MS Visual Studio in .NET

Primer platforme za razvoj sodobnih IS

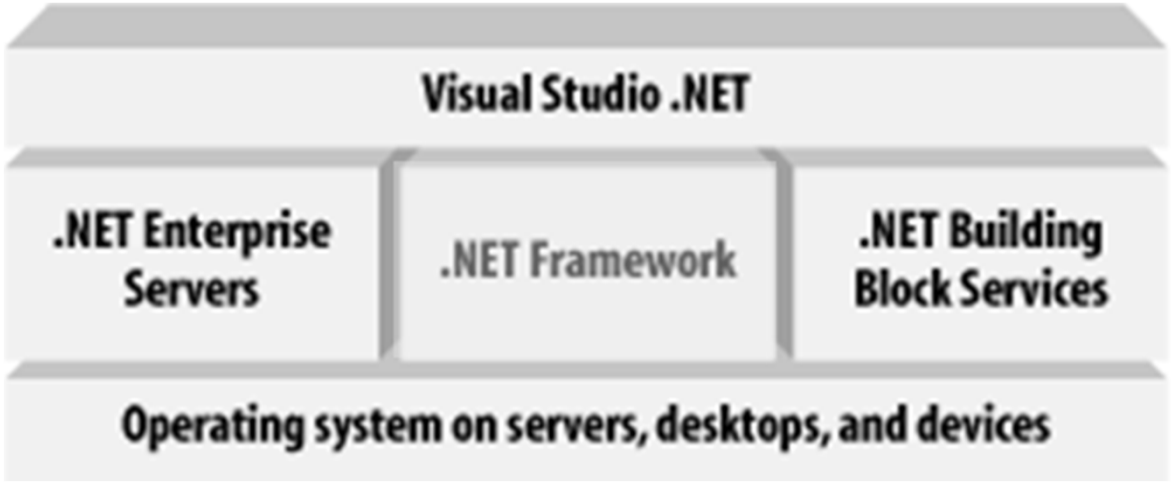
Platforma .NET

- Platformo .NET sestavlja pet osnovnih plasti
  - Operacijski sistemi Microsoft Windows
  - Specializirani strežniki
    - Npr.: Application Center, BizTalk Server, Commerce Server, Exchange Server, Host Integration Server, Internet Security and Acceleration Server, SQL Server
  - Storitve .NET
    - Zagotavljajo jih zunanji ponudniki
    - Običajno jih je mogoče koristiti proti plačilu
    - Npr.: .NET Passport

Platforma .NET

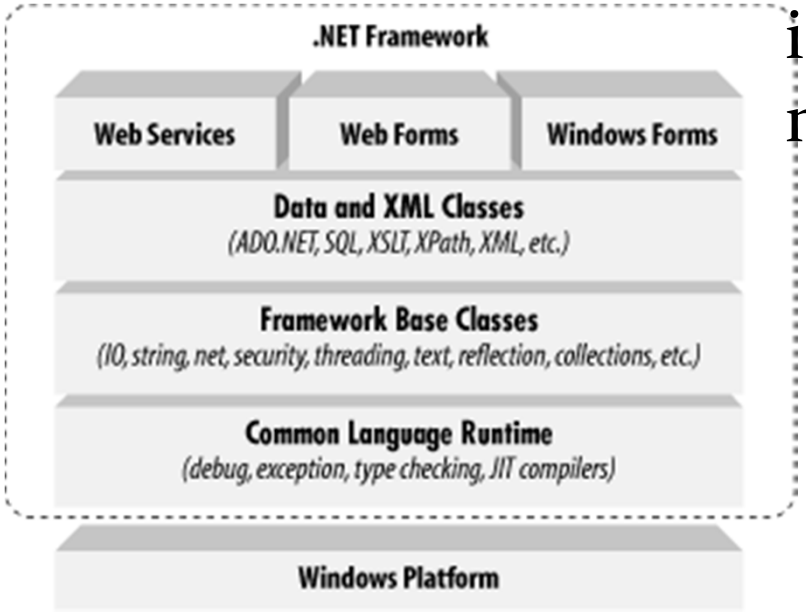
- Platformo .NET sestavlja pet osnovnih plasti
  - Visual Studio
    - Razvojno okolje tipa RAD, ki omogoča razvoj v okolju .NET
  - Ogrodje .NET
    - Zagotavlja infrastrukturo za razvoj in izvajanje aplikacij
    - Vključuje Common Language Runtime
      - Izvajanje programske kode v različnih jezikih
      - Skupno ogrodje razredov, ki jih je mogoče uporabiti v vseh jezikih .NET

Platforma .NET



Arhitektura ogrodja .NET

- Osnovni razredi ogrodja .NET so razvrščeni po imenskih prostorih (ang. namespace)



Pomembni imenski prostori (1)

Imenski prostor	Opis
System	Osnovni razredi, ki jih uporabljajo skoraj vsi programi. Nekateri od razredov: Object, Char, String, Array, Exception,... Vključuje tudi nekatere bolj kompleksne razrede kot npr. GC in AppDomain.
System.IO	Zagotavlja nabor razredov za delo z vhodno-izhodnimi podatkovnimi tokovi (data stream) in razrede za upravljanje datotečnega sistema (kreiranje, urejanje, brisanje datotek). Vključuje razrede kot: FileStream, MemoryStream, Path in Directory.

Pomembni imenski prostori (2)

Imenski prostor	Opis
System.Collections	Vključuje nabor razredov, ki omogočajo upravljanje z zbirkami objektov. Primeri: ArrayList, DictionaryBase, Hashtable, Queue, and Stack.
System.Threading	Vključuje nabor razredov, ki podpirajo večnitno programiranje. Primeri: Thread, ThreadPool, Mutex in AutoResetEvent.
System.Reflection	Vključuje nabor razredov, ki podpirajo dinamično povezovanje in preverjanje tipov. Primeri: Assembly, Module in MethodInfo.

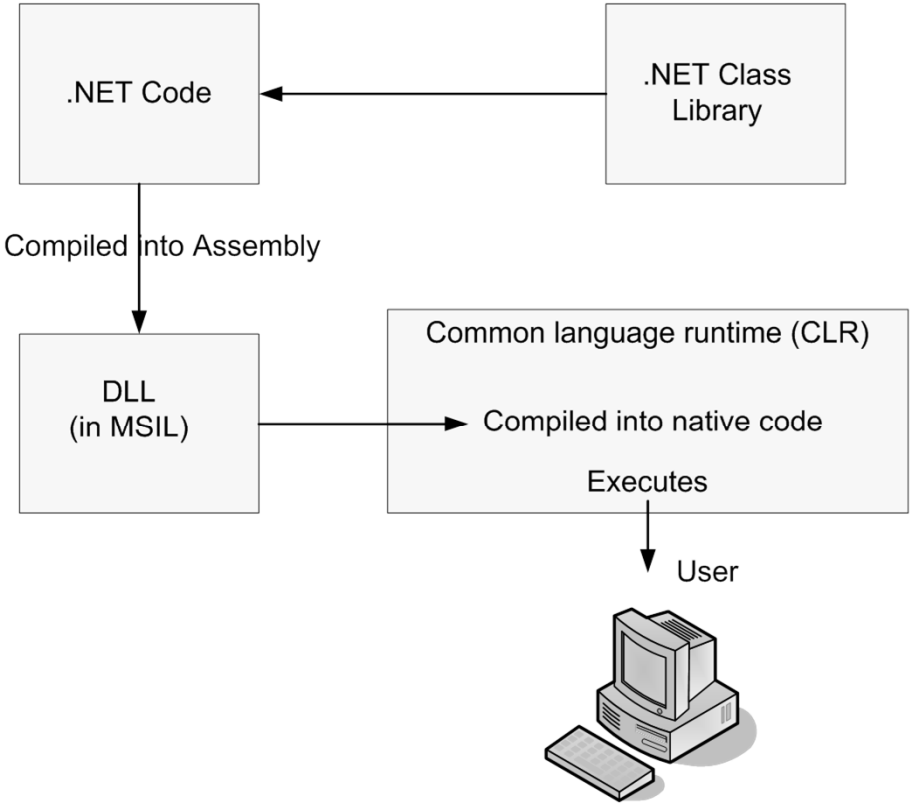
Pomembni imenski prostori (3)

Imenski prostor	Opis
System.Security	Vključuje nabor razredov in pripadajočih imenskih prostorov za zagotavljanje varnosti. Primeri imenskih prostorov: Cryptography, Permissions, Policy in Principal.
System.Net	Vključuje nabor razredov in pripadajočih imenskih prostorov za podporo omrežja. Primeri razredov: IPAddress, Dns, and HttpWebRequest.
System.Data	Vključuje razrede za delo s podatki - ADO.NET.

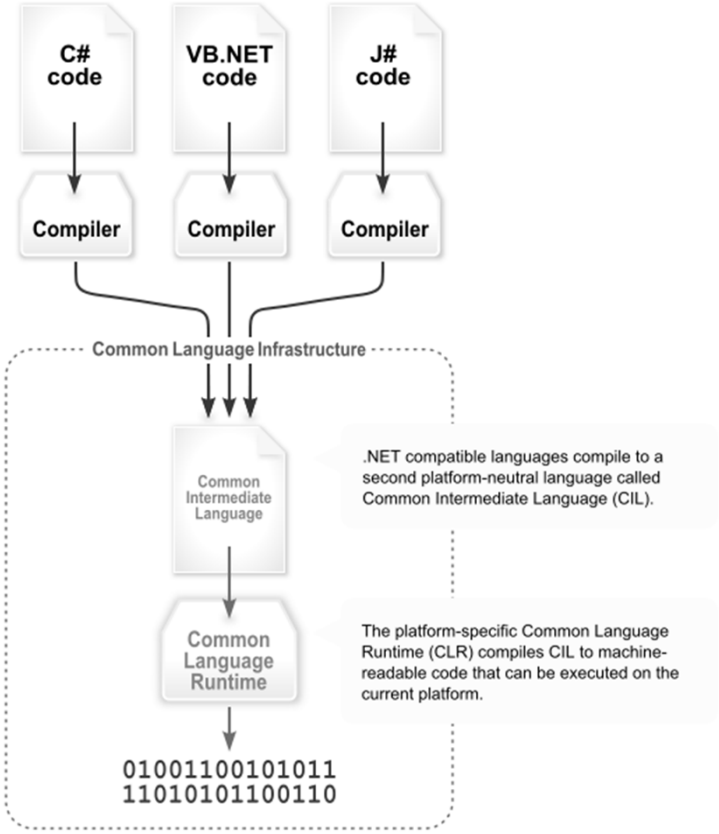
Pomembni imenski prostori (4)

Imenski prostor	Opis
System.Web.Services	Vsebuje razrede za delo s spletnimi storitvami.
System.Web.UI	Vsebuje razrede za delo z ASP.NET spletnimi aplikacijami.
System.Windows.Forms	Vsebuje razrede za zaslonske vmesnike klasičnih aplikacij v Windows.

Izvajanje aplikacije .NET



Prevajanje aplikacije .NET



Visual Studio in programski jezik C#

Visual Studio

- Objektno usmerjeno programiranje
- Uporaba komponent in gradnikov (ang. control)
- Vizualno oblikovanje GUI
- Dogodkovno usmerjeno programiranje
- Aplikacije za Windows in spletne aplikacije (.aspx)

Programski jezik C#

- Objektno usmerjen jezik
- Sintaksa podobna C++ in Javi
- Občutljiv na velikost črk
  - ois ≠ OiS

Programski jezik C#

- Spletno iskanje „C sharp“
- Vsebina
  - Obvladovanje napak
  - Objektno usmerjeno programiranje
  - Podatkovna polja
  - Upravljanje z viri

Objektno usmerjeno programiranje

Razred (1)

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

Razred (2)

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

Vidljivost (1)

- Vidljivost atributov in operacij
  - private (privzeto): do zasebne operacije/atributa je mogoče dostopati le znotraj razreda
  - public: do javne operacije/atributa je mogoče dostopati tako znotraj kot izven razreda
  - protected: do zaščitene operacije/atributa je mogoče dostopati le iz naslednikov razreda (dedovanje)
- Vidljivost je mogoče določiti tudi za celoten razred
  - Razredi znotraj drugih razredov

Vidljivost (2)

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

Konstruktor (1)

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

Konstruktor (2)

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

Prekrivanje operacij (1)

- 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

Prekrivanje operacij (2)

```
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 (1)

- 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 novega objekta (2)

```
// 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 (1)

- 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);
```

Uporaba imenskih prostorov (2)

```
// Nekateri imenski prostori, ki jih uporabljamo na vajah
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Data.SqlClient;
```

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
        }

        // Latnosti 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;
          }
}
```

Dedovanje (4)

```
// 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);

MessageBox.Show(k.Obseg().ToString());
MessageBox.Show(k.Povrsina().ToString());
```

Abstraktni razredi (1)

- Namen
  - Implementacija skupne funkcionalnosti
- Lahko ima abstraktne operacije
- Ni mogoče kreirati objektov
  - Tudi brez abstraktnih operacij
- Dedovanje v …
  - Abstraktni podrazred
  - Navaden podrazred
    - Prepisati (ang. override) vse abstraktne operacije

Abstraktni razredi (2)

```
abstract class KrogAbstr
{

        // Abstraktne operacije
        abstract public double Povrsina();

        // Preostale lastnosti in operacije
        protected double radij;
        public KrogAbstr(double zacetniRadij)
        {
                radij = zacetniRadij;
        }
        public double Obseg()
        {
                return 2 * Math.PI * radij;
        }
}
```

Abstraktni razredi (3)

```
// Dedovanje iz abstraktnega razreda
class Krog : KrogAbstr
{

          public Krog(double zacetniRadij) : base(zacetniRadij) { }

          // Prepis abstraktnih operacij
          override public double Povrsina()
          {
                    return Math.PI * radij * radij;
          }
}

Krog k = new Krog(1);

MessageBox.Show("Obseg: " + k.Obseg());
MessageBox.Show("Obseg: " + k.Povrsina());
```

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
{
          …
}
```

Cast

- Pretvorba nekega podatkovnega tip v drugega
- 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"
MessageBox.Show(((Krog)element).Povrsina().ToString());
```

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
{
          Console.WriteLine("Executing the try statement.");
          throw new NullReferenceException();
}
catch (NullReferenceException e)
{
          Console.WriteLine("{0} Caught exception #1.", e);
}
catch
{
          Console.WriteLine("Caught exception #2.");
}
finally
{
          Console.WriteLine("Executing finally block.");
}
```

Primeri razredov Exception (1)

Razred	Opis
System.ArithmeticException	A base class for exceptions that occur during arithmetic operations, such as System.DivideByZeroException and System.OverflowException.
System.ArrayTypeMismatchException	Thrown when a store into an array fails because the actual type of the stored element is incompatible with the actual type of the array.

Primeri razredov Exception (2)

Razred	Opis
System.DivideByZeroException	Thrown when an attempt to divide an integral value by zero occurs.
System.IndexOutOfRangeException	Thrown when an attempt to index an array via an index that is less than zero or outside the bounds of the array.
System.InvalidCastException	Thrown when an explicit conversion from a base type or interface to a derived type fails at run time.

Primeri razredov Exception (3)

Razred	Opis
System.NullReferenceException	Thrown when a null reference is used in a way that causes the referenced object to be required.
System.OutOfMemoryException	Thrown when an attempt to allocate memory (via new) fails.
System.OverflowException	Thrown when an arithmetic operation in a checked context overflows.

Primeri razredov Exception (4)

Razred	Opis
System.StackOverflowException	Thrown when the execution stack is exhausted by having too many pending method calls; typically indicative of very deep or unbounded recursion.
System.TypeInitializationException	Thrown when a static constructor throws an exception, and no catch clauses exists to catch it.

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 vedno objekt
    - Ni res!
  - 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;

// Katere definicije so pravilne?
int[] polje4 = new int[3] { 9, 3, 7, 2 };
int[] polje5 = new int[4] { 9, 3, 7 };
int[] polje6 = new int[4] { 9, 3, 7, 2};
```

Dostop do elementov polja (1)

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

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

Dostop do elementov polja (2)

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

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

Zbirke (1)

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

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())
                    MessageBox.Show(element.ToString());
          else if (element.GetType() == krog.GetType())
                    MessageBox.Show(((Krog)element).Povrsina().ToString());
```

Zbirke – Queue (1)

```
using System.Collections;
…
string niz = "Niet";
Krog krog = new Krog(5);

Queue q = new Queue();

q.Enqueue(niz);
q.Enqueue(krog);

foreach (object element in q)
          if (element.GetType() == niz.GetType())
                    MessageBox.Show(element.ToString());
          else if (element.GetType() == krog.GetType())
                    MessageBox.Show(((Krog)element).Povrsina().ToString());

q.Dequeue(krog);
```

Zbirke – Queue (2)

```
using System.Collections.Generic;
…
string niz = "Jan Plestenjak";
Krog krog = new Krog(5);

// Vrsta točno določenega tipa (razreda) elementov
Queue<Krog> q = new Queue<Krog>();

q.Enqueue(niz);     // Napaka
q.Enqueue(krog);

foreach (Krog k in q)
{
          MessageBox.Show(k.Povrsina().ToString());
}

q.Dequeue(krog);
```

Zbirke – SortedList

```
string niz = "Laibach";
Krog krog = new Krog(5);

SortedList sl = new SortedList();

sl.Add(1, niz);
sl.Add(2, krog);

for (int i = 0; i < sl.Count; i++)
  if (sl.GetByIndex(i).GetType() == niz.GetType())
  {
    MessageBox.Show(sl.GetByIndex(i).ToString());
  }
  else if (sl.GetByIndex(i).GetType() == krog.GetType())
  {
    MessageBox.Show(((Krog)sl.GetByIndex(i)).Povrsina().ToString());
  }
```

Zbirke in gradniki

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

```
listBox1.Items.Clear();
listBox1.Items.Add("OiS");
```

Upravljanje z viri

Smetar (ang. garbage collector)

- Objekti ob kreiranju zasedejo del pomnilnika
  - Ob uničenju se sprosti pomnilnik
- Objektov ni mogoče uničiti neposredno
  - Ob izhodu iz aplikacije se uničijo vsi objekti
  - Objekti brez sklicev se uničijo ob prvem naslednjem zagonu smetarja (upravljavca pomnilnika)
- Smetar
  - Samodejni zagon po potrebi
    - Sproščanje virov
  - Eksplicitni zagon: npr. System.GC.Collect();

Spletne aplikacije

Vsebina (1)

- Uvod v spletne aplikacije
- ASP.NET
  - Uvod v ASP.NET
  - Življenjski cikli, dogodki in pomembni objekti
  - Hranjenje in prenos stanja
  - Varnost
  - Oblikovanje (CSS)

Vsebina (2)

- ASP.NET in ADO.NET
  - Razred DataReader
  - Razred DataSet
  - Gradnik SqlDataSource

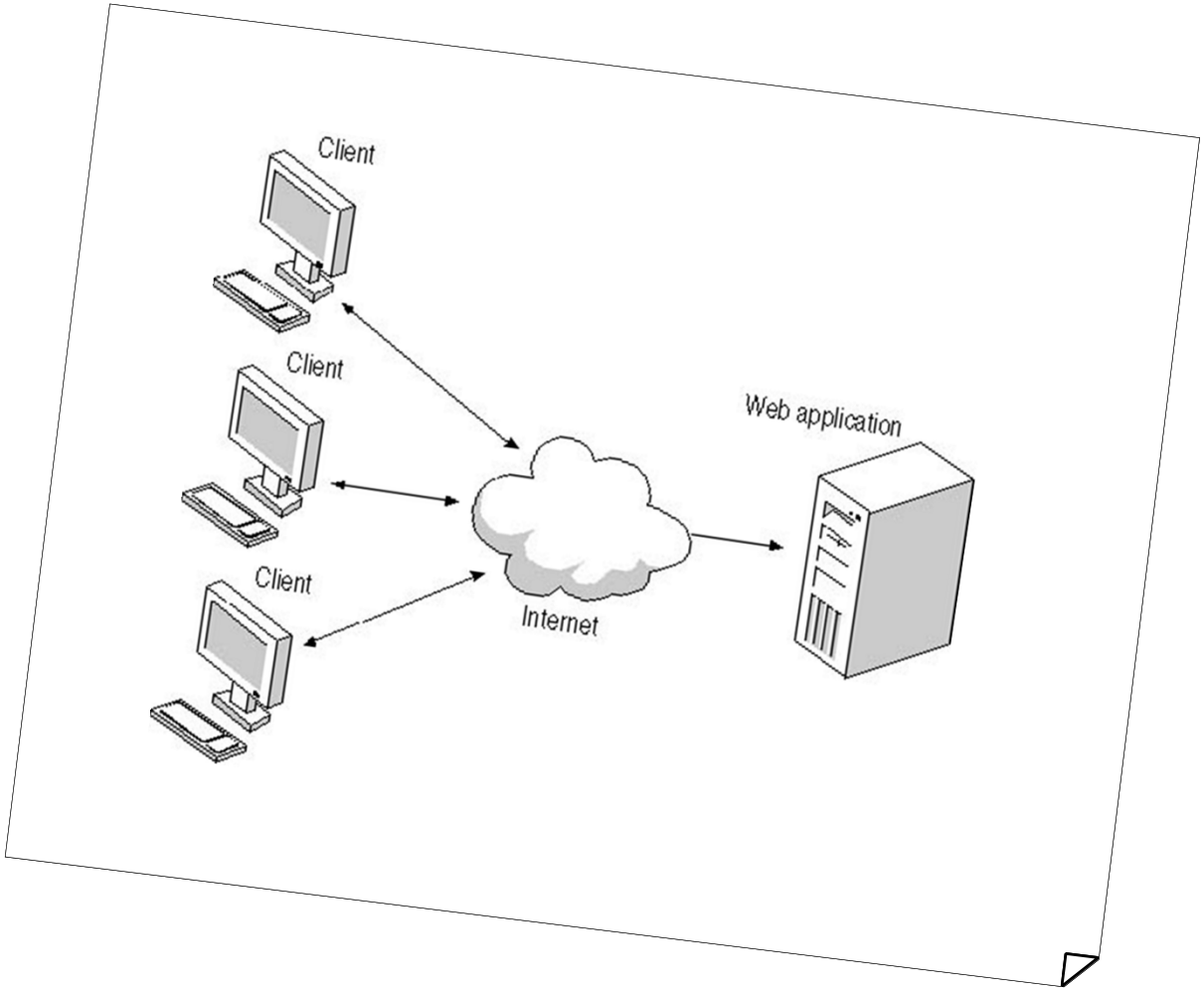
Uvod v spletne aplikacije

Tipi aplikacij z dostopom do spleta

- Spletne aplikacije
  - Aplikacije na strežniku, dostop z brskalnikom
- Spletne storitve
  - Storitve na strežniku, aplikacijski dostop
- Aplikacije za Windows
  - Dostop do aplikacij ali storitev na spletu (prijava, pomoč, posodobitve ipd.)
- Aplikacije P2P
  - Medsebojna komunikacija preko spleta

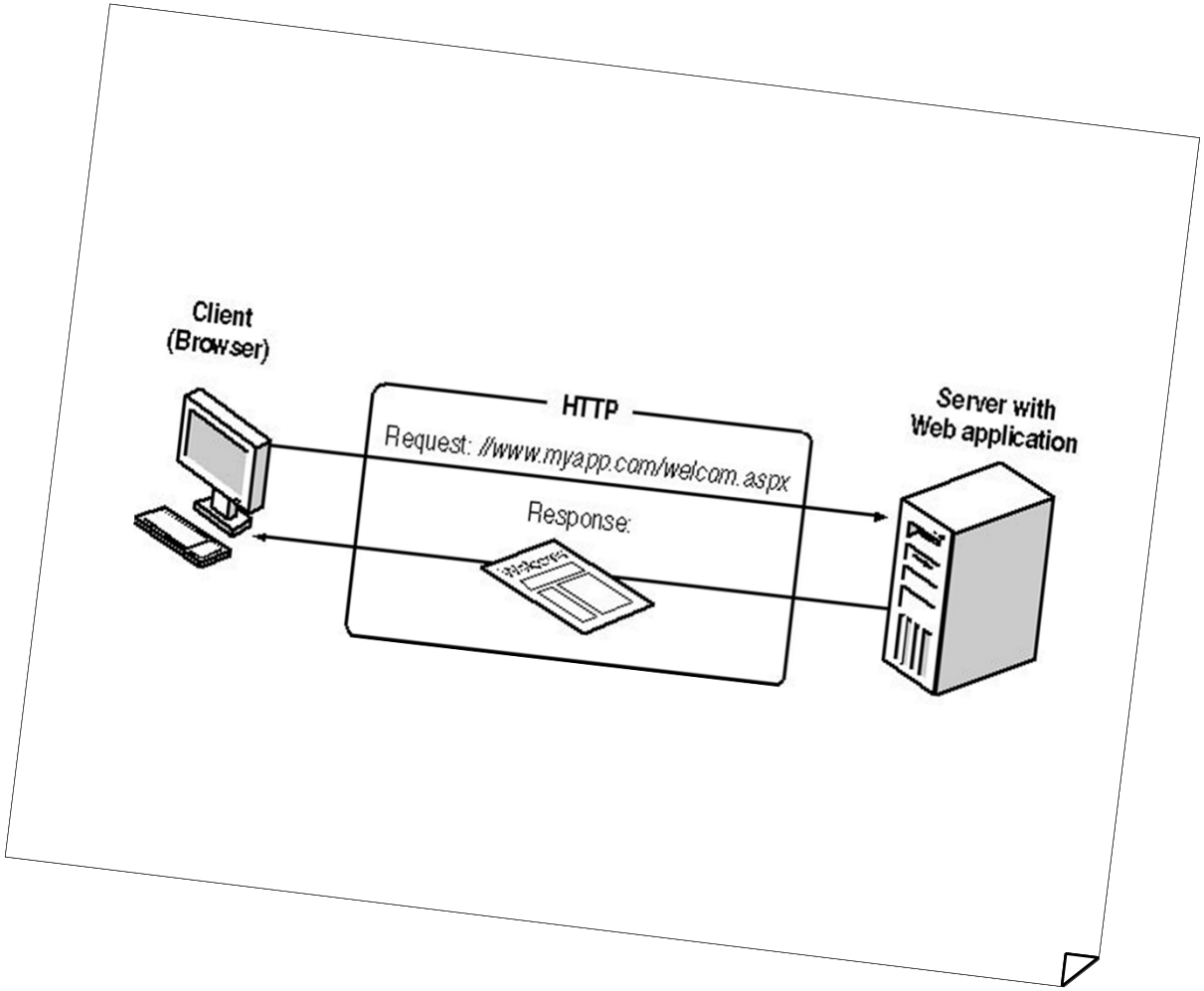
Delovanje spletnih aplikacij (1)

Spletna aplikacija je nameščena na spletnem strežniku IIS



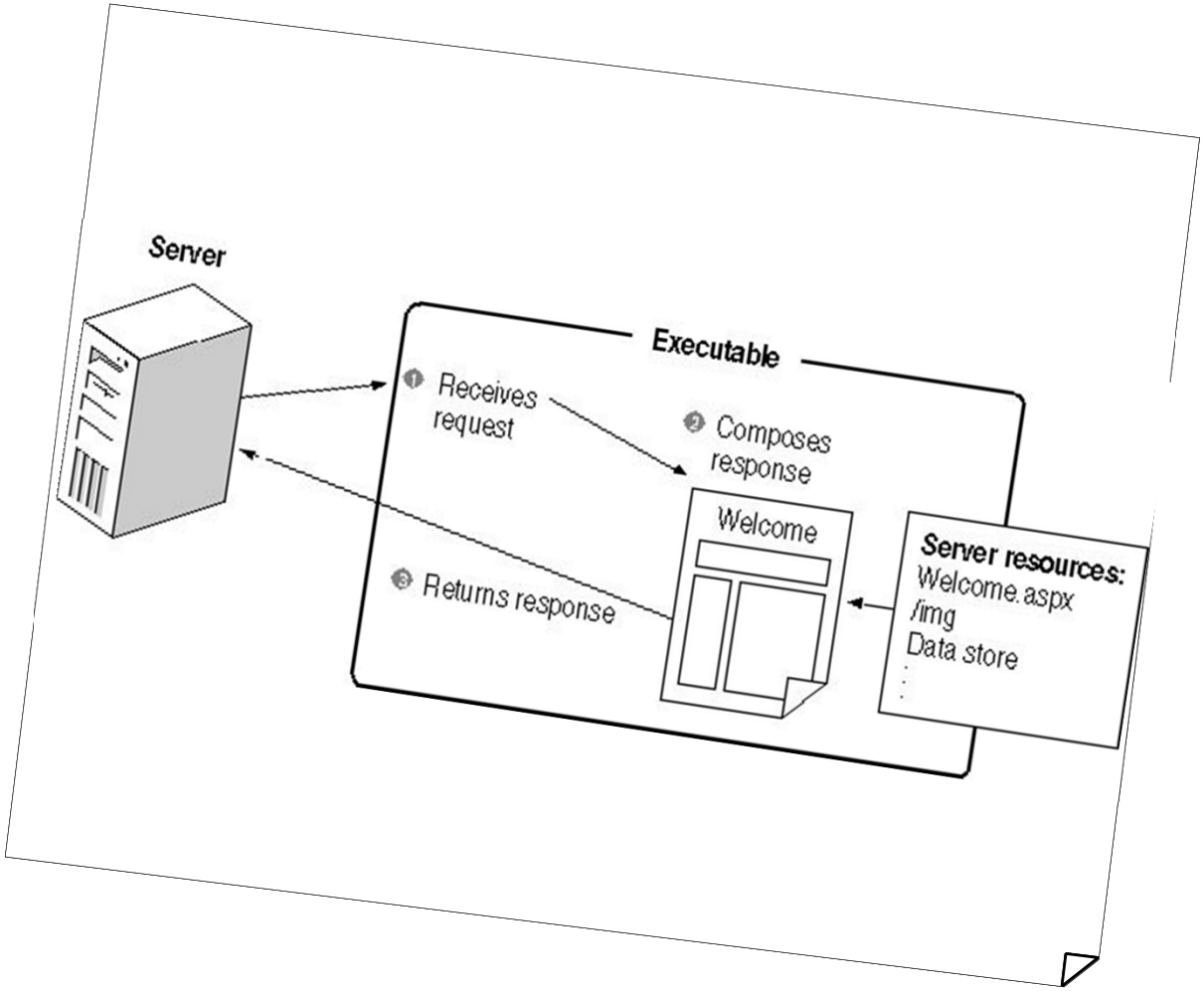
Delovanje spletnih aplikacij (2)

Zahteva in odgovor



Delovanje spletnih aplikacij (3)

Obdelava zahteve na strežniku in oblikovanje odgovora



Aplikacije za Windows vs. spletne aplikacije

Aplikacije za Windows

- „Običajni“ gradniki
- Objekti se uničijo ob izhodu iz aplikacije
- Izvajanje večinoma pri odjemalcu
  - Strežnik za shranjevanje podatkov
- Nameščena programska oprema (odjemalec)

Spletne aplikacije

- Spletni gradniki
  - Prikaz odvisen od brskalnika
- Objekti se uničijo po poslanem odgovoru
- Izvajanje večinoma na strežniku
- Nameščen spletni brskalnik

ASP.NET

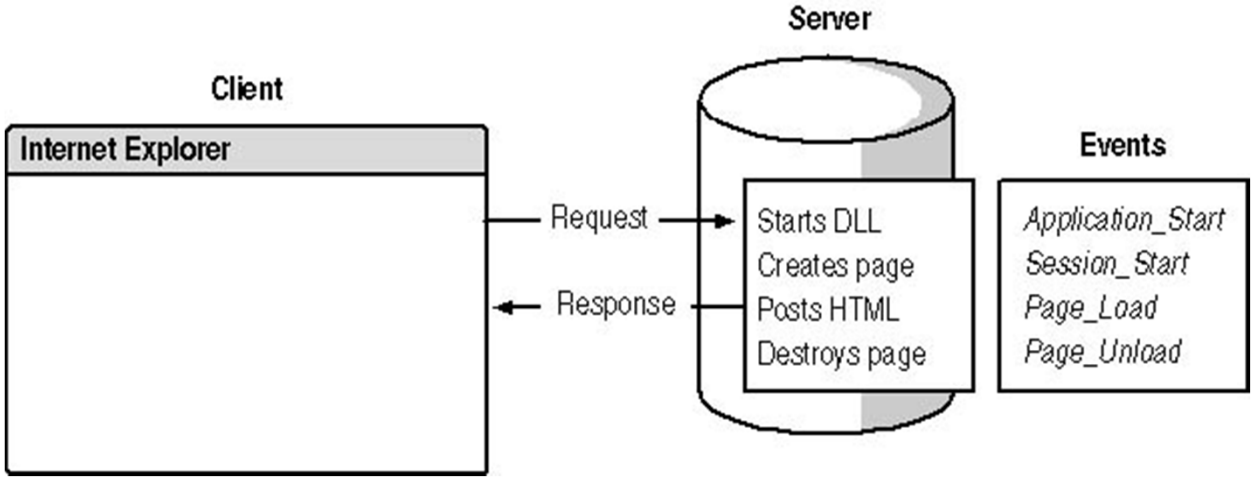
Življenjski cikli, dogodki in pomembni objekti

Spletna aplikacija, seja in spletna stran

- Spletna aplikacija
  - Izvajanje na strežniku
  - Uničenje ko potečejo vse seje
- Seja
  - Kreiranje ob novi povezavi
  - Uničenje po preteku določenega časa (ang. timeout)
- Spletna stran
  - Kreiranje na strežniku
  - Se ne shranjuje (uniči po odgovoru strežnika)
  - Uničijo se tudi vse spremenljivke

Življenjski cikel spletnih aplikacij (1)

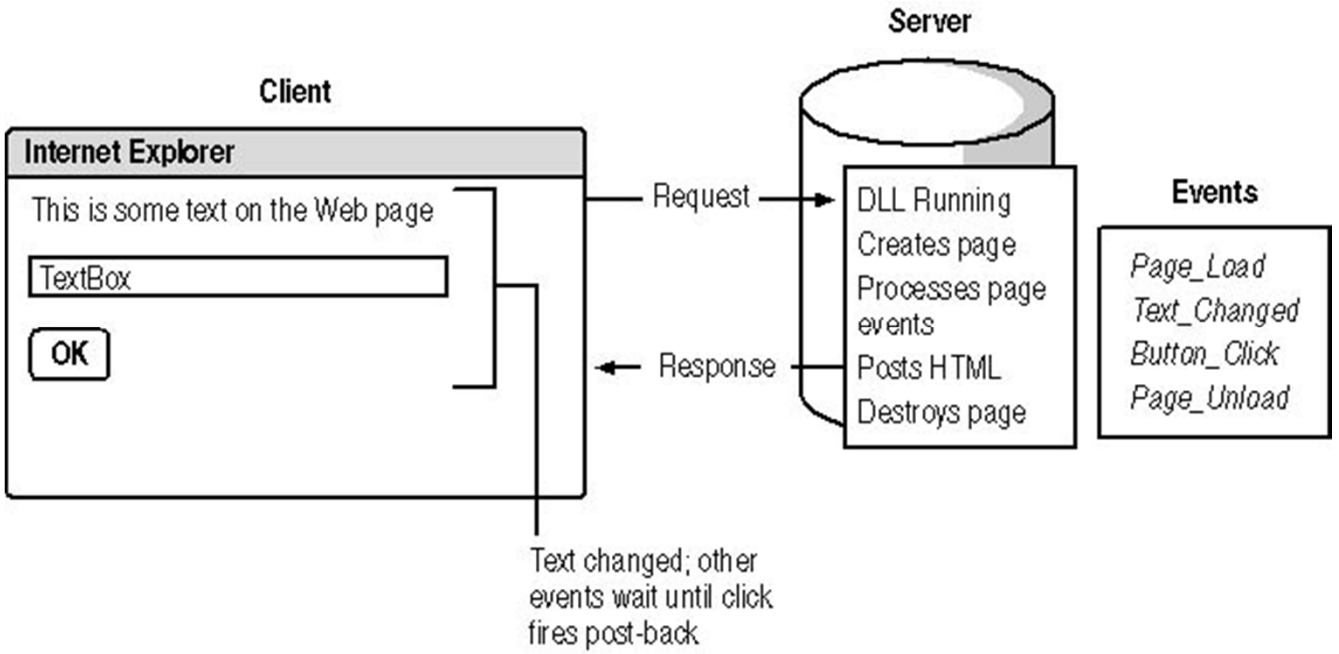
- Zagon spletne aplikacije
  - Datoteka .dll
  - Začetek nove seje
  - Izdelava, pošiljanje in uničenje spletne strani



Življenjski cikel spletnih aplikacij (2)

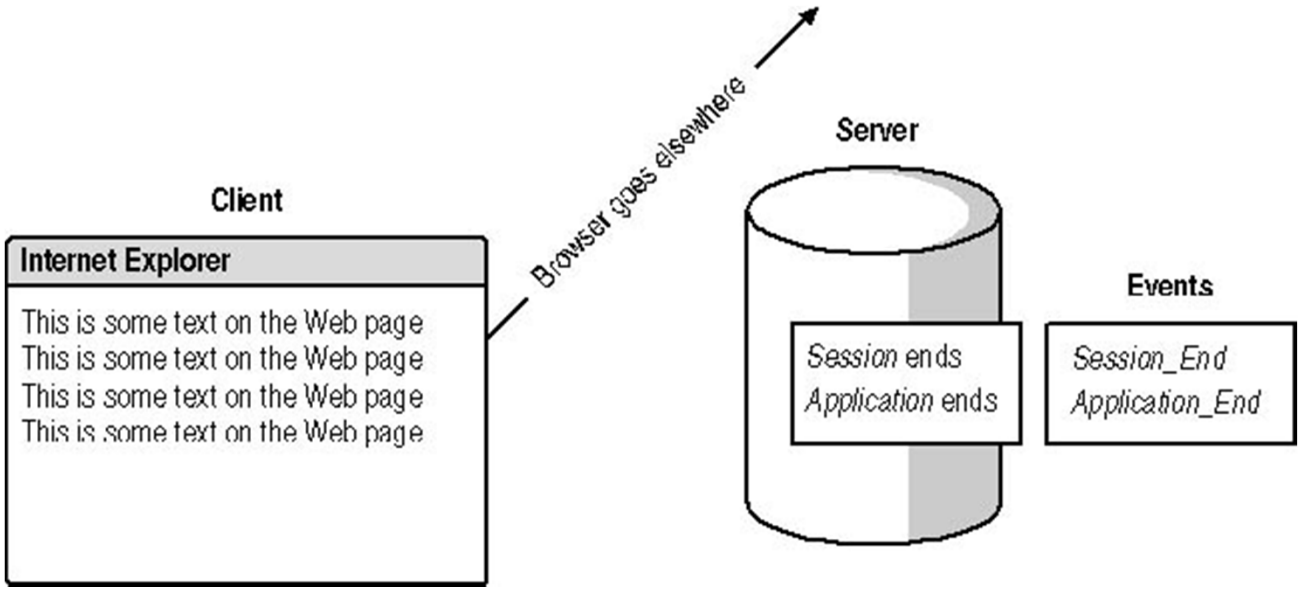
- Delovanje spletne aplikacije
  - Post-back
    - Sproži ga dogodek Button_Click
    - Pošiljanje nove zahteve
  - Vsi ostali dogodki „čakajo“ na post-back
    - Obravnava dogodkov na strani strežnika

Življenjski cikel spletnih aplikacij (3)



Življenjski cikel spletnih aplikacij (4)

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



Življenjski cikel seje

- Kreiranje ob novi povezavi
- Uničenje
  - Timeout
  - Ob klicu Session.Abandon()

Dogodki na spletni strani (1)

Dogodek	Opis
Page_Init	The server controls are loaded and initialized from the Web form' s view state. This is the first step in a Web form' s life cycle.
Page_Load	The server controls are loaded on the Page object. View state information is available at this point, so this is where you put code to change control settings or display text on the page.
Page_PreRender	The application is about to render the Page object.
Page_Unload	The page is unloaded from memory.

Dogodki na spletni strani (2)

Dogodek	Opis
Page_Error	An unhandled exception occurs.
Page_AbortTransaction	A transaction is aborted.
Page_CommitTransaction	A transaction is accepted.
Page_DataBinding	A server control on the page binds to a data source.
Page_Disposed	The Page object is released from memory. This is the last event in the life of a Page object.

Lastnost AutoEventWireup

- Samodejno procesiranje dogodkov Page_* (Page_Init,Page_Load, itd.)
  - V glavi spletne strani (datoteka .aspx)

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

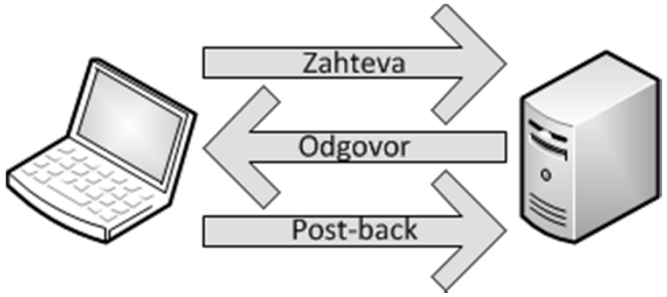
- V primeru AvtoEventWireup=„true“ ni potrebe, da bi eksplicitno zapisali vezavo na operacijo za obravnavo dogodka:

```
private void InitializeComponent() {
          this.Load += new System.EventHandler(this.Page_Load);
}
```

Lastnost IsPostBack

- Uporaba v programski kodi (npr. dogodkih)

```
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
}
```



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

Pomembni objekti

- Application
- Page
- Request
- Response

Objekt Application (1)

Objekt	Opis
Application	Save data items in the Application state.
Context	Get Trace, Cache, Error, and other objects for the current context.
Request	Read a request and get Browser, ClientCertificates, Cookies, and Files objects from the current request.
Response	Write text or data to a response and get Cache, Cookies, and Output objects from the current response.

Objekt Application (2)

Objekt	Opis
Server	Process requests and responses. The Server object provides helper methods for URL encoding and decoding.
Session	Save data items in the Session state.
User	Get authentication information about the user who is making the current request. By default, Web applications allow anonymous access.

Objekt Application (3)

```
if (Request.Browser.MajorVersion < 4)
{
    // Onemogoči napredne lastnosti brskalnika
}

Response.Write("Pošiljam...");
```

Objekt Page (1)

Objekt	Opis
Application	Save data items in the Application state.
Cache	Control how responses are cached on the server.
Controls	Get at controls on the page.
Request	Read a request and get Browser, ClientCertificates, Cookies, and Files objects from the current request.
Response	Write text or data to a response and get Cache, Cookies, and Output objects from the current response.

Objekt Page (2)

Objekt	Opis
Server	Process requests and responses. The Server object provides helper methods for URL encoding and decoding.
Session	Save data items in the Session state.
Trace	Turn tracing on or off and write to the trace log.

Objekt Page (3)

```
// Dodajanje gradnika v obrazec (med <form> in </form>).
TextBox tb = new TextBox();
tb.Text = "Test";

// Deluje, če na strani .aspx ni programske kode
FindControl("Form1").Controls.Add(tb);
```

Objekt Request (1)

Objekt	Opis
Browser	Determine the capabilities of the browser making the request. Browser properties provide the browser version number, determine whether it is the AOL browser, determine whether the browser supports cookies, and supply other information.
ClientCertificates	Authenticate the client.
Cookies	Get information from the client in the form of cookies.

Objekt Request (2)

Objekt	Opis
Files	Get files that are uploaded by the client.
InputStream	Read and write to the raw data sent in the request.
QueryString	...
...	

Objekt Request (3)

```
private void Page_Load(object sender, System.EventArgs e)
{
    if (!IsPostBack)
        // Ali so piškotki vključeni
        if (Request.Browser.Cookies)
            // Ali obstaja piškotek UName
            if ((Request.Cookies["UName"] != null))
                // Dostop do piškotka
                Session["User"] = Request.Cookies["UName"].Value;
}
```

Objekt Response (1)

Objekt	Opis
Cache	Determine how the server caches responses before they are sent to the client.
Cookies	Set the content of cookies to send to the client.
Output	Get or set the raw data returned to the client as the response.
Write	...
Redirect	...
...	

Objekt Response (2)

```
private void Page_Load(object sender, System.EventArgs e)
{
    if (!IsPostBack)

        // Ali so piškotki vključeni
        if (Request.Browser.Cookies)
        {

            // Kreiranje piškotka
            HttpCookie cookUname = new HttpCookie("UName");
            cookUname.Value = "Wombat";

            // Pošiljanje piškotka
            Response.Cookies.Add(cookUname);
        }
}
```

Preusmerjanje

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

```
Response.Redirect("http://www.google.si");
Server.Transfer("Vaja25.aspx");
```

Dinamično dodajanje gradnikov

- Uporaba gradnika PlaceHolder

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

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

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

ASP.NET

Hranjenje in prenos stanja

Hranjenje in prenos stanja

- Samodejen prenos stanja v gradnikih
- Načini prenosa stanja spremenljivk
  - Parametri v naslovu (ang. query string)
  - Piškotki (ang. cookies)
  - Lastnost ViewState
  - Sejna spremenljivka (ang. session state)
  - Aplikacijska spremenljivka (ang. application state)
- Prenos nizov
  - Sejna in aplikacijska spremenljivka tudi prenos objektov

Parametri v naslovu

- 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

- Hranjenje podatkov pri odjemalcu
  - Zavračanje piškotkov
  - Vrednost na voljo po post-back-u

```
// 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-u

```
// 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
  - 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
  - Do konca zadnje seje
  - Vrednost na voljo takoj
  - Vrednost dostopna v vseh sejah

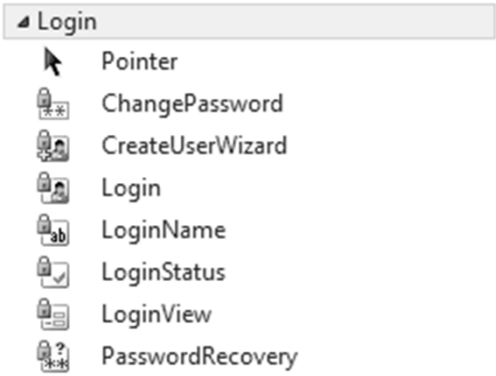
```
// Pisanje v aplikacijsko spremenljivko
Application["Naziv"] = "Grof";

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

ASP.NET

Varnost

Varnost in ASP.NET

- Vgrajen mehanizem za zagotavljanje varnosti
  - Prijava
  - Registracija novega uporabnika
  - Itd.
- Gradniki
- Podatkovna baza
- Upravljanje dostopa v web.config
- http://msdn.microsoft.com/en-us/library/ms178329.aspx



ASP.NET

Oblikovanje (CSS)

Oblikovanje

- Datoteka .css
  - Običajno v master page
  - Enotno oblikovanje celotne spletne aplikacije
- http://www.w3schools.com/css/
- Povezava preko HTML elementa <link>

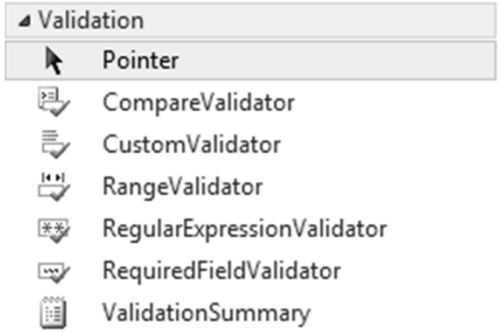
```
<link href="~/Styles/Site.css" rel="stylesheet" type="text/css" />
```

ASP.NET

Validacija vnosa

Validacija vnosa

- Validacija vnosa
  - Gradniki za validacijo
    - Pri odjemalcu
    - Na strežniku



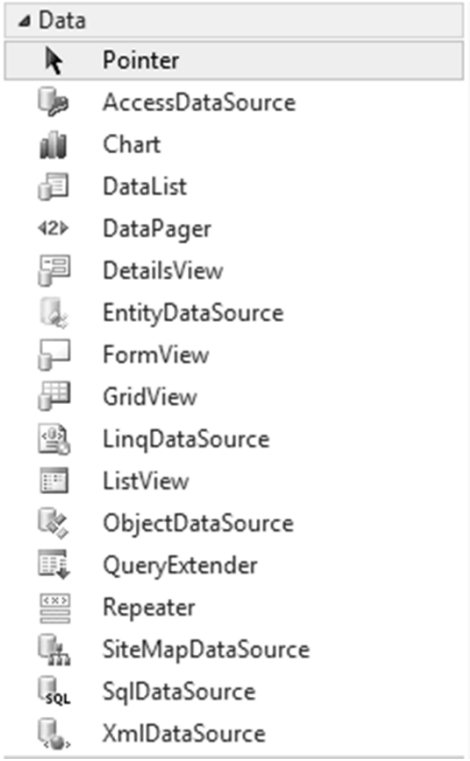
ASP.NET in ADO.NET

Dostop do podatkovne baze

- Načini dostopa
  - Razred DataReader
  - Gradnik SqlDataSource
- Kombiniranje več načinov na isti strani

Gradniki za delo s podatki

- Skupina gradnikov Data
  - GridView
  - DataList
  - DetailsView
  - FormView



Razred DataReader

- Ključni razredi, ki jih uporabljamo za delo z DataReader
  - SqlConnection
  - SqlCommand
  - SqlDataReader
- Posamični stavki SQL
- „Ročno“ dodajanje in upravljanje gradnikov
  - Operacija gradnikov DataBind (!)

Izvajanje stavkov SQL

```
using System.Data.SqlClient;

// Povezava
SqlConnection connection = new SqlConnection(@"data source=.\SQLEXPRESS;Integrated
Security=SSPI;AttachDBFilename=|DataDirectory|\MinIS.mdf;User Instance=true");

// Stavek SQL
SqlCommand cmdUpdateModel = new SqlCommand("UPDATE Model SET Oznaka = @oznaka, FK_Znamka =
@fk_znamka WHERE ID = @id");
cmdUpdateModel.Connection = connection;
cmdUpdateModel.Parameters.AddWithValue("@id", id);
cmdUpdateModel.Parameters.AddWithValue("@oznaka", TextBox1.Text);
cmdUpdateModel.Parameters.AddWithValue("@fk_znamka", DropDownList1.SelectedValue);

// Izvedba
connection.Open();
cmdUpdateModel.ExecuteNonQuery();
connection.Close();
```

Koncept Lookup (1)

```
using System.Data.SqlClient;
using System.Data;

SqlConnection connection = new SqlConnection(@"data source=.\SQLEXPRESS;Integrated
Security=SSPI;AttachDBFilename=|DataDirectory|\MinIS.mdf;User Instance=true");

SqlCommand cmdLookupZnamka = new SqlCommand("SELECT Id, Naziv FROM Znamka");
cmdLookupZnamka.Connection = connection;

// Tabela
DataTable dtZnamka = new DataTable("Znamka");
dtZnamka.Columns.Add(new DataColumn("Id", typeof(int)));
dtZnamka.Columns.Add(new DataColumn("Naziv", typeof(string)));
```

Koncept Lookup (2)

```
// Branje podatkov iz podatkovne baze
connection.Open();
SqlDataReader dr = cmdLookupZnamka.ExecuteReader();

DataRow row;
while (dr.Read())
{
    row = dtZnamka.NewRow();
    row["Id"] = dr["Id"];
    row["Naziv"] = dr["Naziv"];
    dtZnamka.Rows.Add(row);
}
dr.Close();
connection.Close();
```

Koncept Lookup (3)

```
// Povezava gradnika s podatki iz tabele
DropDownList1.DataSource = dtZnamka;
DropDownList1.DataTextField = "Naziv";
DropDownList1.DataValueField = "Id";
DropDownList1.DataBind();

// Usklajevanje izbrane vrednosti s podatki v podatkovni bazi
DropDownList1.SelectedValue = dr["FK_Znamka"].ToString();
```

Gradnik SqlDataSource

- Povezava s podatkovno bazo
  - Ena ali več tabel
- Enostavno povezovanje parametrov z viri
  - Gradniki
  - Parametri v naslovu
  - Piškotki
  - Sejne spremenljivke
  - Itd.