1. Povezivanje Windows Forms i podataka korišćenjem C# programskog jezika i ADO.NET-a

Ciljevi lekcije

- Upoznati se sa osnovama mehanizma povezivanja baza podataka sa Win. Forms kontrolama

Neophodno predznanje

- Rad u Windows okruženju
- Relacione baze podataka
- Osnove programskog jezika C#
- Osnove ADO.NET-a

1.1. Uvod

Gotovo sve aplikacije koje se mogu nazvati značajnim koriste podatke u nekom obliku. Prilikom kreiranja ovakvih aplikacija neophodno je posedovati efikasne mehanizme za skladištenje, pribavljanje i manipulaciju nad podacima. Ukoliko se pribavljeni podaci prikazuju korisnicima kroz kontrole korisničkog interfejsa aplikacije, neophodno je posedovati intuitivne načine za prikazivanje podataka koji će omogućiti korisnicima da na lak i efikasan način manipulišu podacima.

.NET platforma poseduje predefinisani skup kontrola koje je moguće koristiti za prikazivanje i manipulaciju nad podacima. Ove kontrole ugrađene su u Windows Forms i ASP.NET interfejse za razvoj desktop i Web aplikacija. Ugrađene kontrole omogućavaju projektantima desktop i Web aplikacija relativno jednostavan način kreiranja korisničkog interfejsa koji ima mogućnost povezivanja sa bazom podataka.

Povezivanje korisničkog interfejsa tj kontrola od kojih je korisnički interfejs izgrađen, je jedan od problema koji su projektanti aplikacija u prošlosti rešavali na različite načine. Bilo je neophodno kreirati konekciju ka bazi podataka i očitati podatke, a zatim razviti programski kod koji bi grafički prikazao pribavljene podatke ili popuniti kontrole koje se prikazuju korisnicima. Programski kod razvijen na ovaj način morao je biti različit u zavisnosti od posmatrane situacije i potreba korisnika. Takođe, ukoliko se korisnicima dozvoli mogućnost izmene pribavljenih podataka, bilo je neophodno razviti programski kod koji bi pribavljao izmenjene podatke iz delova korisničkog interfejsa, skladištio te podatke u memorijske strukture na klijentskom računaru i vršio upis izmenjenih podataka u bazu podataka. Ovakav pristup zahtevao je mnogo programskog koda koji se često ponavljao i bio podložan greškama.

Predefinisani skup kontrola ugrađen u .NET platformu enkapsulira prethodno opisani proces i smanjuje količinu programskog koda koji je potrebno razviti. Deo koda smešten je u kontrole sa kojima se povezuju podaci i koje prikazuju podatke, dok je ostatak koda smešten u objekte koji korisnicima nisu vidljivi i koji olakšavaju povezivanje podataka i kontrola. .NET platforma takođe uvodi obrasce po kojima se razvija programski kod kojim se vrši lako povezivanje podataka sa kontrolama koje ih prikazuju korisnicima.

1.2. Povezivanje podataka sa Windows Forms kontrolama

Većinu kontrola ugrađenih u .NET platformu je moguće povezati sa podacima. Ovakvo ponašanje posledica je činjenice da su sve **Windows Forms** kontrole izvedene iz klase **Control** koja implementira **IBindableComponent** interfejs. Svaka kontrola koja implementira **IBindableComponent** interfejs poseduje kolekciju objekata klase **Binding** i naziva se **DataBindings**. Pojedine kontrole poseduju složenije mehanizme povezivanja sa podacima. Dodatna funkcionalnost ogleda se kroz dodatne atribute članove klase poput **DataSource**, **DataMember**, **DisplayMember** i **ValueMember** atribute.

Povezivanje sa podacima u .NET platformi predstavlja mehanizam za automatsko povezivanje i sinhronizovanje podataka koji su uskladišteni u operativnoj memoriji sa kontrolama korisničkog interfejsa koje prikazuju podatke. Korišćenjem ovog mehanizma moguće je povezati kontrolu sa podacima i u potpunosti prepustiti kontroli upravljanje nad prikazom podataka. Prilikom korišćenja ovog mehanizma neophodno je sagledati pravce toka podataka i vremenske trenutke kada se zapravo prenose

podaci. Ukoliko je prenos podataka jednosmeran, podaci se prenose od izvora podataka ka kontroli čiji se atributi popunjavaju pribavljenim podacima. Ukoliko je prenos podataka dvosmeran, izmenjene vrednosti atributa kontrola ažuriraju vrednosti podataka u odgovarajućem izvoru podataka. Većina mehanizama .NET platforme za povezivanje kontrola sa podacima je dvosmerna. Dvosmerni mehanizmi povezivanja kontrola sa podacima omogućava da se ažuriranje pribavljenih podataka vrši automatski tj bez potrebe za razvijanjem dodatnog programskog koda koji bi vršio ažuriranje podataka.

Kako bi mehanizam koji povezuje podatke i kontrole bio pokrenut, neophodno je postojanje programskog koda koji će pokrenuti funkcionisanje mehanizma. Pokretač mehanizma može biti:

- Izvršenje linije koda koja uvodi povezivanje kontrole i podataka
- Izvršenje linije koda koja osvežava kontrolu, izvor podataka ili sam mehanizam povezivanja
- Izvršenje događaja koji je pokrenut zbog promena u izvoru podataka ili u kontroli

Postoje dva primarna oblika povezivanja kontrola .NET platforme sa izvorima podataka: jednostavno povezivanje (eng. simple data binding) i složeno povezivanje (eng. complex data binding).

1.2.1. Jednostavno povezivanje Windows Forms kontrola sa podacima

Jednostavno povezivanje kontrole sa izvorom podataka vrši mapiranje atributa kontrole na atribut izvora podataka. Primer ovakvog povezivanja biće prikazan u nastavku na primeru povezivanja **ComboBox** kontrole sa tabelom RADNIK baze podataka PROBA. **ComboBox** objekat poseduje **DisplayMember** atribut koji određuje koji će atribut iz izvora podataka kontrola koristiti za prikazivanje svake od stavki. Moguće je odrediti i **ValueMember** atribut **ComboBox** objekta. Ovaj atribut sadrži dodatne informacije koje se vezuju za svaku od stavki prikazanih u **ComboBox** objektu. Obično se za svaku prikazanu stavku **ComboBox** objekta vezuje primarni ključ ili referenca na objekat kada bi se svaka od stavki mogla jedinstveno identifikovati u izvoru podataka radi dalje manipulacije nad podacima.

Jednostavno povezivanje kontrola sa podacima

```
ComboBox c = new ComboBox();

MySqlConnection conn = new MySqlConnection();

conn.ConnectionString = "server=localhost;user id=ucenik;password=test;database=proba";

String strSQL = "Select * from RADNIK";

MySqlCommand comm = new MySqlCommand(strSQL, conn);

MySqlDataAdapter adapter = new MySqlDataAdapter(comm);

DataSet ds = new DataSet();

try

{

   conn.Open();

   adapter.Fill(ds, "Radnik");

   conn.Close();
```

```
    c.DataSource = ds.Tables["Radnik"];

    c.DisplayMember = "Ime";

    c.ValueMember = "MatBr";

    c.DropDownStyle = ComboBoxStyle.DropDownList;

}

catch(Exception exc)
{
   //obrada izuzetka
}
```

U ovom primeru **DataSource** atribut **ComboBox** objekta je tabela Radnik koja se nalazi u **DataSet** objektu **ds**. Atributu **DisplayMember ComboBox** objekta je dodeljeno ime atributa (u ovom slučaju ime kolone tabele Radnik) čije će vrednosti biti pročitane iz svakog reda tabele izvora podataka i prikazane u **ComboBox** kontroli kao tekst. Atribut **ValueMember ComboBox** objekta ukazuje na atribut MatBr tabele Radnik, koji je primarni ključ, kako bi odgovarajući red tabele Radnik mogao da bude selektovan kada korisnik izabere neku od stavki **ComboBox** objekta.

1.2.2. Složeno povezivanje Windows Forms kontrola sa podacima

Složeno povezivanje kontrole sa izvorom podataka je mehanizam koji povezuje kontrole sa kolekcijama podataka. Kontrole koje se povezuju sa podacima ovim mehanizmom imaju mogućnost prikaza kolekcija podataka. Najčešće se koriste kontrole koje imaju mogućnost tabelarnog prikaza podataka. **DataGridView** kontrola pripada ovoj grupi kontrola.

DataGridView kontrola pripada **Windows Forms** prostoru imena. Ova kontrola prikazuje podatke koji su vezani za nju u tabelarnom obliku. Ova kontrola sastoji se od redova i kolona pri čemu presek svakog reda i kolone predstavlja ćeliju. Ćelija je osnovna jedinica predstavljanja podataka u kontroli. Izgled i ponašanje ćelije je moguće prilagoditi potrebama aplikacije korišćenjem atributa i događaja koje kontrola poseduje. **DataGridView** poseduje posebne ćelije u zaglavljima redova i kolona. Ćelije smeštene u zaglavljima su posebno grafički naznačene i ukazuju na mod rada kontrole poput sortiranja, ažuriranja i selekcije podataka ili dodavanja novog reda. Ćelije mogu biti različitih tipova čak i u istoj koloni ukoliko podaci nisu vezani za kontrolu, što u našem primeru neće biti slučaj. U nastavku će biti prikazan programski kod koji vrši vezivanje podataka iz tabele RADNIK baze podataka PROBA za **DataGridView** kontrolu.

Složeno povezivanje kontrola sa podacima

```
DataGridView dgv = new DataGridView();

MySqlConnection conn = new MySqlConnection();

conn.ConnectionString = "server=localhost;user id=ucenik;password=test;database=proba";

String strSQL = "Select * from RADNIK";

MySqlCommand comm = new MySqlCommand(strSQL, conn);

MySqlDataAdapter adapter = new MySqlDataAdapter(comm);
```

```
DataSet ds = new DataSet();

try
{

   conn.Open();

   adapter.Fill(ds, "Radnik");

   conn.Close();

   dgv.DataSource = ds.Tables["Radnik"];

   dgv.Width = this.Width;

   this.Controls.Add(dgv);

}

catch (Exception exc)
{

   //obrada izuzetka

}
```

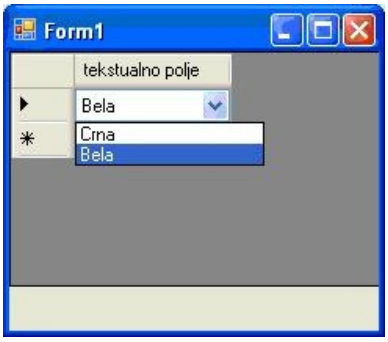
Kako je ranije rečeno, **DataGridView** kontrola kao izvor podataka koristi kolekcije podataka. Primer ovakvog izvora podataka je **DataSet** objekat. Iako **DataSet** objekat može sadržati više tabela, od kojih svaka predstavlja kolekciju podataka, **DataGridView** kontrola je projektovana tako da u jednom trenutku može prikazivati samo jednu od kolekcija koje sadrže podatke. Na primer, svaka od tabela u **DataSet** objektu predstavlja kolekciju podataka. Ukoliko **DataSet** poseduje više tabela, **DataGridView** ima mogućnost jednovremenog prikaza samo jedne od tabela posmatranog **DataSet** objekta.

DataGridView dozvoljava projektantima aplikacija određivanje nivoa interakcije korisnika sa podacima tj projektanti aplikacija određuju da li će korisnici imati mogućnost ažuriranja, brisanja i dodavanja podataka. Podrazumevane karakteristike **DataGridView** kontrole dozvoljavaju korisnicima da ažuriraju podatke u pojedinačnim ćelijama, da selektuju redove i brišu ih korišćenjem Delete tastera na tastaturi i da dodaju novi red korišćenjem poslednjeg prikazanog praznog reda kontrole. Da bi se korisnicima zabranilo dodavanje novog reda i brisanje selektovanog reda potrebno je podesiti atribute **AllowUserToAddRows** i **AllowUserToDeleteRows** na vrednost **false**, respektivno. Takođe, moguće je zabraniti dodavanje novog reda, brisanje selektovanog reda i ažuriranje podataka postavljanjem vrednost **ReadOnly** atributa kontrole na vrednost **true**. **DataGridView** kontroli je moguće dinamički dodavati kolone i redove. Da bi se dodala nova kolona u **DataGridView** kontrolu neophodno je definisati templejt ćelija kolone. Templejt ćelija kolone određuje tip koji će biti korišćen za ćeliju nove kolone svaki put kada

se u kontrolu dodaje novi red. Svi tipovi ćelija izvedeni su iz **DataGridViewCell** klase i mogu biti tekstualna polja, padajuće liste, linkovi, dugmići i slike. Na osnovu tipova ćelija moguće je dinamički kreirati redove i dodavati ih u **DataGridView** kontrolu. U nastavku će biti dat primer programskog koda koji vrši kreiranje **DataGridView** kontrole i u kontrolu dodaje jednu kolonu a zatim dodaje jedan red.

Korišćenje DataGridView kontrole

```
DataGridView dgv = new DataGridView();

DataGridViewTextBoxColumn newCol = new DataGridViewTextBoxColumn();

newCol.HeaderText = "tekstualno polje";

dgv.Columns.Add(newCol);

DataGridViewRow newRow = new DataGridViewRow();

DataGridViewComboBoxCell comboCell = new DataGridViewComboBoxCell();

comboCell.Items.Add("Crna");

comboCell.Items.Add("Bela");

comboCell.Value = "Bela";

newRow.Cells.Add(comboCell);

dgv.Rows.Add(newRow);
```



Prilikom izmene sadržaja ćelija **DataGridView** kontrole od velikog je značaja da li je kontrola povezana sa podacima. Ukoliko je povezivanje izvršeno, svi izmenjeni podaci će automatski biti zapamćeni u izvoru podataka (u bazi podataka) iz kojeg su pribavljeni. Ukoliko se u ćeliji nalazi link ili dugme, korisnici neće biti u mogućnosti da menjaju sadržaj ćelije jer ove kontrole ne dozvoljavaju promene njihovog sadržaja korisnicima. Ukoliko se u ćeliji nalazi padajuća lista, pod izmenom sadržaja smatra se selektovanje neke druge stavke u odnosu na trenutno selektovanu ili prepisivanje trenutno prikazanog teksta u stavki padajuće liste (da bi prepisivanje sadržaja trenutno selektovane stavke bilo moguće, **DisplayStyle** atribut padajuće liste mora imati vrednost "**ComboBox**").

1.3. Pitanja

Pokušajte da odgovorite na sledeća pitanja. Nakon toga pogledajte ponovo materijal u ovoj lekciji. Za svaki tačan odgovor dodelite sebi 2 poena, za delimično tačan 1, a za netačan 0. Pogledajte ponovo one delove lekcije za koje ste imali 0 poena.

1. Šta je povezivanje kontrola sa podacima u .NET platformi?
2. Šta može biti pokretač mehanizma za povezivanje kontrola sa podacima?
3. Koji su primarni oblici povezivanja Windows Forms kontrola .NET platforme sa podacima?
4. Na koji način funkcioniše jednostavno povezivanje Windows Forms kontrole sa izvorom podataka?
5. Ukoliko se vrši povezivanje podataka sa ComboBox kontrolom, koji atribut ComboBox kontrole određuje koji će atribut izvora podataka popuniti podacima stavke kontrole?
6. Šta omogućava složeno povezivanje Windows Forms kontrola sa podacima?
7. Na koji način DataGridView kontrola prikazuje podatke koji su vezani za nju?
8. Koja klase se najčešće koristi kako bi se instancirali objekti koji predstavljaju izvore podataka korišćene pri složenom povezivanju?
9. Koja je razlika između GridView i ListBox kontrole u pogledu mogućnosti prikaza podataka sa kojima su povezane?

1.4. Zadatak

ADO.NET

1. Korišćenjem ADO.NET-a izvršiti povezivanje Windows Forms ComboBox kontrole sa podacima o radnicima. Svaka stavka ComboBox kontrole treba da prikazuje ime i prezime radnika, dok je ValueMember atribut ComboBox kontrole povezan sa matičnim brojevima radnika. Podaci o radnicima čuvaju se u tabeli RADNIK baze podataka PROBA.

LITERATURA:

Baze podataka, elektronski fakultet, Niš, 2013
https://edoc.pub/baze-podatakapdf-pdf-free.html