

NOP – KLASA – POVEZIVANJE S BAZOM

C# - OSNOVE

- U programiranju postoje par osnovnih petlji/naredbi/upita koje se često koriste.
 - FOR petlja koja vrti varijablu N puta.

```
for (int i = 0; i < N; i++)  
{  
    // kod koji će se izvršavati.  
}
```

- DO WHILE petlja koja se izvršava dok je neki uvjet ispunjen.

```
do  
{  
    // kod  
} while (N>15);
```

- IF/else IF uvjet koji provjerava neki uvjet

```
if (N == 15)  
{  
    // kod  
}  
else  
{  
    // kod  
}
```

- SWITCH – case.

```
switch (N)
{
    case 1:
        // neka metoda ili kod pri tom slučaju.
        break;

    case 2:
        // // neka metoda ili kod pri tom slučaju.
        break;

    case 3:
        //...
        break;

    default:
        break;
}
```

- FOREACH petlja koja je trenutno najkorištenija od svih.
Ta petlja će ići kroz svaki element ili indeks neke liste/niza.

```
foreach (var broj in listaBrojeva)
{
    // kod
}
```

- Lambda upit – sa lambda upitima možemo pronaći neke podatke ili saznati postoje li uopće, u kodu umjesto “Find” naredbe možemo pisati FindAll, Average, Select ...

```
List<string> listaImena = new List<string>
{"Ivan", "Marko", "Josip", "Marin" };

string trazenoIme = "Marko";

var rezultat = listaImena.Find(ime => ime == trazenoIme);

Console.WriteLine(rezultat);
```

C# - KLASA

- Klasa je “predložak” koda koji služi za kreiranje objekata te klase.
- Osnovna ideja klase jest da se kod učini jednostavnijim i bržim/optimiziranijim.
- Objekt klase jest jedna instanca te klase, npr. imamo klasu :

```
internal class Osoba
{
    0 references
    public int ID { get; set; }

    0 references
    public string Ime { get; set; }

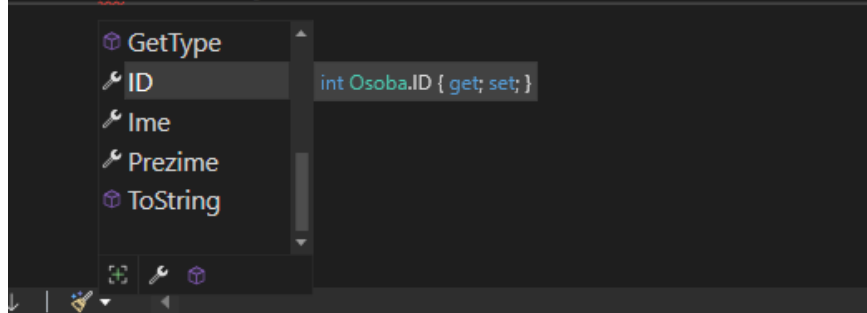
    0 references
    public string Prezime { get; set; }
}
```

U klasi Osoba možemo kreirati objekt jedne osobe koja će imati svoja svojstva : ID, ime i prezime.

```
string ime = "Teofil";
string prezime = "Djovani";
int ID = 1000;

Osoba o = new Osoba(ID,ime,prezime);

o.ID = ID;
```



- Također je u klasama bitan konstruktor klase, on inicijalizira objekte.

```
1 reference
public Osoba(int id, string ime, string prezime)
{
    ID = id;
    Ime = ime;
    Prezime = prezime;
}
```

- Također postoji razlika u **public** članovima i **static** članovima.

Recimo da u klasi osoba imamo dvije metode, jednu public, i jednu static metodu.

Osoba.cs :

```
0 references
public static void MetodaStaticna()
{
}

1 reference
public void PublicMetoda()
{
}
```

Program.cs :

```
o.PublicMetoda();

Osoba.MetodaStaticna();
```

Public metodu ćemo pozvati preko jednog objekta klase Osoba, a statičnu metodu ćemo pozvati putem naziva klase Osoba.

- Overridanje

- Recimo da želimo ispisati sve osobe iz naše klase putem FOREACH petlje

```
foreach (var osoba in Osoba.popisOsoba)
{
    Console.WriteLine(osoba);
}
```

- Kada bi ovaj kod pokrenuli, ispisao bi se doslovni objekt, ne bi se ispisao ID, ime i prezime od svake osobe.
- Zato služi overridanje, kada korisnik želi ispisati objekt, da mu se ispiše zapravo ono što traži.

```
0 references
public override string ToString()
{
    return ID + " " + Ime + " " + Prezime;
}
```

Override metoda vraća objekt u string obliku.

- Nasljeđivanje je postupak pojednostavljivanja koda pri kojem neka klasa može doslovce naslijediti neka svojstva od druge klase, npr.

```
0 references
internal class Auto:Proizvod
{
    0 references
    public int ID { get; set; }

    0 references
    public int BrojKonja { get; set; }

    0 references
    public int Model { get; set; }
}
```

U klasi Auto imamo ID, broj konja i model auta, no taj auto će također naslijediti i svojstva od klase proizvod (iznad piše Auto:Proizvod) :

```
1 reference
internal class Proizvod
{
    0 references
    public int Cijena { get; set; }

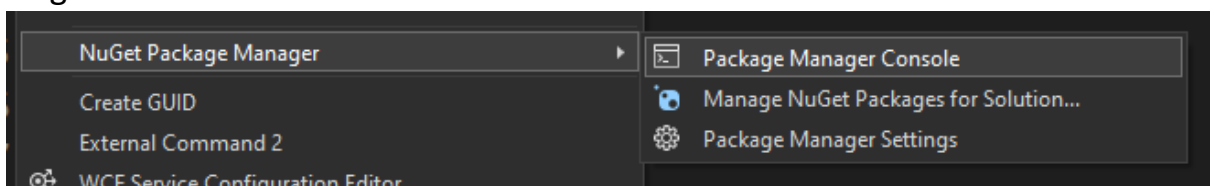
    0 references
    public int Kolicina { get; set; }

    0 references
    public int Masa { get; set; }
}
```

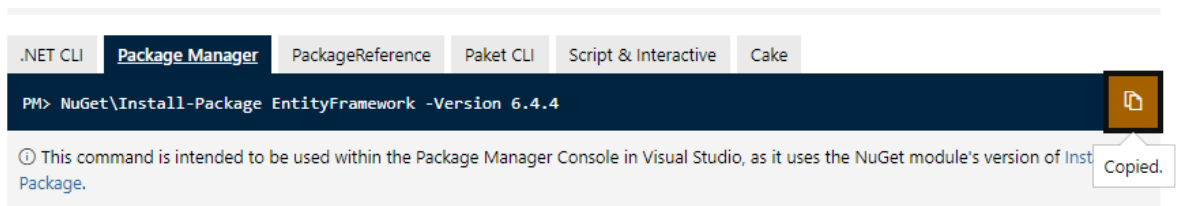
Auto sada zapravo ima i svoju cijenu, količinu i masu.

C# - POVEZIVANJE S BAZOM PODATAKA

- U C#-u se možemo povezati sa SQL bazom podataka putem Entity frameworka, s time smo postigli činjenicu da možemo zapravo urediti podatke i kada se vratimo nazad u program ponovo možemo vidjeti te podatke.
- Za kreiranje baze podataka koristimo SSMS (SQL Server Management Studio)
- Za instalaciju Entity Frameworka u naš projekt moramo se koristiti Nuget-ovom konzolom

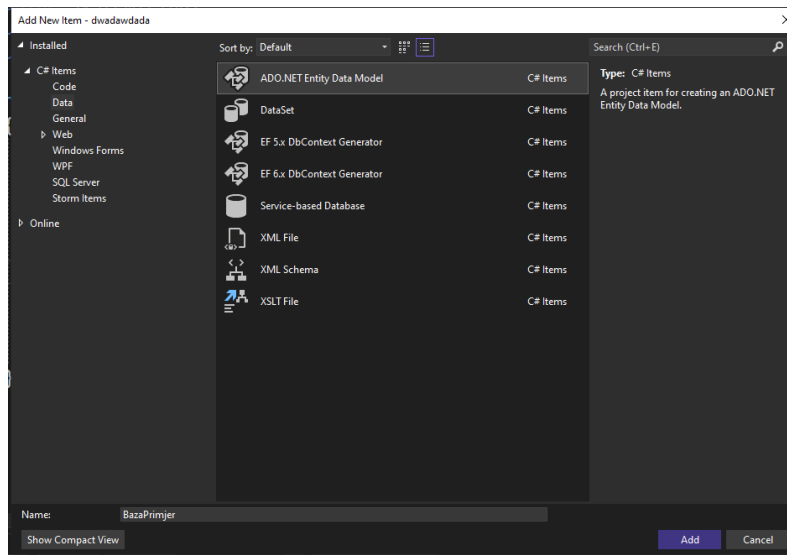


- Te u konzolu ubacimo :

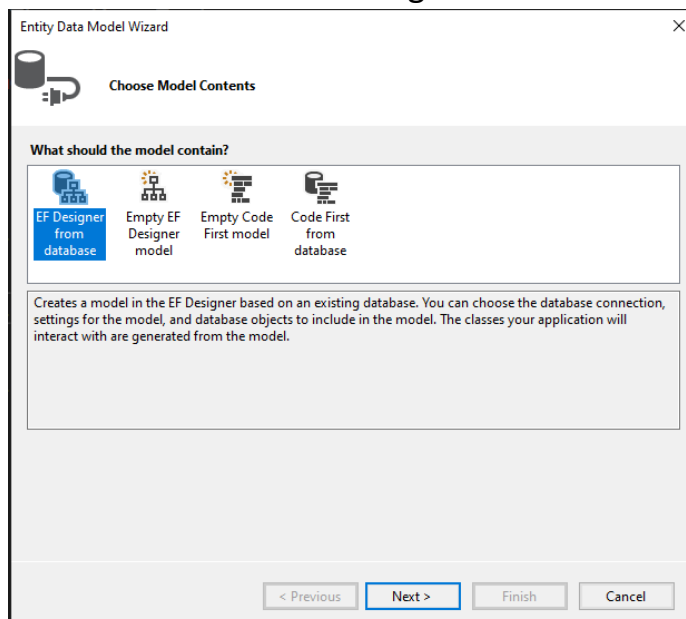


(LINK: <https://www.nuget.org/packages/EntityFramework>)

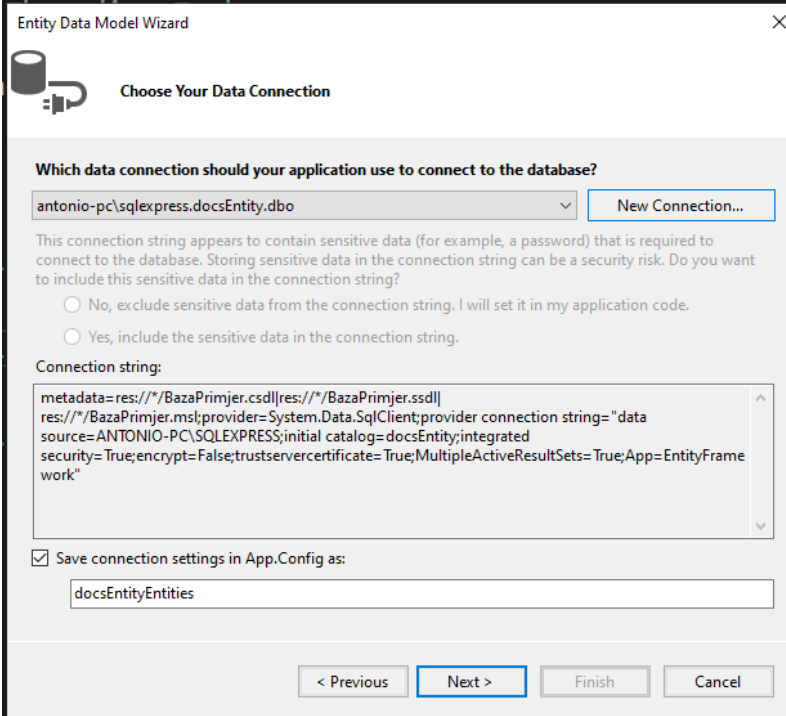
- Da spojimo bazu podataka moramo napraviti novi ITEM u našem Solution Exploreru :



Zatim odaberemo “EF designer from database” :



→ Stisnemo new Connection:



Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

antonio-pc\sqlexpress.docsEntity.dbo New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Connection string:

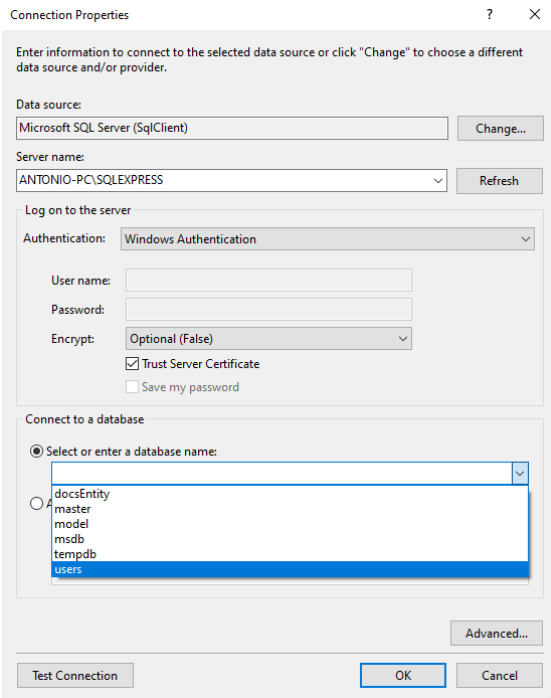
```
metadata=res://*/BazaPrimjer.csdl|res://*/BazaPrimjer.ssdl|
res://*/BazaPrimjer.msl;provider=System.Data.SqlClient;provider connection string="data
source=ANTONIO-PC\SQLSERVER;initial catalog=docsEntity;integrated
security=True;encrypt=False;trustservercertificate=True;MultipleActiveResultSets=True;App=EntityFrame
work"
```

☒ Save connection settings in App.Config as:

docsEntityEntities

< Previous **Next >** Finish Cancel

Zatim odaberemo naš lokalni server:



Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient) Change...

Server name: ANTONIO-PC\SQLSERVER Refresh

Log on to the server

Authentication: Windows Authentication

User name:

Password:

Encrypt: Optional (False)

☒ Trust Server Certificate

☐ Save my password

Connect to a database

☒ Select or enter a database name:

☐ docsEntity

☐ master

☐ model

☐ msdb

☐ tempdb

☐ users

Advanced...

Test Connection OK Cancel

Nakon spajanja na bazu možemo napraviti context baze, s kojim doslovno možemo raditi nad podacima u nekoj tablici.

```
0 references
internal class Program
{
    static usersEntities contextBaza = new usersEntities();

    0 references
    static void Main(string[] args)
    {
```

Sa contextom možemo urediti podatke, ispisati podatke iz baze, brisati podatke, ažurirati podatke i slično...

U "var" tipu varijable ćemo spremiti sve korisnike iz baze te to pretvoriti u listu i putem FOREACH petlje ih ispisati.

```
var popisLjudi = contextBaza.Korisnici.ToList();

foreach (var osoba in popisLjudi)
{
    Console.WriteLine(osoba.id + " " + osoba.username);
}
```

Primjer dodavanja neke osobe u bazu:

```
Korisnici k = new Korisnici();

k.username = Console.ReadLine();

k.password = Console.ReadLine();

contextBaza.Korisnici.Add(k);

contextBaza.SaveChanges();
```

Obavezno je spremiti promjene u bazi sa zadnjom naredbom.