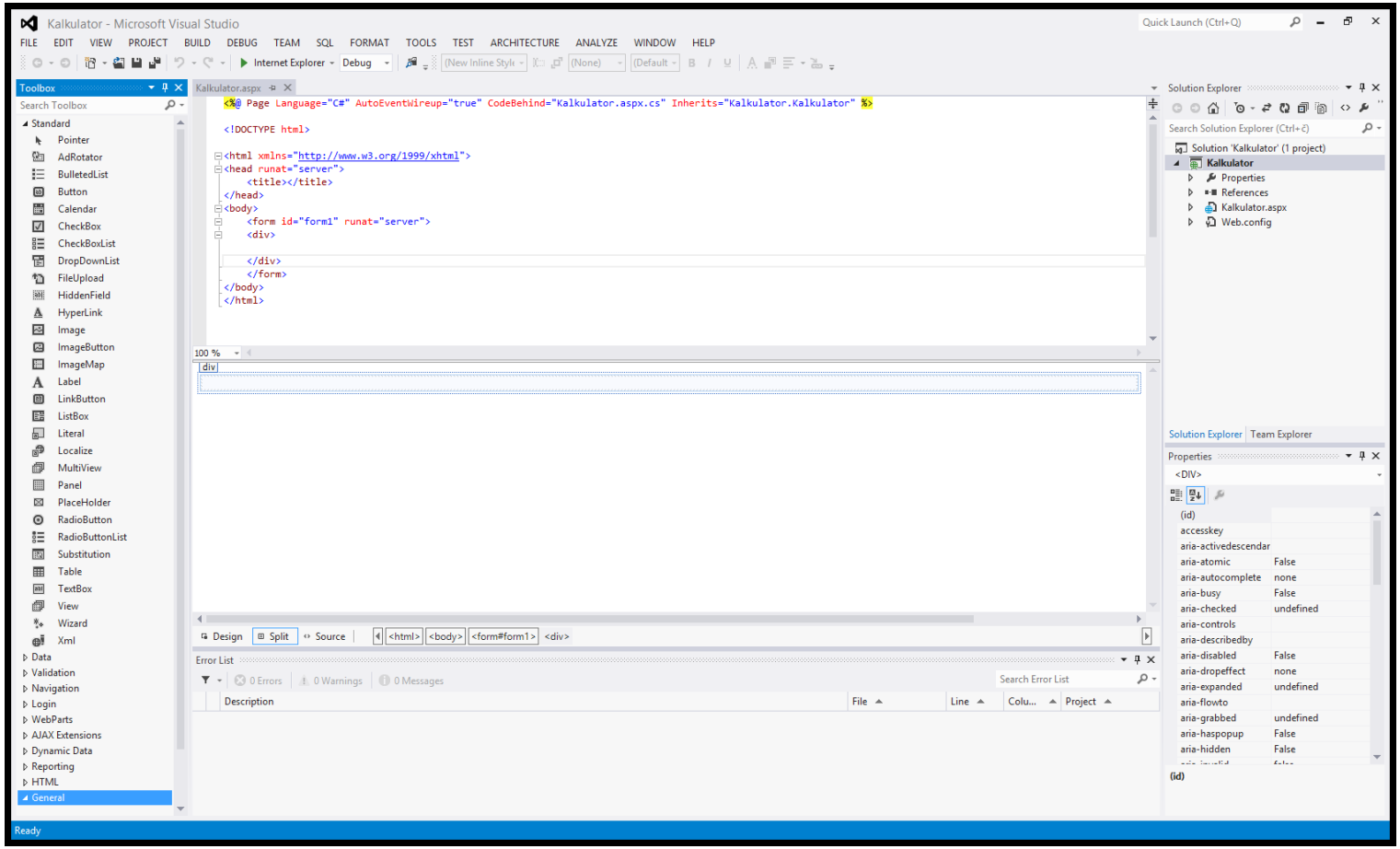
Izdelava nove spletne strani (1)



Izdelava nove spletne strani (2)

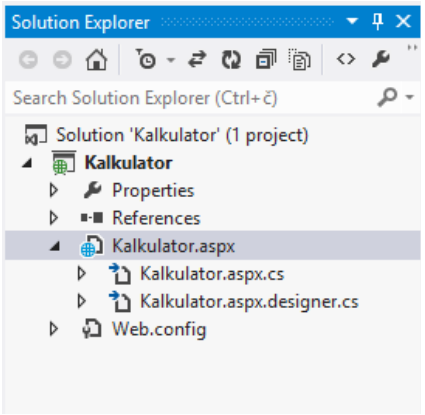


Urejanje spletne strani (1)



Urejanje spletne strani (2)

- **Solution Explorer**
  - Pregled strukture projekta (spletne strani, knjižnice, nastavitvene datoteke itd.)
- **Reference**
  - Seznam referenciranih komponent (knjižnic)
  - Vključevanje dodatnih komponent
- **Datoteke .aspx**
  - Posamezna spletna stran
  - HTML + JavaScript
  - Samo lastnosti (brez dogodkov)
  - Koda, ki se izvede na strežniku
- **Datoteke .aspx.cs**
  - Koda v povezavi z datoteko .aspx
  - Dogodki



Urejanje spletne strani (3)

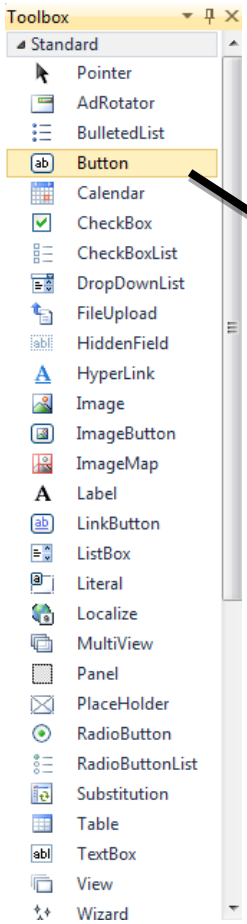
- Uporaba vizualnega urejevalnika
- Dodajanje vizualnih gradnikov na obrazce
  - **Gradnik** (*ang. Control*)
    - Lastnosti (*ang. property*)
    - Dogodki (*ang. event*)
      - Izvajanje (poljubne) programske ob dogodkih
    - Operacije
- Oblikovanje gradnikov
  - Postavitev na poljubno mesto, ki je dovoljeno s standardnim HTML

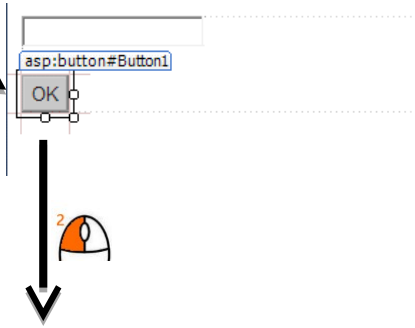
Urejanje spletne strani (4)

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

        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="OK" />

    </div>
    </form>
</body>
</html>
```



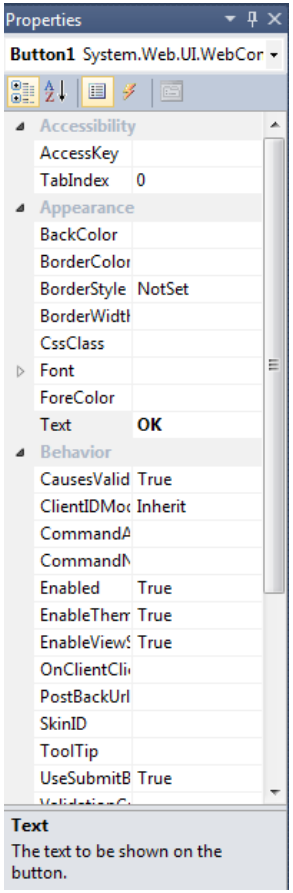


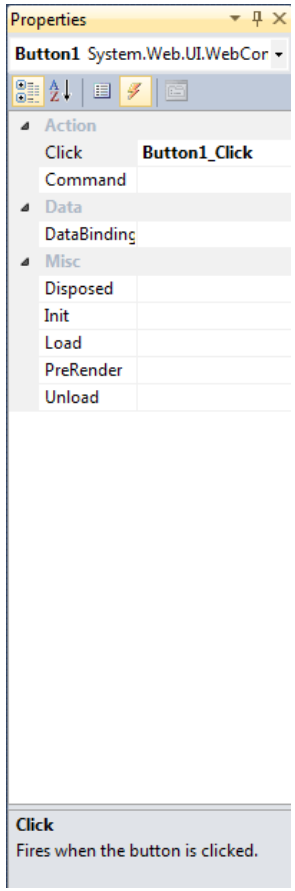

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

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

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            TextBox1.Text = "Hello world!";
        }
    }
}
```





Strani .aspx in C#

- Pisanje programske kode C# v bloku <% … %>

```
<%
 int x = 1;
 x++;
%>
```

- Povezava z datoteko .aspx.cs
  - Programska logika
  - V glavi spletne strani (datoteka .aspx)

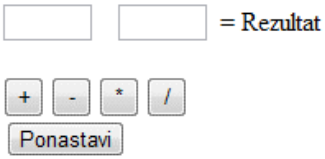
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Kalkulator.aspx.cs" Inherits="PrvaSpletnaAplikacija.Kalkulator"%>
```

Zagon spletne aplikacije

- Debug -> Start Without Debugging (Ctrl + F5)
- Spletni strežnik
  - IIS (*Internet Information Services*)
  - Integrirani spletni strežnik znotraj VS
    - Razvojni strežnik
    - Zagon ob prvem zagonu aplikacije
    - Kasneje osveževanje strani v brskalniku

Vaja 1

- **Izdelajte preprost spletni kalkulator**
- Izdelava kalkulatorja z vizualnim urejevalnikom
- Dodajanje gradnikov
  - Določanje lastnosti v vizualnem urejevalniku
  - Določanje dogodkov (seštevanje, odštevanje, deljenje, množenje)
- Branje in spreminjanje lastnosti kontrol med delovanjem programa



Pretvarjanje podatkovnih tipov
Razred **Convert**

```
int i = Convert.ToInt32(TextBox1.Text);
double d =
Convert.ToDouble(TextBox1.Text);
string s1 = Convert.ToString(a);
string s2 = a.ToString();
```

Razred, konstruktor, prekrivanje, imenski prostori, dedovanje, vmesniki…

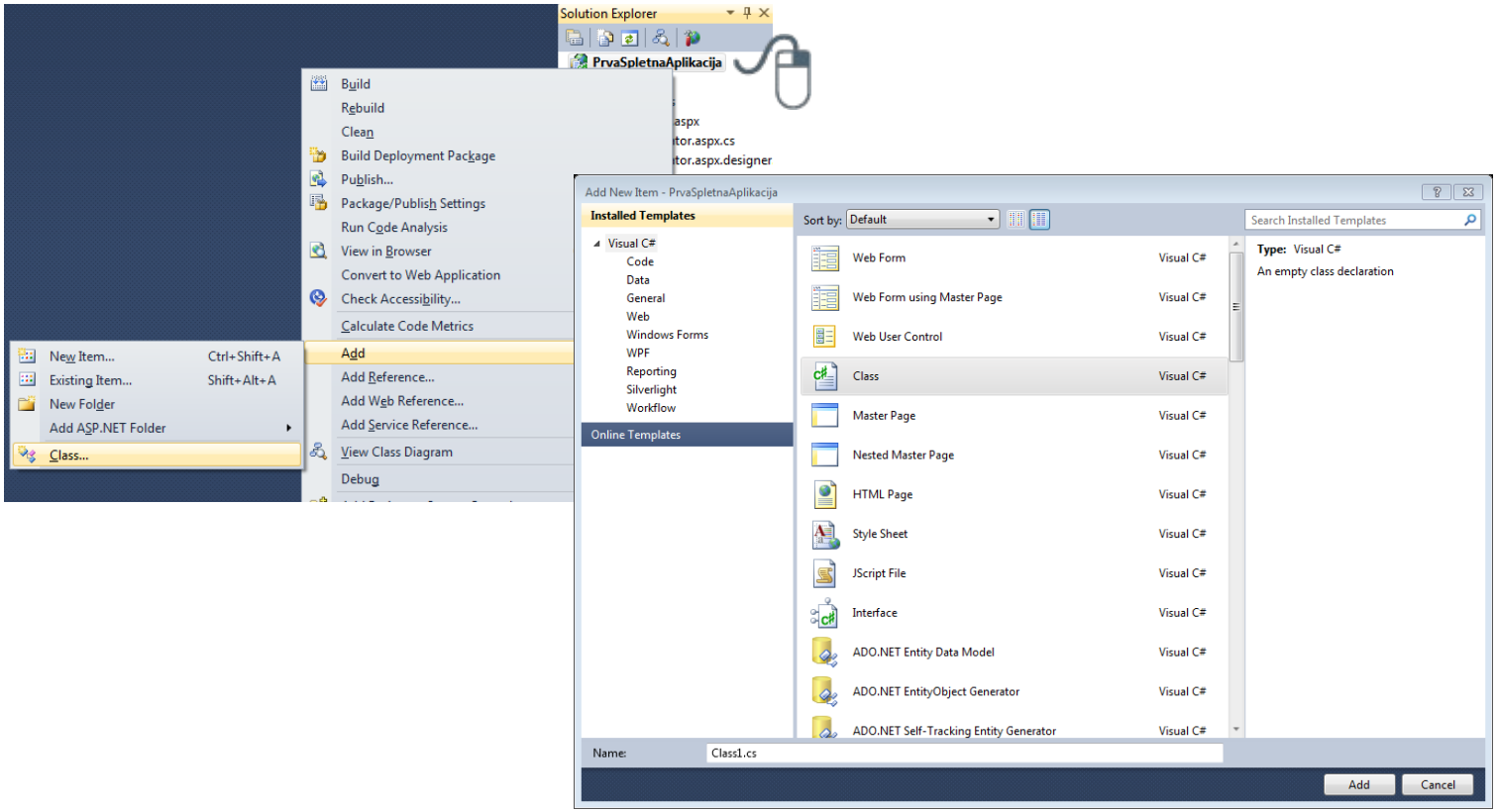
OBJEKTNO USMERJENO PROGRAMIRANJE

Razred

- Klasifikacija, enkapsulacija, dedovanje itd.
- Razred združuje objekte, ki imajo
  - Skupne lastnosti (atribute)
  - Skupno obnašanja (operacije)
  - Skupne povezave
  - Skupen pomen

```
class Krog
{
	double radij;
	double Povrsina()
	{
		return Math.PI * radij * radij;
	}
}
```

Dodajanje novega razreda



Konstruktor

- Operacija, ki kreira novo instanco razreda (objekt)
- Enako ime kot razred
- Konstruktor lahko izdelamo sami
  - Če ga ne izdelamo, prevajalnik sam izdela osnovni konstruktor

```
class Krog
{
	private double radij;
	public Krog(double zacetniRadij)
	{
		radij = zacetniRadij;
	}
	public double Povrsina()
	{
		return Math.PI * radij * radij;
	}
}
```

Prekrivanje operacij

- Prekrivanje (ang. overload)
- Definiranje dveh ali več operacij z enakim imenom in različnimi nabori atributov znotraj razreda
  - Možno je prekrivanje konstruktorja
- Nabor atributov pri klicu operacije določa operacijo, ki se izvede

```
class Krog
{
    private double radij;
    public Krog(double zacetniRadij)
    {
        radij = zacetniRadij;
    }
    public double Povrsina()
    {
        return Math.PI * radij * radij;
    }
    public double Povrsina(double novRadij)
    {
        return Math.PI * novRadij * novRadij;
    }
}
```

Kreiranje novega objekta

- Tip spremenljivke je enak imenu razreda
- Nov objekt se kreira z ukazom **new**, ki mu sledi klic konstruktorja
- V spremenljivko se zapiše sklic na nov objekt

```
// Kreiranje objekta
Krog k;
k = new Krog(15);

// Združitev obeh korakov
Krog k = new Krog(15);

// Klic operacij nad objektom
k.Povrsina(10);
double povrsina = k.Povrsina();
```

Vnaprej definirani razredi ogrodja .NET

- Vnaprej definirani razredi, vmesniki in podatkovni tipi
  - Ključni gradniki ogrodja .NET
  - Lahko se jih uporabi pri programiranju lastnih aplikacij
  - Gradniki aplikacij, komponent in gradnikov v .NET
- Razredi so razporejeni po posameznih imenskih prostorih glede na njihov namen

Uporaba imenskih prostorov

- Dostopni imenski prostori
  - Del referenciranih komponent (projektne reference)
- Dostop do razredov v imenskih prostorih
  - Na začetku datoteke – **using**
  - Pri klicu razreda

```
// Navedba imenskega prostora pri klicu razreda
System.Drawing.Pen p = new System.Drawing.Pen(System.Drawing.Color.Red);

// Navedba imenskega prostora na začetku datoteke
using System.Drawing;
…
Pen p = new Pen(Color.Red);
```

Dedovanje (1)

- Razred neposredno deduje le iz enega razreda
  - Osnovni razred (ang. parent class oz. base class)
  - Podrazred (ang. child class)
- Podrazred
  - Podeduje vse ne-zasebne lastnosti in operacije
  - Definira svoje lastnosti in operacije
- Vsi razredi so podrazredi System.Object
  - Tudi če ni eksplicitno zapisano
  - Vsaj posredno

Dedovanje (2)

```
// Dedovanje iz razreda OsnovniRazred
class Podrazred : OsnovniRazred
{

      // Klicanje konstruktorja osnovnega razreda
      public Podrazred(string parameter) : base(parameter)
      {
            // Programska koda konstruktorja podrazreda
      }

      // Lastnosti in operacije podrazreda
      …
}
```

Dedovanje (3)

```
// Osnovni razred
class Krog
{
	protected double radij;
	public Krog(double zacetniRadij)
	{
		radij = zacetniRadij;
	}
	public double Povrsina()
	{
		return 3.141592 * radij * radij;
	}
	public double Povrsina(double novRadij)
	{
		return 3.141592 * novRadij * novRadij;
	}
}
```

```
// Podrazred
class KrogPlus : Krog
{
	public KrogPlus(double zacetniRadij) : base(zacetniRadij) { }
	public double Obseg()
	{
		return 2 * Math.PI * radij;
	}
}

// Podrazred podeduje operacije osnovnega razreda
KrogPlus k = new KrogPlus(2);

Response.Write(k.Obseg().ToString());
Response.Write(k.Povrsina().ToString());
```

Razredne (statične) operacije

- Klic z uporabo imena razreda
- Ni potrebno kreirati objekta

```
class Math
{
      public static double Sqrt(double d) { … }
}

// Klic statične operacije
double d = Math.Sqrt(42.24);

// Klic operacije, če ne bi bila razredna
Math m = new Math();
double d = m.Sqrt(42.24);
```

Vmesniki (1)

- Vmesniki (ang. interface) ločujejo
  - Predstavitev: Kaj nudi?
  - Implementacija: Kako je implementirano?
- Ni mogoče kreirati objektov
- Ni konstruktorjev in destruktorjev
- Vse operacije so javne
  - Brez oznak
- Naštete operacije
  - Niso implementirane
  - Implementirane v razredu, ki realizira vmesnik

Vmesniki (2)

```
interface ILik
{
      double Obseg();
}

// Razred, ki realizira vmesnik
class Trikotnik : ILik
{
      public double Obseg() { … }
}

// Razred, ki hkrati deduje od drugega razreda in realizira vmesnik
class BarvniTrikotnik : BarvniLik, ILik
{
      …
}
```

Vaja 2

- **Izdelajte razred Pravokotnik, ki naj vključuje**
  - Zasebni atribut sirina
  - Zasebni atribut visina
  - Javno operacijo IzracunObsega()
  - Javno operacijo IzracunPloscine()
  - Javni operaciji DolociSirino(novaSirina) in DolociVisino(novaVisina)
  - Konstruktor brez parametrov, ki oba atributa postavi na 5
  - Konstruktor s parametri zacetnaSirina in zacetnaVisina
- **Razred Pravokotnik vključite v Kalkulator in dodajte gumba Obseg in Ploščina**
  - Ob pritisku na tipko naj se z uporabo izdelanega razreda izračuna ploščina oziroma obseg za podane parametre
    - Izračun ploščine: konstruktor s parametri
    - Izračun obsega: konstruktor brez parametrov

Vaja 2 - rešitev

```
class Pravokotnik {
        private int sirina;
        private int visina;
        public Pravokotnik() {
                sirina = 5;
                visina = 5;
        }
        public Pravokotnik(int novaSirina, int novaVisina) {
                sirina = novaSirina;
                visina = novaVisina;
        }
        public int IzracunObsega() {
                return 2 * (sirina + visina);
        }
        public int IzracunPloscine() {
                int ploscina = sirina * visina;
                return ploscina;
        }
        public void DolociSirino(int novaSirina) {
                sirina = novaSirina;
        }
        public void DolociVisino(int novaVisina) {
                visina = novaVisina;
        }
}
```

OBVLADOVANJE NAPAK

Blok try-catch-finally (1)

- Programski blok **try-catch-finally** omogoča obvladovanje napak
  - „Lovijo" se napake v bloku **try** (npr. rezerviranje virov)
  - Napake se obravnavajo v bloku **catch**
  - Blok **finally** se izvede vedno (npr. sproščanje virov)

Blok try-catch-finally (2)

```
try
{
      Response.Write("Izvajam sumljivo kodo.<br />");
      int a = 1 / 0;          // primer napake: deljenje z nic (DivisionByZeroException)
      Pravokotnik a;
      a.IzracunPloscine();    // primer napake: neznan objekt (NullReferenceException)
      throw new FormatException(); // primer kako vrzemo izjemo
}
catch (NullReferenceException e)
{
      Response.Write("Napaka 1: ” + e.Message.ToString());
}
catch (FormatException)
{
      Response.Write(sporociloNapaka2);
}
catch
{
      Response.Write("Neznana napaka.");
}
finally
{
      Response.Write("Zakljucujem z delom.");
}
```

Vaja 3

- Dopolnite Kalkulator tako, da bo z njim mogoče obvladovati napačne vnose v vnosna polja
- Omogočite obvladovanje specifičnih napak
  - Prikaz specifičnih sporočil ob napakah
    - Vnos znaka, ki ni številka;
    - Deljenje z 0;
    - Prazno vnosno polje;