Generički rečnici

Rečnici Dictionary<TKey, TValue>

- Dictionary je kolekcija koja pridružuje ključ (key) odgovarajućoj vrednosti (value).
- Generički tip: Dictionary<TKey, TValue>
- Pristup vrednosti se vrši pomoću ključa, bez pretrage cele kolekcije
- Elementi rečnika su instance strukture KeyValuePair<TKey, TValue>

Svojstva kolekcije Dictionary<TKey, TValue>

Svojstvo	Opis
Count	Broj elemenata (parova ključ–vrednost) u rečniku
Keys	Kolekcija svih ključeva (Dictionary<TKey, TValue>.KeyCollection)
Values	Kolekcija svih vrednosti (Dictionary<TKey, TValue>.ValueCollection)
Item[TKey key] (indekser)	Omogućava pristup ili postavljanje vrednosti po ključu

Metode kolekcije Dictionary<TKey, TValue>

Metoda	Opis
Add(TKey key, TValue value)	Dodaje novi par ključ–vrednost (greška ako ključ postoji)
ContainsKey(TKey key)	Proverava da li rečnik sadrži dati ključ
ContainsValue(TValue value)	Proverava da li rečnik sadrži datu vrednost
Remove(TKey key)	Uklanja element sa datim ključem
TryGetValue(TKey key, out TValue value)	Bezbedno pokušava da vrati vrednost za ključ
Clear()	Briše sve elemente iz rečnika

Štampanje sadržaja rečnika

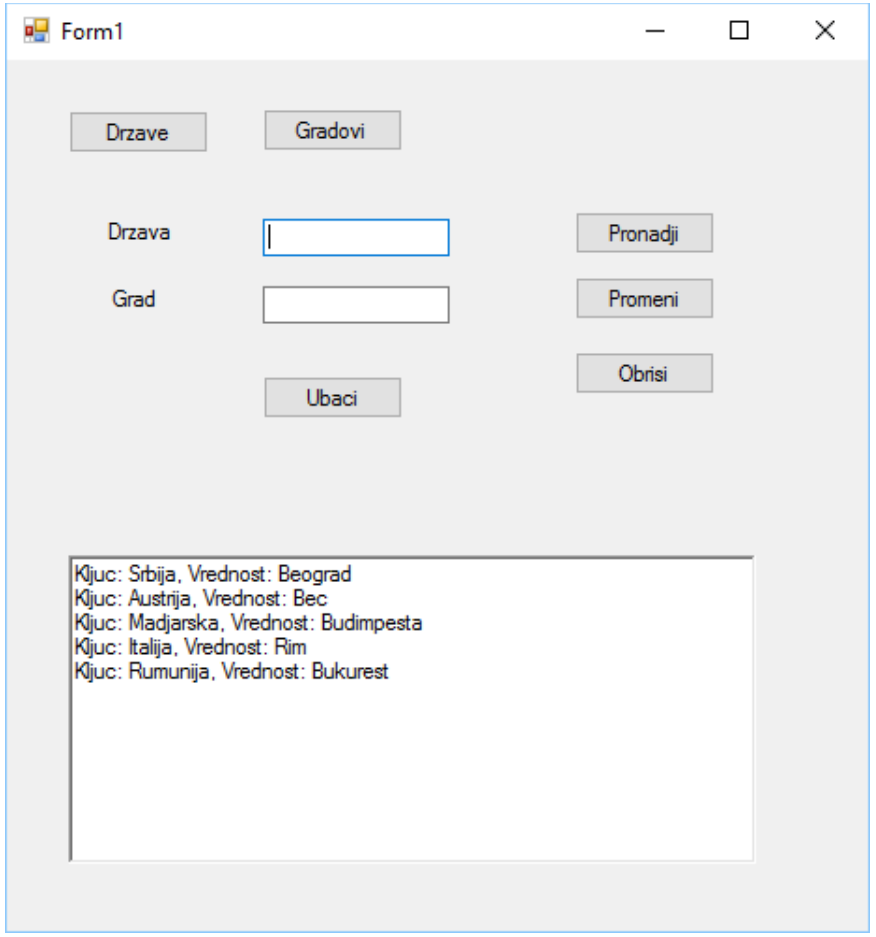
```
public void Stampaj<Tkey,Tvalue>(Dictionary<Tkey,Tvalue> dict)
{
    richTextBox1.Clear();

    foreach (var red in dict)
    {
        richTextBox1.AppendText($"Kljuc: {red.Key}, Vrednost: {red.Value}\n");
    }
}
```

Upotreba kolekcije Dictionary

```
private Dictionary<string, string> recnik = new Dictionary<string, string>();
```

```
private void Form1_Load(object sender, EventArgs e)
{
    recnik.Add("Srbija", "Beograd");
    recnik.Add("Austrija", "Beč");
    recnik.Add("Mađarska", "Budimpešta");
    recnik.Add("Italija", "Rim");
    recnik.Add("Rumunija", "Bukurešt");

    Stampaj(recnik);
}
```

Korisnički interfejs



Pristup ključevima rečnika

```
private void button1_Click(object sender, EventArgs e)
{
    richTextBox1.Clear();

    foreach (string s in recnik.Keys)
    {
        richTextBox1.AppendText(s + "\n");
    }
}
```

Pristup vrednostima rečnika

```
private void button2_Click(object sender, EventArgs e)
{
    richTextBox1.Clear();

    foreach (string s in recnik.Values)
    {
        richTextBox1.AppendText(s + "\n");
    }
}
```

String funkcija

```
public string VelikoPrvoSlovo(string rec)
{
    rec = rec.Trim().ToLower();

    if (rec.Length == 0)
        return string.Empty;

    if (rec.Length == 1)
        return rec.ToUpper();

    return rec.Substring(0, 1).ToUpper() + rec.Substring(1);
}
```

Unos reda u rečnik

```
private void button3_Click(object sender, EventArgs e)
{
    string drzava = VelikoPrvoSlovo(textBoxDrzava.Text);
    string grad = VelikoPrvoSlovo(textBoxGrad.Text);

    if (!recnik.ContainsKey(drzava))
    {
        recnik.Add(drzava, grad);
        Stampaj(recnik);
    }
    else
    {
        MessageBox.Show("Drzava se vec nalazi u recniku");
    }

    textBoxDrzava.Clear();
    textBoxGrad.Clear();
    textBoxDrzava.Focus();
}
```

Pristup vrednosti na osnovu ključa

```
private void buttonPronadji_Click(object sender, EventArgs e)
{
    string drzava = VelikoPrvoSlovo(textBoxDrzava.Text);

    if (recnik.ContainsKey(drzava))
    {
        textBoxGrad.Text = recnik[drzava];
    }
    else
    {
        MessageBox.Show("Drzava se ne nalazi u recniku");
        textBoxDrzava.Clear();
        textBoxDrzava.Focus();
    }
}
```

Brisanje člana rečnika

```
private void buttonObrisi_Click(object sender, EventArgs e)
{
    string drzava = VelikoPrvoSlovo( textBoxDrzava.Text.Trim());

    if (recnik.ContainsKey(drzava))
    {
        recnik.Remove(drzava);
        Stampaj(recnik);
    }
}
```

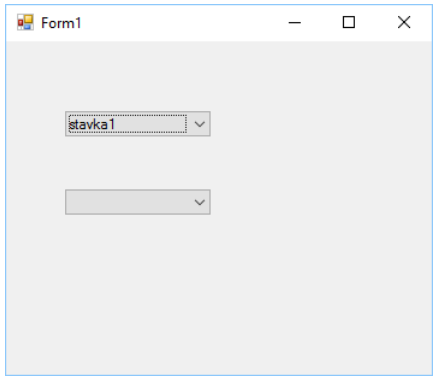
Kontrole ComboBox i ListBox

ComboBox kontrola

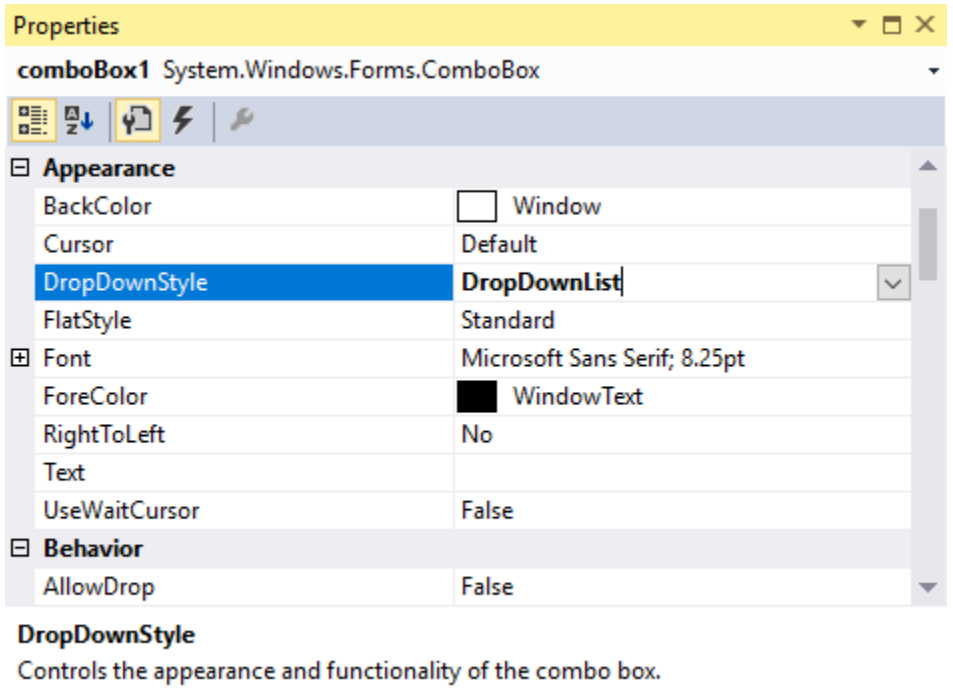
- Svojstvo **Items** predstavlja kolekciju stavki ComboBox kontrole.
- **SelectedIndex** vraća indeks trenutno izabrane stavke
  - int selectedIndex = comboBox1.SelectedIndex;
- **SelectedItem** vraća izabranu stavku, tipa object
  - object selectedItem = comboBox1.SelectedItem;
- Dodavanje stavki u ComboBox:
  - comboBox1.Items.Add(stavka1);
  - comboBox1.Items.AddRange(new object[] { "stavka1", "stavka2", "stavka3" });

Dodavanje stavki iz koda

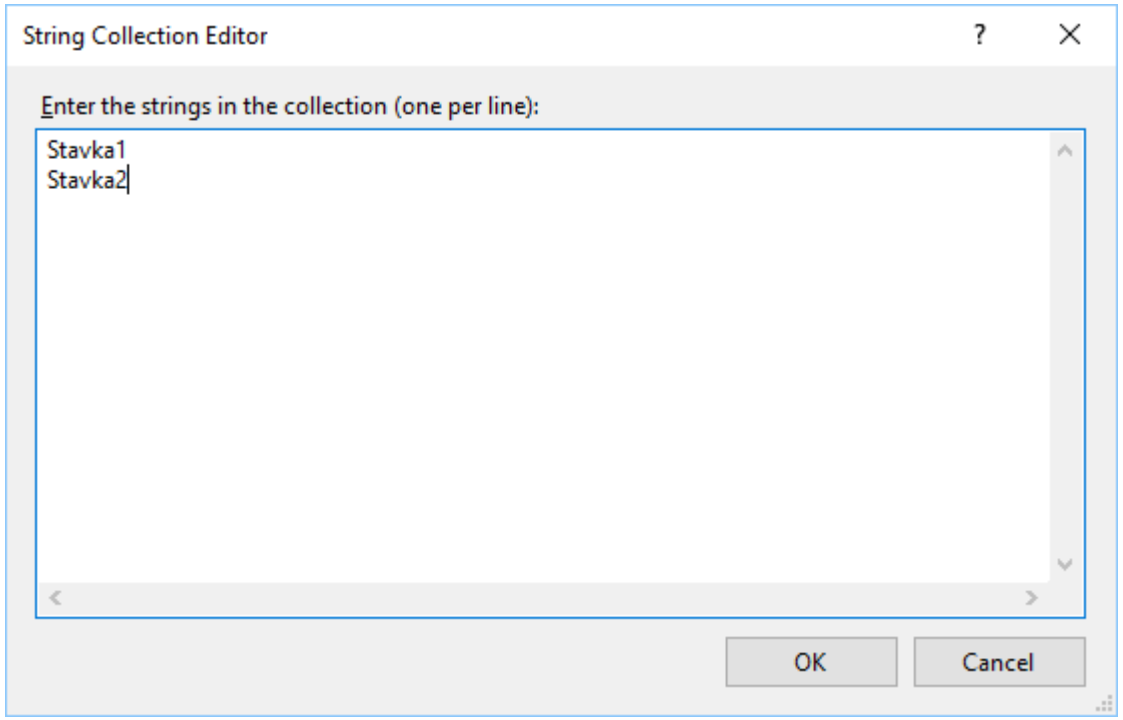
```
private void Form1_Load(object sender, EventArgs e)
{
    comboBox1.Items.Add("stavka1");
    comboBox1.Items.Add("stavka2");
    comboBox1.Items.Add("stavka3");
    comboBox1.SelectedIndex = 0;

    comboBox2.Items.AddRange
        (new object[]{"stavka1","stavka2","stavka3","stavka4"});
}
```



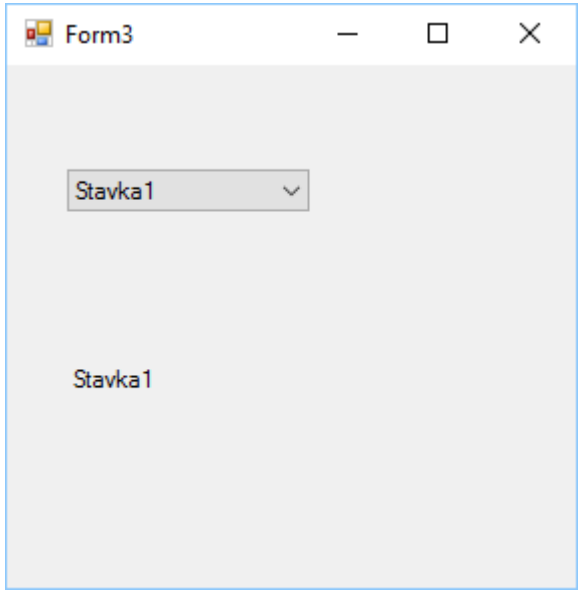
Svojstvo DropDownStyle



Dodavanje stavki u ComboBox u dizajn modu



SelectedIndexChanged događaj

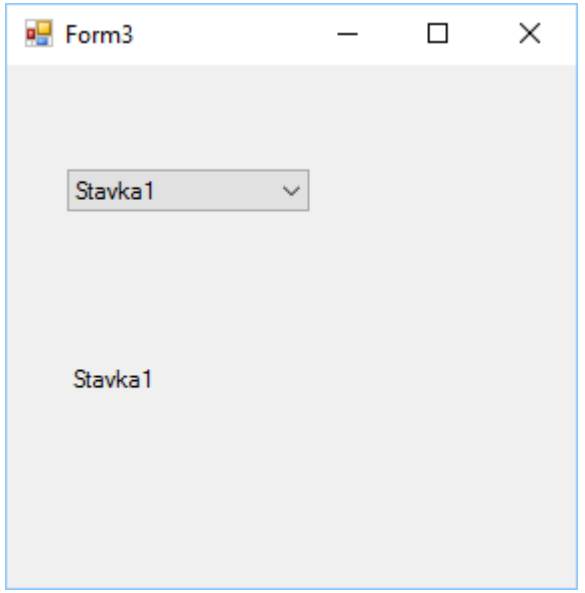



```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    label1.Text = comboBox1.SelectedItem.ToString();
}
```

Povezivanje sa nizom

```
private void Form3_Load(object sender, EventArgs e)
{
    string[] stavke = {"Stavka1", "Stavka2", "Stavka3", "Stavka4"};
    comboBox1.DataSource = stavke;
    comboBox1.SelectedIndex = -1;
}
```

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (comboBox1.SelectedIndex > -1)
    {
        label1.Text = comboBox1.SelectedItem.ToString();
    }
    else
    {
        label1.Text = "";
    }
}
```



Kontrola ComboBox automatski prikazuje prvu stavku niza

Povezivanje sa rečnikom

- Dictionary se ne može direktno koristiti kao izvor podataka (DataSource) za ComboBox
- Razlog je što Dictionary nije lista, već kolekcija parova ključ–vrednost
- ComboBox očekuje izvor podataka koji:
  - ima redosled elemenata
  - ponaša se kao lista stavki
- BindingSource omogućava da se rečnik koristi kao izvor podataka za ComboBox
- BindingSource razlaže rečnik kao kolekciju KeyValuePair<TKey, TValue>, pogodnu za vezivanje
- Svojstvo DisplayMember određuje koje svojstvo objekta će biti prikazano u ComboBox-u (npr. "Value")

Povezivanje sa rečnikom

```
private void Form4_Load(object sender, EventArgs e)
{
    Dictionary<int, string> recnik = new Dictionary<int, string>();
    recnik.Add(1, "Vrednost1");
    recnik.Add(2, "Vrednost2");
    recnik.Add(3, "Vrednost3");

    BindingSource bs = new BindingSource();
    bs.DataSource = recnik;

    comboBox1.DataSource = bs;

    comboBox1.DisplayMember = "Value";

    comboBox1.SelectedIndex = -1;

}
```

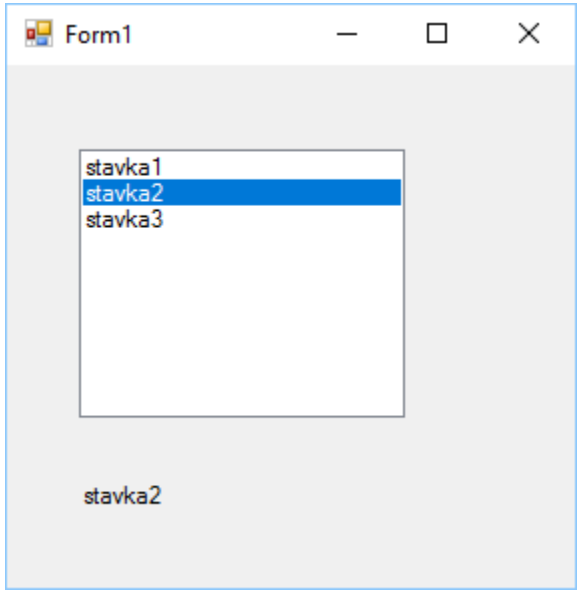
Povezivanje sa rečnikom

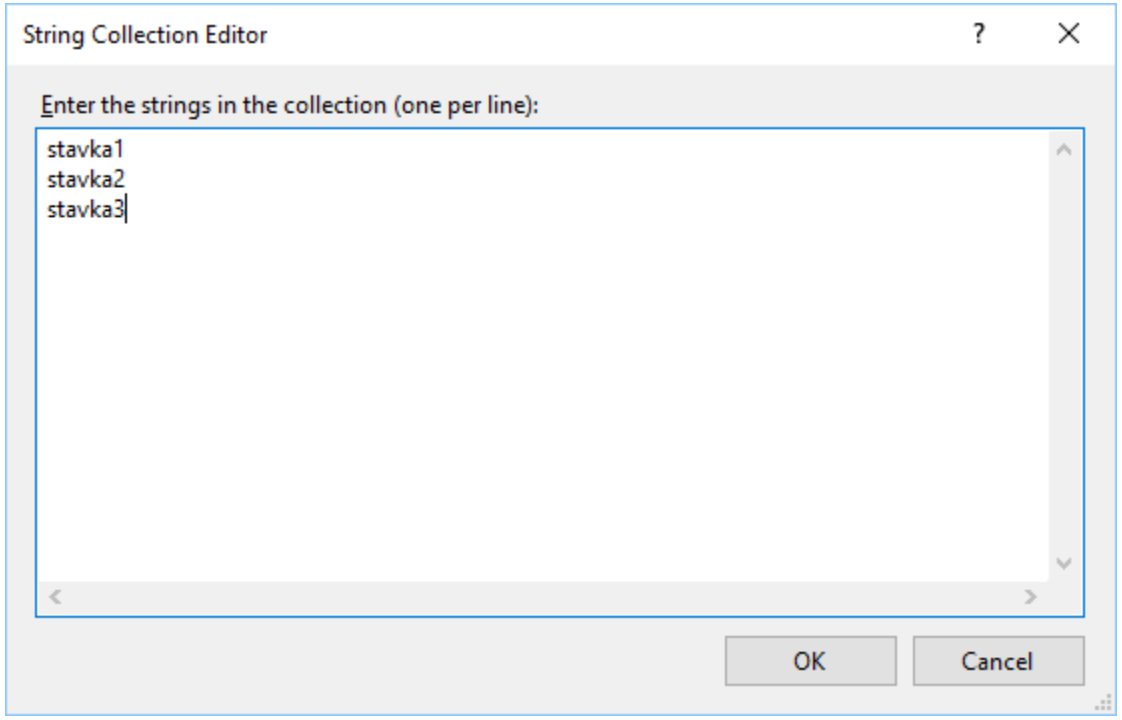
```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (comboBox1.SelectedIndex > -1)
    {
        KeyValuePair<int, string> selektovano =
            (KeyValuePair<int, string>)comboBox1.SelectedItem;

        label1.Text = selektovano.Key + " " + selektovano.Value;
    }
    else
    {
        label1.Text = "";
    }
}
```

Svojstva ListBox kontrole

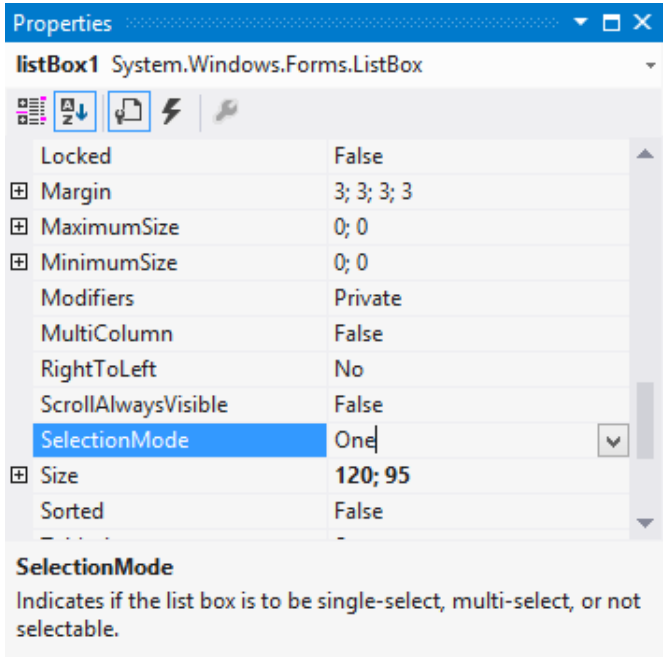
- **SelectedIndices** vraća kolekciju indeksa selektovanih stavki.
- **SelectedIndex** vraća indeks selektovane stavke kada je SelectionMode = One (u suprotnom vraća indeks prve selektovane stavke ili -1 ako nema selekcije)
- **SelectionMode** određuje koliko stavki može biti selektovano u ListBox kontroli:
  - **One** – može se selektovati samo jedna Stavka
  - **MultiSimple** – omogućava selektovanje više stavki (klikom miša)
  - **MultiExtended** – omogućava selektovanje više stavki korišćenjem tastera SHIFT i CTRL
- **SelectedItems** vraća kolekciju selektovanih stavki tipa object
- **SelectedItem** vraća selektovanu stavku tipa object kada je SelectionMode = One

Dodavanje stavki



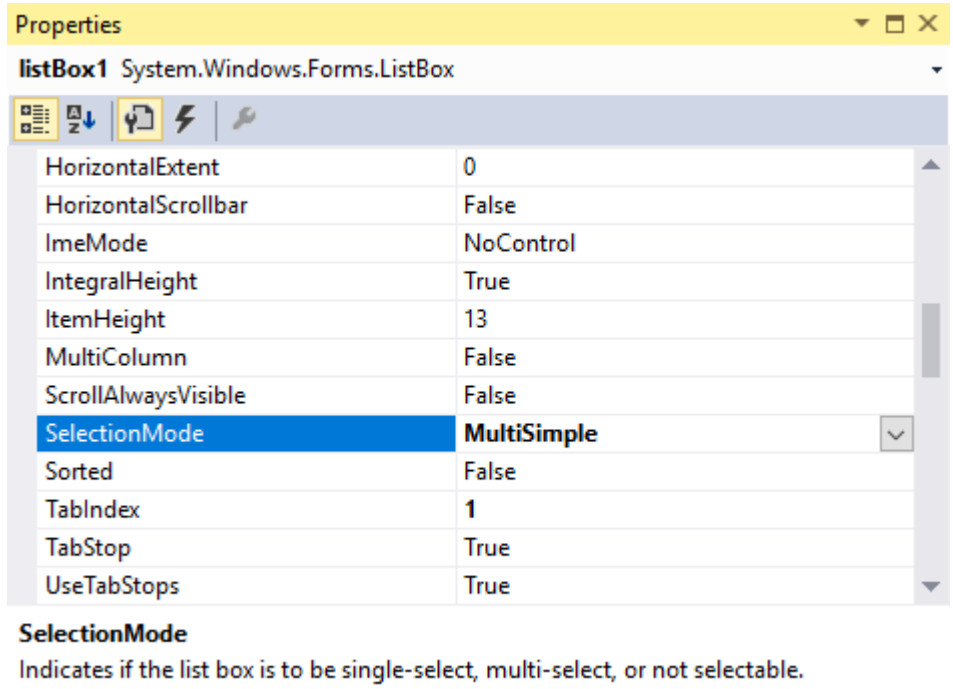


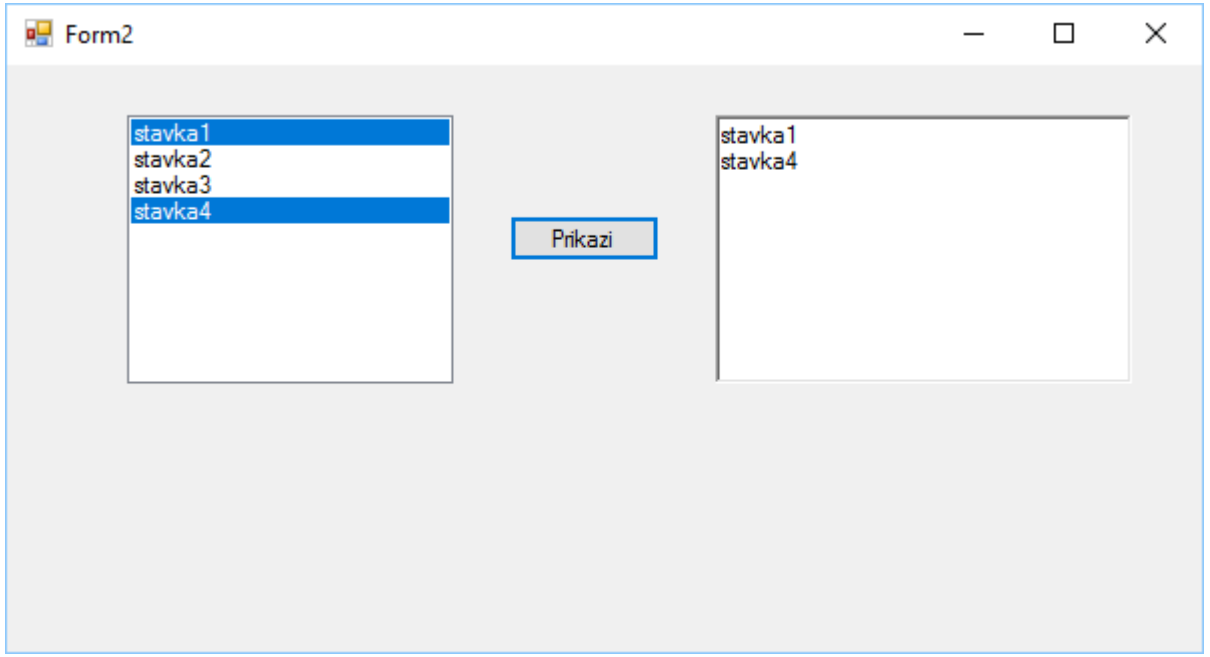
Događaj SelectedIndexChanged




```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    label1.Text = listBox1.SelectedItem.ToString();
}
```

ListBox kontrola - MultiSimple

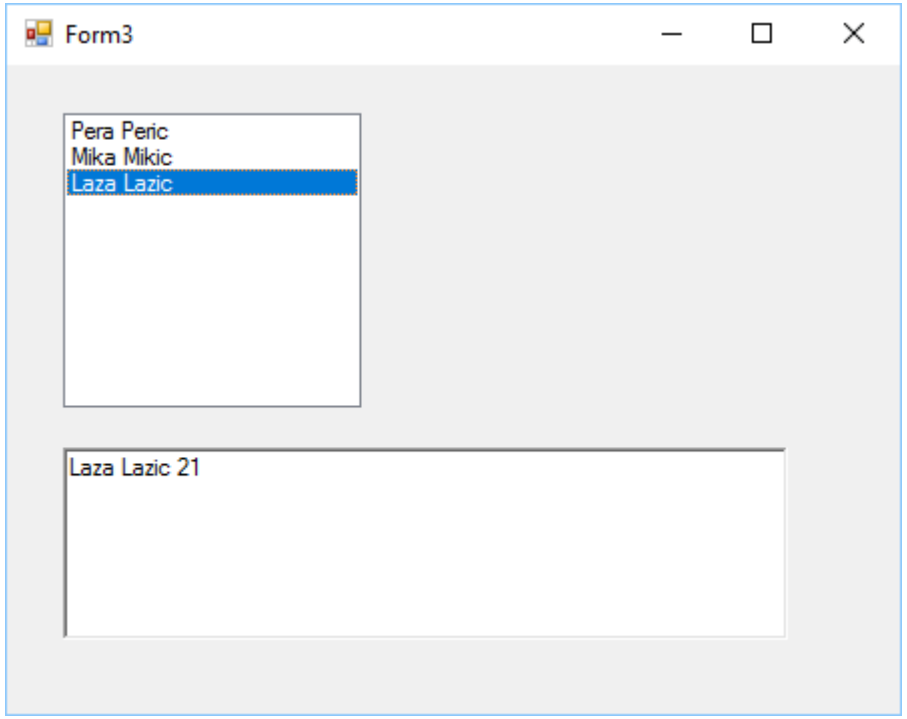




ListBox kontrola sa višestrukom selekcijom

```
private void button1_Click(object sender, EventArgs e)
{
    richTextBox1.Clear();

    foreach (object stavka in listBox1.SelectedItems)
    {
        richTextBox1.AppendText(stavka.ToString() + "\n");
    }
}
```

Korisnički interfejs



Prikaz objekata klase u ListBox kontroli

- Objekti klase mogu se prikazivati u ListBox kontroli, dok se u pozadini čuva ceo objekat
- **DisplayMember** određuje koje svojstvo objekta će biti prikazano u ListBox-u
- Prikaz preko DisplayMember je jasan i fleksibilan način rada sa objektima
- Ako je metoda ToString() redefinisana, ListBox prikazuje njen rezultat.
- Prikaz preko ToString() je jednostavan, ali manje fleksibilan od DisplayMember-a

Klasa Osoba

```
class Osoba
{
    public int OsobaId { get; set; }
    public string Ime { get; set; }

    public string Prezime { get; set; }
    public int Starost { get; set; }

    public string PunoIme
    {
        get
        {
            return Ime + " " + Prezime;
        }
    }
    //public override string ToString()
    //{
    //    return Ime + " " + Prezime;
    //}
}
```

Load procedura forme

```
private void Form2_Load(object sender, EventArgs e)
{
    // 1. Kreiranje liste osoba
    List<Osoba> osobe = new List<Osoba>
    {
        new Osoba { OsobaId = 1, Ime = "Pera", Prezime = "Peric", Starost = 24 },
        new Osoba { OsobaId = 2, Ime = "Mika", Prezime = "Mikic", Starost = 22 },
        new Osoba { OsobaId = 3, Ime = "Laza", Prezime = "Lazic", Starost = 21 }
    };

    // 2. Povezivanje cele liste sa ListBox kontrolom
    listBox1.DataSource = osobe;

    // 3. Definisanje prikaza i vrednosti
    listBox1.DisplayMember = "PunoIme";   // šta se vidi u ListBox-u
    listBox1.ValueMember = "OsobaId";     // šta predstavlja vrednost stavke

    // 4. Bez početne selekcije
    listBox1.SelectedIndex = -1;
}
```

Čitanje selektovane stavke

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (listBox1.SelectedIndex >-1)
    {
        Osoba selektovano = (Osoba)listBox1.SelectedItem;
        richTextBox1.Text = selektovano.Ime + " " + selektovano.Prezime + " " +
        selektovano.Starost;
    }
    else
    {
        richTextBox1.Clear();
    }
}
```

Lambda izrazi

Generički delegat Func

- Delegat je tip podataka koji predstavlja referencu na metodu
- **Func** je generički delegat koji se koristi za pokazivanje na metode koje vraćaju vrednost
- **Func<TResult>** je pokazivač na metodu bez ulaznih parametara koja vraća vrednost tipa TResult
- **Func<T, TResult>** je pokazivač na metodu koja ima jedan ulazni parametar tipa T i vraća rezultat tipa TResult
- **Func<T1, T2, TResult>** je pokazivač na metodu koja ima dva ulazna parametra tipa T1 i T2 i vraća rezultat tipa TResult

Lambda izrazi

- Anonimna funkcija je funkcija koja nema ime, može imati ulazne parametre i može vraćati vrednost
- Lambda izrazi omogućavaju **definisanje anonimnih funkcija**
- Lambda izrazi se definišu korišćenjem operatora =>
- Lambda izrazi imaju sažetu i preciznu sintaksu

Primer Lambda izraza

```
x => x * x
```

Za dato x, izračunaj x * x

Visual C# izvodi povratni tip na osnovu tela metode, konteksta

```
private void Button1_Click(object sender, EventArgs e)
{
    Func<double,double> f1 = x => x * x;

    double rezultat = f1(5);

    MessageBox.Show(rezultat.ToString());
}
```

Lambda izraz za metodu sa dva ulazna parametra

```
private void Button2_Click(object sender, EventArgs e)
{
    Func<int, int, int> f1 = (x, y) => x + y;
    int rezultat = f1(5, 6);

    MessageBox.Show(rezultat.ToString());
}
```

Lambda izraz za metodu bez parametara

```
private void Button3_Click(object sender, EventArgs e)
{
    Func<string> f1 = () => DateTime.Now.ToShortDateString();

    string rezultat = f1();

    MessageBox.Show(rezultat);
}
```

List<T>.ForEach(Action<T>) metoda

```
private void button1_Click(object sender, EventArgs e)
{
    List<string> imena = new List<string>
    {
        "Marko",
        "Ana",
        "Lazar",
        "Dejan"
    };

    imena.ForEach(i => richTextBox1.AppendText(i + "\n"));
}
```

Delegat Action<T> može da pokazuje na metodu sa ulaznim parametrom tipa T koja nema povratnu vrednost

Ekstenzione metode

- Ekstenziona metoda omogućava da dodamo metode postojećoj klasi bez potrebe za nasleđivanjem ili izmenom originalne klase
- To su **statičke metode**, ali se pozivaju kao da su **nestatičke metode** objekta klase koju proširujemo
- Ekstenzione metode moraju biti definisane u **statičkoj klasi**
- Klasa sa ekstenzionim metodama mora biti vidljiva u opsegu koda **koji će je pozivati**
- Prvi parametar ekstenzione metode mora imati ključnu reč **this** ispred tipa koji se proširuje
- Ekstenziona metoda je statička metoda sa najmanje istom vidljivošću kao i klasa koja je sadrži

Primer ekstenzione metode koja proširuje klasu String

```
static class EkstenzioneMetode
{
    public static string VelikoPrvoSlovo(this string s)
    {
        rec = rec.Trim().ToLower();

    if (rec.Length == 0)
        return string.Empty;

    if (rec.Length == 1)
        return rec.ToUpper();

    return rec.Substring(0, 1).ToUpper() + rec.Substring(1);
    }
}
```

Poziv ektenzione metode

```
private void button1_Click(object sender, EventArgs e)
{
    string s1 = "MARKO";
    string ime = s1.VelikoPrvoSlovo();
    richTextBox1.Text = ime;
}
```

Pitanje 1

Neka je comboBox1 instanca kontrole ComboBox. Nakon izvršavanja naredbe:
string selektovano = comboBox1.SelectedItem;

a. promenljiva selektovano dobija vrednost koja je indeks selektovane stavke
b. promenljiva selektovano dobija vrednost koja je vrednost selektovane stavke
c. generiše se izuzetak jer nije izvršeno odgovarajuće kastovanje

Odgovor: c

Pitanje 2

Stavke ComboBox kontrole su tipa:

a. string
b. object
c. double

Odgovor: b

Pitanje 3

Ako želimo da obezbedimo da nijedna stavka ComboBox kontrole ne bude selektovana, treba da postavimo:

a. comboBox1.SelectedIndex = 0;
b. comboBox1.SelectedIndex = -1;
c. comboBox1.SelectedItem = -1;

Odgovor: b

Pitanje 4

Neka je instanca ListBox kontrole pod nazivom listBox1 povezana sa generičkom listom objekata klase Osoba pomoću svojstva DataSource.Ako je PunoIme svojstvo klase Osoba, na koji način se vrednost tog svojstva prikazuje kao tekst stavke u ListBox kontroli?

a. listBox1. BindingMember = "PunoIme";
b. listBox1.DisplayMember = "PunoIme";
c. listBox1.ValueMember = "PunoIme";

Odgovor: b

Pitanje 5

Kreirana je Windows Forms aplikacija u kojoj se koristi ListBox kontrola u režimu jednostruke selekcije (SelectionMode.One).U Load proceduri forme napisan je sledeći kod:

```
private void Form1_Load(object sender, EventArgs e){
string[] stavke = { "stavka1", "stavka2", "stavka3", "stavka4" };
listBox1.DataSource = stavke;
}
```

Pokretanjem forme:

a. biće selektovana prva stavka
b. biće selektovana poslednja stavka
c. neće biti selektovana ni jedna stavka

Odgovor: a

Pitanje 6

Način selekcije stavki u ListBox kontroli definiše se korišćenjem:

a. svojstva SelectionMode
b. svojstva SelectionType
c. svojstva Mode

Odgovor: a

Pitanje 7

Red generičkog rečnika Dictionary<int, string> je:

a. vrednost tipa KeyValuePair<int,string>
b. vrednost tipa string
c. vrednost tipa int

Odgovor: a

Pitanje 8

Za brisanje člana sa ključem **k** iz rečnika **dict** koristi se naredba:

a. dict.Delete(k);
b. dict[k].Delete();
c. dict.Remove(k);

Odgovor: c

Pitanje 9

Ako je definisan sledeći kod:

```
Dictionary<int, string> dict = new Dictionary<int, string>();
int a = 0;
string b = "";
if (dict.ContainsKey(a))
   {
      //???
   }
```

Koja linija koda se može napisati u bloku if?

a. b = dict[a];
b. a = dict[b];
c. a= dict(a);
d. b=dict(a);

Odgovor: a

Pitanje 10

Instanca delegata je referenca na:

a. klasu
b. metodu
c. svojstvo

Odogovor: b

Pitanje 11

Pomoću lambda izraza definiše se:

a. anonimna funkcija
b. sql upit
c. funkcija koja ima svoje ime

Odogovor: a