

Zadatak sa predavanja za vezbanje

Primer kolokvijuma, SI Vežbe 2020

Kreirati privatni *github* projekat pod nazivom **KOL-SI-<broj_indeksa>**. Na novom projektu, napraviti *development* granu (*remote*) a zatim klonirati projekat lokalno. Napraviti novu lokalnu granu (iz *development*) pod nazivom **<broj_indeksa>** i na njoj raditi zadatak. Na kraju rada (!), *push*-ovati granu na *remote* repozitorijum. (2)

Koristeći *Visual Studio* alat napraviti nov *Windows Forms* projekat (.NET Framework) pod nazivom *PresentationLayer* i dodati odgovarajuće projekte koje čine troslojnu arhitekturu (*DataLayer* i *BusinessLayer*). (2)

Kreirati novu lokalnu bazu podataka (na instanci po želji) pod nazivom *FacultyDB* i u njoj napraviti tabelu *Students* sa kolonama: *Id* (PK, *IDENTITY*), *Name* (*nvarchar*(100), NOT NULL), *IndexNumber* (*nvarchar*(20), NOT NULL), *AverageMark* (decimal(16,2), NULL). (3)

Na odgovarajućem sloju kreirati odgovarajući *repository* i u njemu metodu koja vraća listu svih studenata iz baze podataka. U istom repozitorijumu dodati i metodu koja omogućava unos jednog studenta u bazu podataka. Biće neophodno kreirati i model-klasu koja se "mapira" na prethodno kreiranu tabelu u bazi podataka. (4)

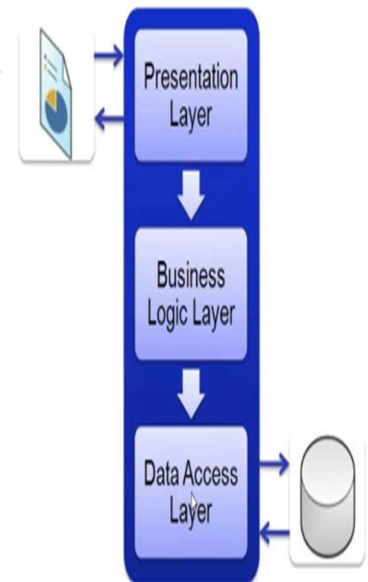
Na sloju poslovne logike "povezati" se na sloj koji komunicira sa bazom podataka i napraviti tri metode: jednu koja vraća sve studente iz baze podataka, drugu sa argumentom tipa decimalne vrednosti koja vraća studente iz baze podataka koji imaju prosečnu ocenu veću od prosleđene i treću koja omogućava unos studenta u bazu. (3)

Na prezentacionom sloju prikazati sve studente iz baze u listi i omogućiti unos jednog studenta. (3)

Napraviti još i *ASP.NET Web Forms* projekat na kojem treba omogućiti samo prikaz liste svih studenata iz baze podataka. (3)

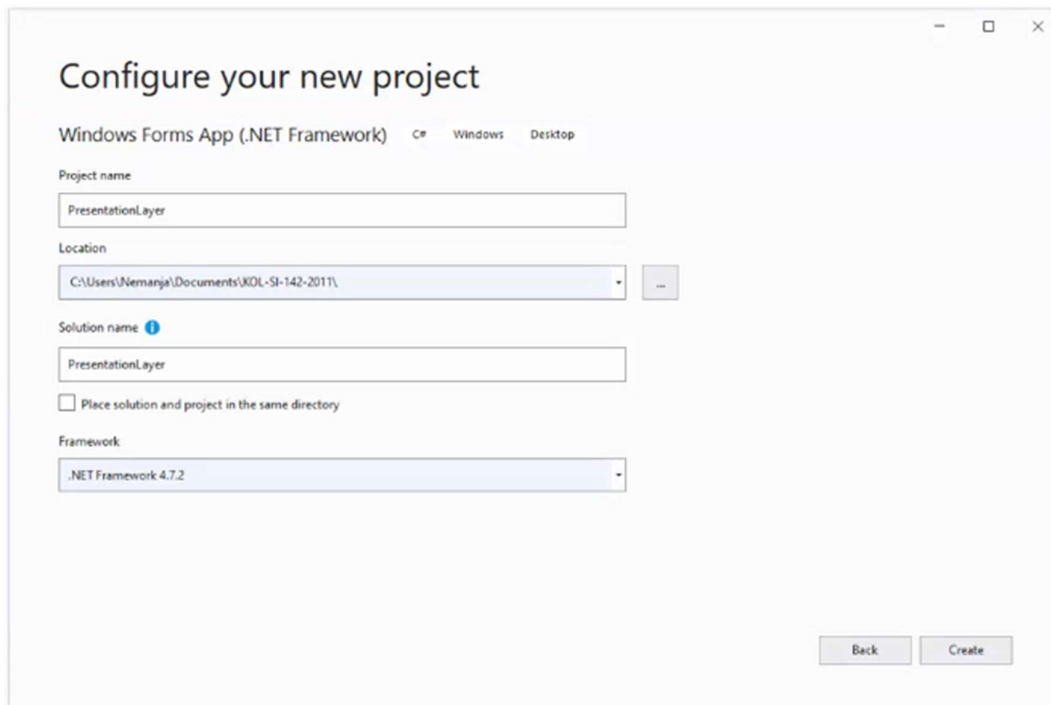
Napomene:

- Koristiti auto svojstva u model klasama
- Klase -> jednina, tabele u bazi -> množina
- Poželjno: redovno *commit*-ovati izmene



Prvo kreiramo folder na desktop, unutar tog foldera mozemo a I ne moramo da kreiramo poseban folder za bazu.

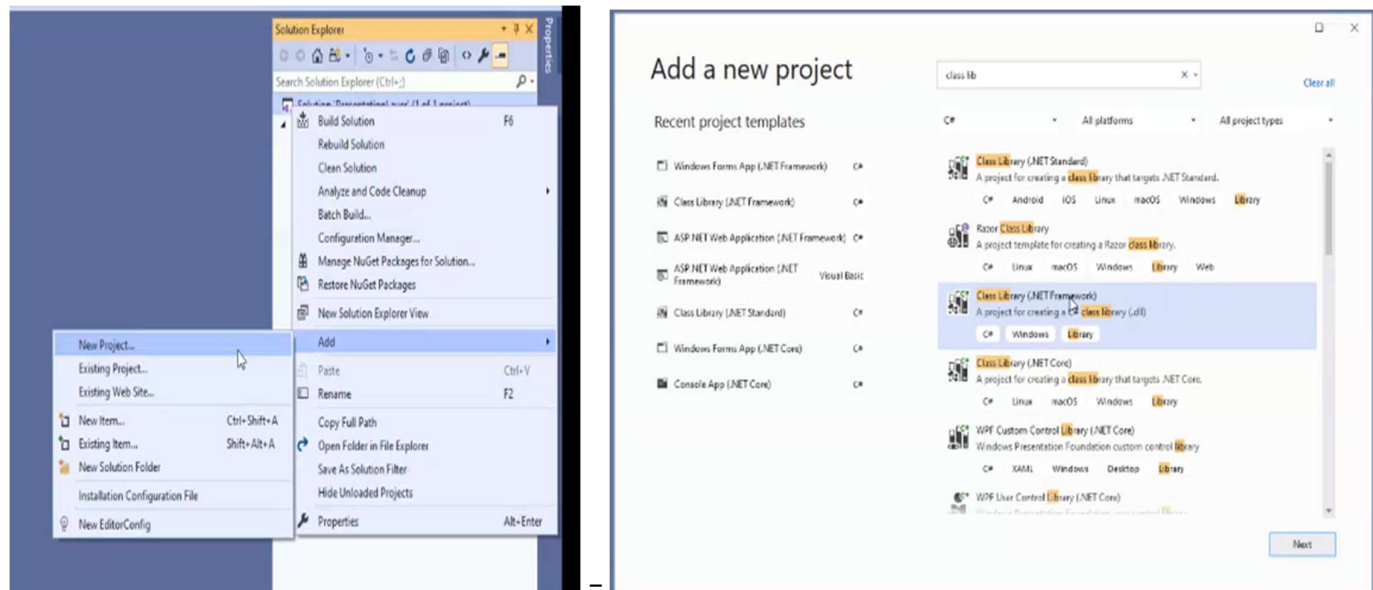
Upalimo Visual Studio I u okviru njega kreiramo projekat u vidu **Windows forme .NetFramework** koja će se zvati **PresentationLayer**, sačuvamo unutar kreiranog folder ana desktop.



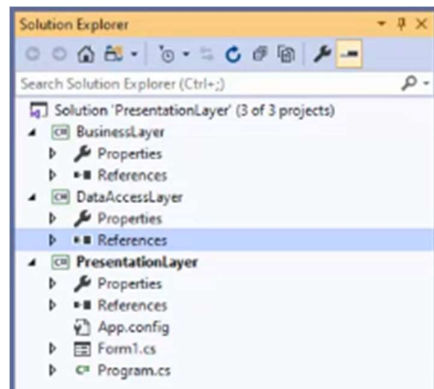
Nakon kreiranja windows forme odmah kreiramo I ostala 2 sloja. *DataAccessLayer* I *BusinessLayer*.

Sloj kreiramo tamo sto idemo desni klik na Solution Explorer sa desne strane odaberemo Add pa New Project nakon čega biramo Class Labrary .net framework I nazovemo po jednom od preostala dva lejera.

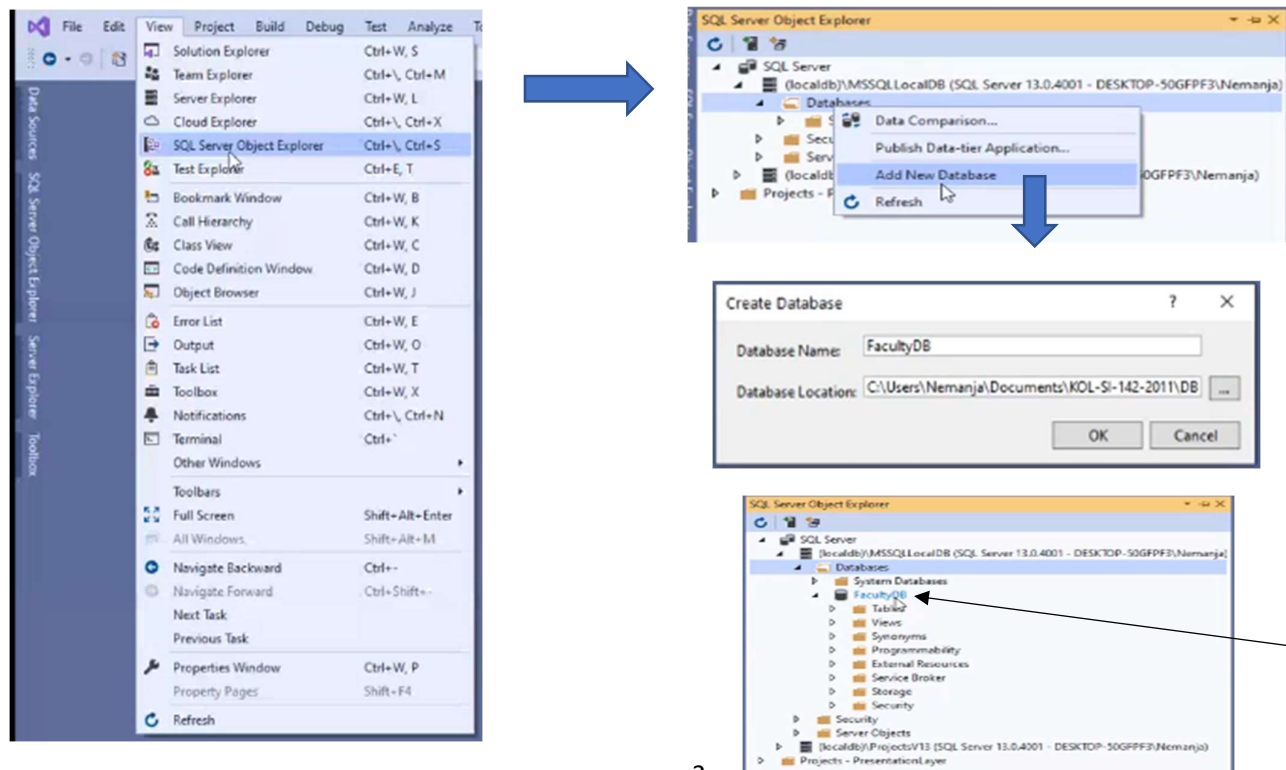
Unutar kreiranog lejera brišemo difoltnu class1 jer nam ne treba.



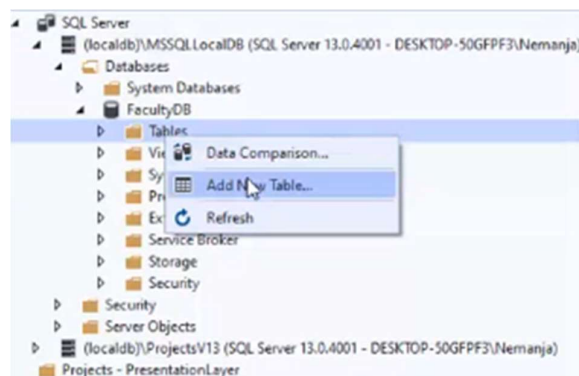
Kao rezultat imamo sledeci prikaz u Solution Exploreru.



Nakon toga kreiramo bazu podataka. Baza se kreira na sledeci nacin:

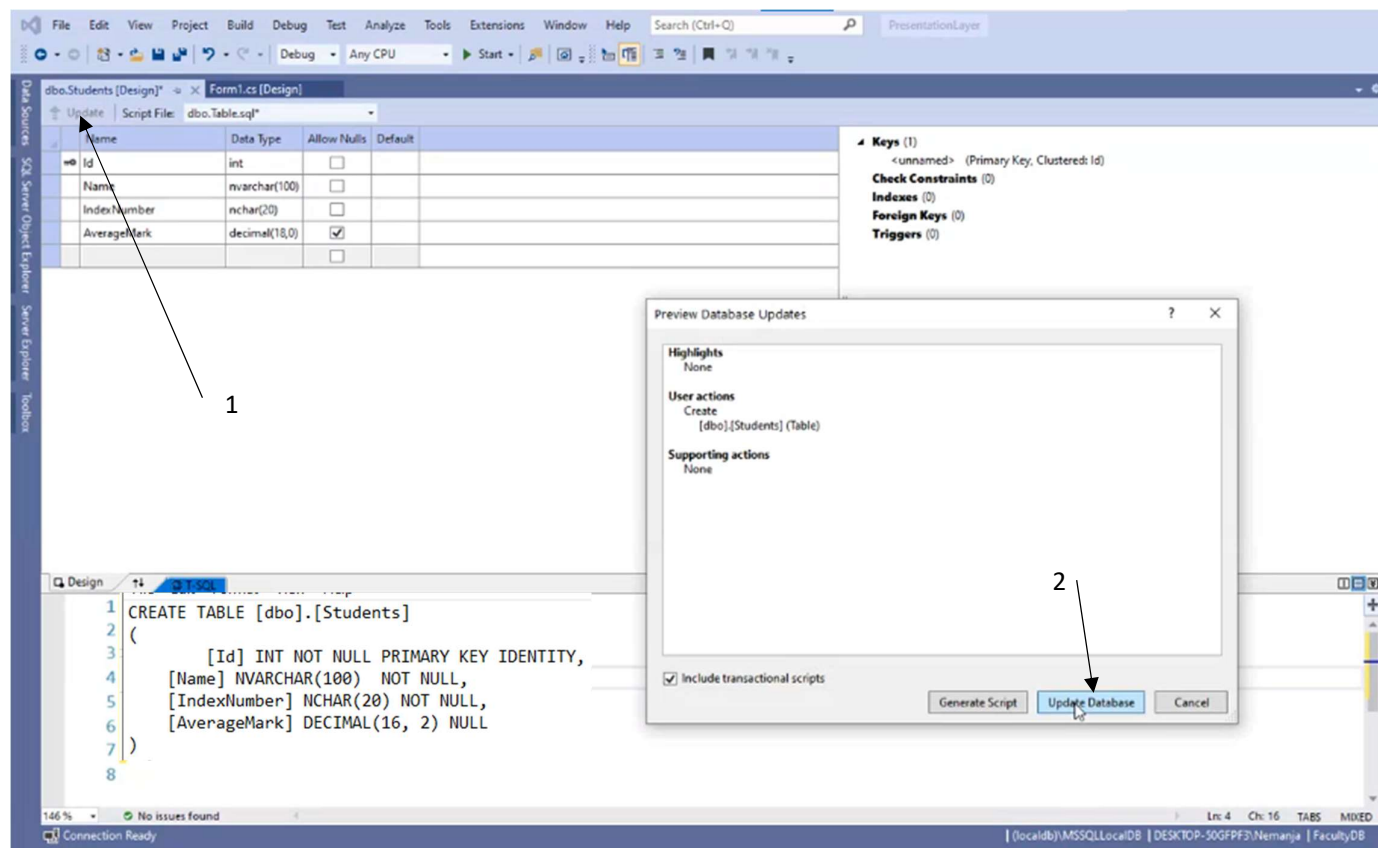


Nakon kreirane baze potrebno je da kreiramo tabelu, ona se kreira na sledeci nacin:

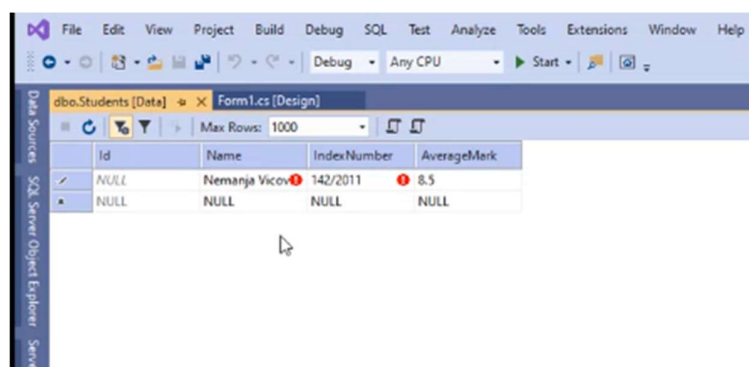
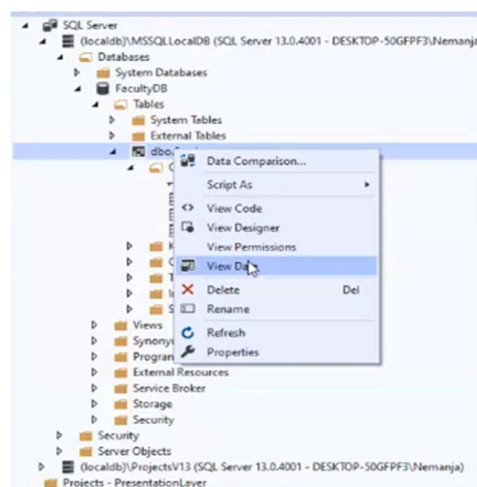


```
CREATE TABLE [dbo].[Students]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [Name] NVARCHAR(100) NOT NULL,
    [IndexNumber] NCHAR(20) NOT NULL,
    [AverageMark] DECIMAL(16, 2) NULL
)
```

Kada kreiramo tabelu potrebno je uraditi Update.



Nakon kreiranja potrebno je popuniti tabelu a to se radi pomocu komande Veiw Data:



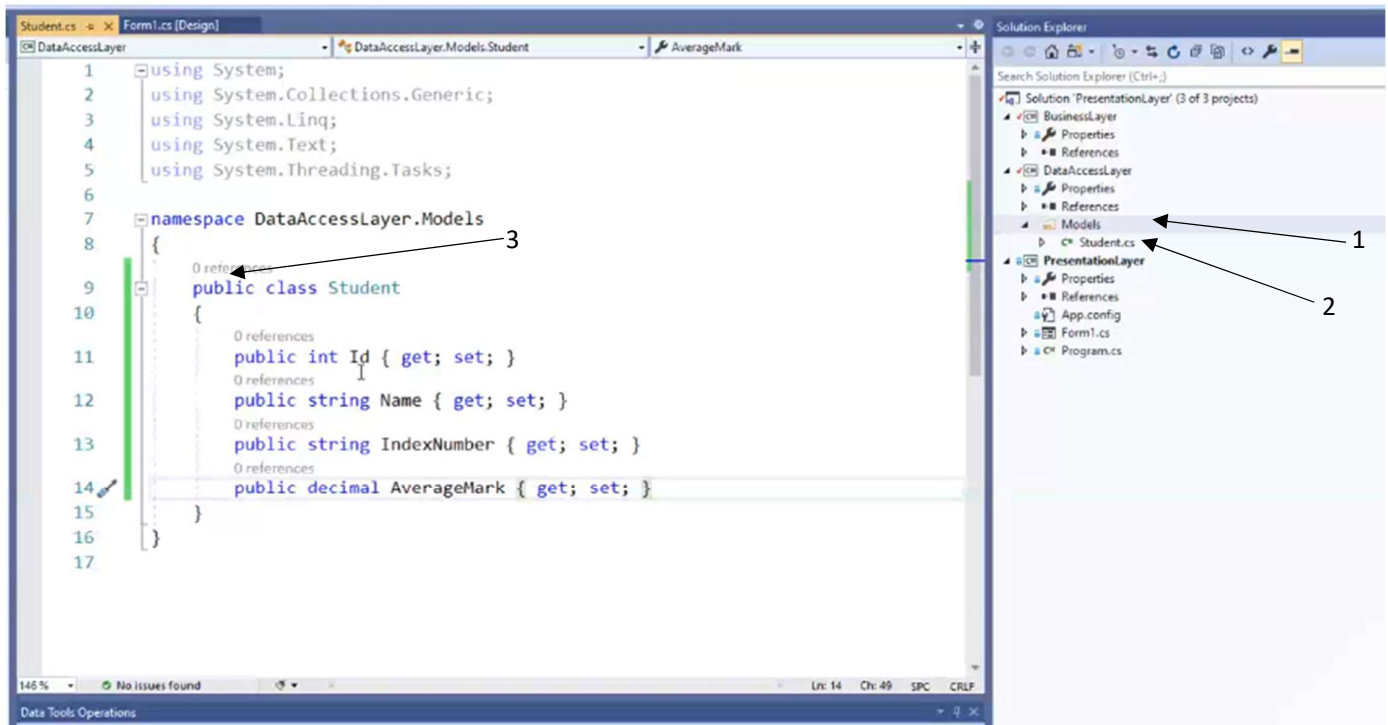
Takodje moguće je postavljati i kver upite ukoliko je to potrebno.

```

1 SELECT * FROM Students
2
3 INSERT INTO Students VALUES('Mladen Janjic', '500/2015', 9.5)

```

Nakon toga kreiramo jedan folder (Models) u okviru DataAccess lejera sa klasom student.

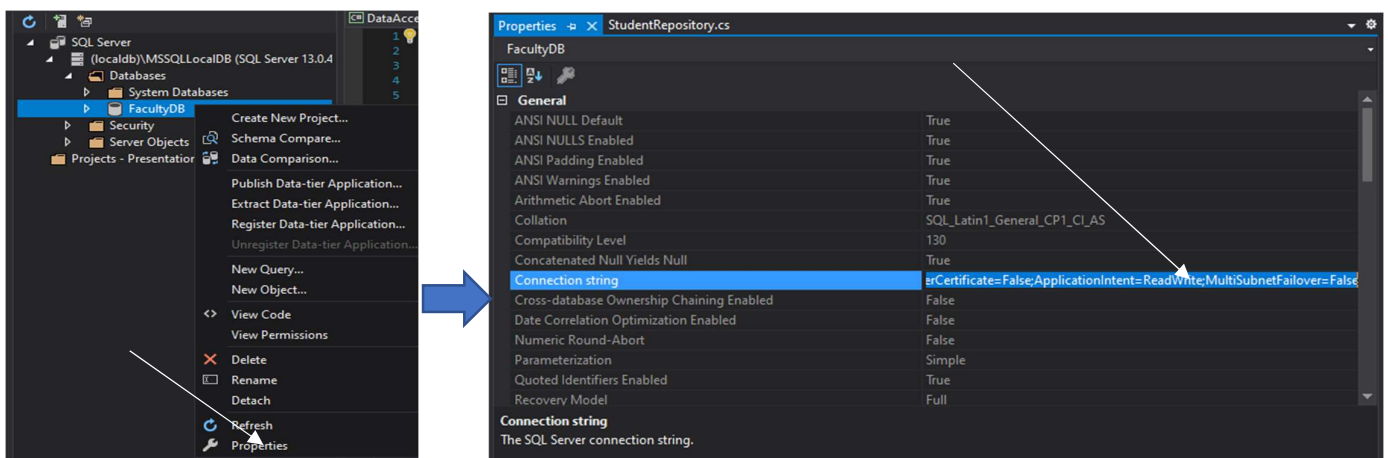


- 1-Kreiran folder Models
- 2-Kreirana klasa student
- 3-Postavljanje public klase da bi mogli da je koristimo i u nekim drugim slucajevima

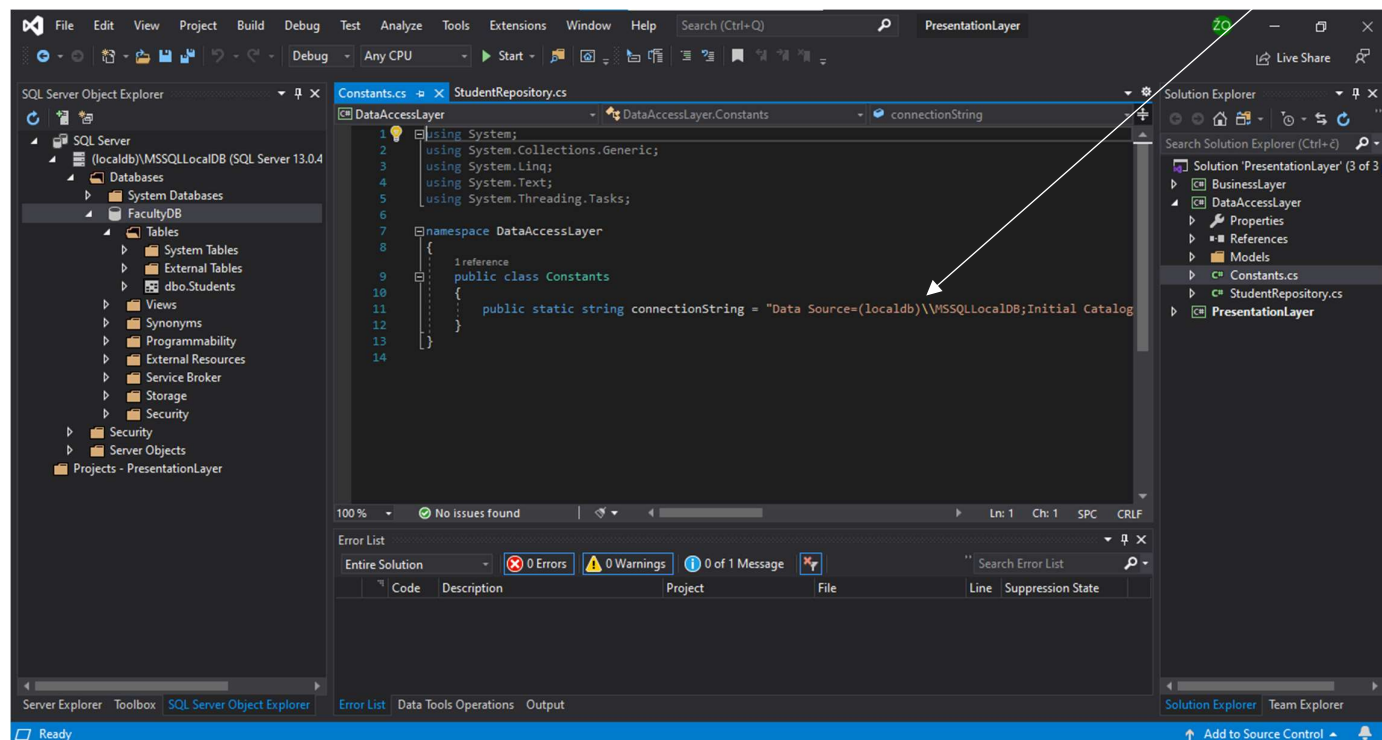
Kreiramo klasu StudentRepository unutar DataAccess lejera.

Nakon toga kreiramo još jednu klasu Constants u koju smeštamo connectionString.

ConnectionString nalazimo tako sto kliknemo desni klik na bazu pa Propertis i kopiramo sledeci kod:



Kod kopiran iz connectionString-a kopiramo u klasu Constants. Vodeći računa da budu 2 bek sleša (\\) ovde:

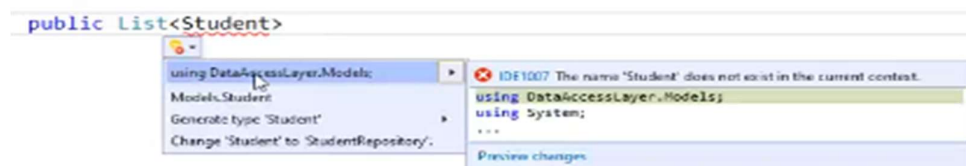


Nakon toga prelazimo u klasu StudentRepository:

Prvo kreiramo metodu koja vraća listu iz baze.

```
0 references
public List<Student> GetAllStudents()
{
    // ...
}
```

Ako javlja gresku pridjemo misom na Student i generisemo using biblioteku u kojoj spadaju liste.



U okviru GetAllStudents() kreiramo sledeću listu:

```
0 references
public List<Student> GetAllStudents()
{
    List<Student> results = new List<Student>();

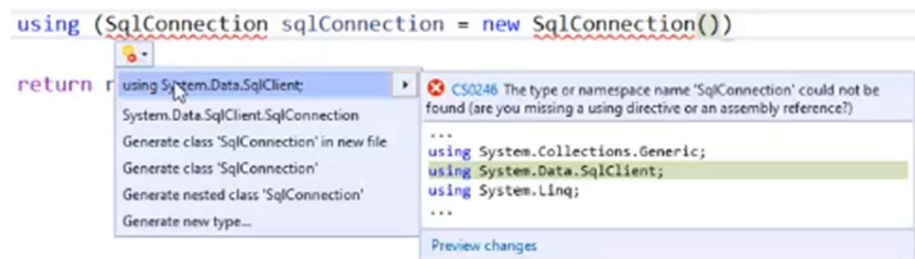
    // ...

    return results;
}
```

Dalje kreiramo usning u okviru kog kućamo sledeće kodove ispod koji slede:

```
using (SqlConnection sqlConnection = new SqlConnection(Constants.connectionString))
{
    // ...
}
```

Ako se pojavi greška samo pridjemo mišom na SqlConnection i generišemo biblioteku koja prepoznaje SqlConn...



Dalje kreiramo sql komandu:

```
SqlCommand sqlCommand = new SqlCommand();
sqlCommand.Connection = sqlConnection;
```

Nakon kreiranja sqlCommand-e kreiramo komand tekst komandu sa sql upitom:

```
sqlCommand.CommandText = "SELECT * FROM Students";
```

Nakon toga otvaramo konekciju sledećom komandom:

```
sqlConnection.Open();
```

Dalje, rezultate iz komande smeštamo u komandu ExecuteReader :

```
SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();
```

Podatke cemo procitati pomocu while upita I metode Read u okvirnju njega:

```
while (sqlDataReader.Read())
{
    // ...
}
```

Posto prethodnim kodom citamo red po red iz tabele Students pisemo sledeci kod;

```
while (sqlDataReader.Read())
{
    Student s = new Student();
    s.Id = sqlDataReader.GetInt32(0);
    s.Name = sqlDataReader.GetString(1);
    s.IndexNumber = sqlDataReader.GetString(2);
    s.AverageMark = sqlDataReader.GetDecimal(3);
    results.Add(s);
}
```

```

public List<Student> GetAllStudents()
{
    List<Student> results = new List<Student>();

    using (SqlConnection sqlConnection = new SqlConnection(Constants.connectionString))
    {
        SqlCommand sqlCommand = new SqlCommand();
        sqlCommand.Connection = sqlConnection;

        sqlCommand.CommandText = "SELECT * FROM Students";

        sqlConnection.Open();

        SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

        while (sqlDataReader.Read())
        {
            Student s = new Student();
            s.Id = sqlDataReader.GetInt32(0);
            s.Name = sqlDataReader.GetString(1);
            s.IndexNumber = sqlDataReader.GetString(2);
            s.AverageMark = sqlDataReader.GetDecimal(3);
            results.Add(s);
        }
    }

    return results;
}

```

Kreiramo novu javnu metodu **InsertStudent**, ispod postojeće **GetAllStudent** i dalje unutar klase **StudentRepository**.

Metoda za unos studenata.

```

public int InsertStudent (Student s)
{
    using (SqlConnection sqlConnection = new SqlConnection(Constants.connectionString))
    {
        SqlCommand sqlCommand = new SqlCommand();
        sqlCommand.Connection = sqlConnection;

        sqlConnection.Open();

        sqlCommand.CommandText =
            string.Format("INSERT INTO Students VALUES ('{0}', '{1}', {2})",
                s.Name, s.IndexNumber, s.AverageMark);

        return sqlCommand.ExecuteNonQuery();
    }
}

```

Ovim završavamo sa **DataAccess** Lejerom.

Kreiramo novu klasu unutar Biznis lejera koja ce se zvati **StudentBusiness** i bice javna.

```
namespace BusinessLayer
{
    // References
    public class StudentBusiness
    {
    }
}
```

Prvo sto moramo da uradimo je povezivanje izmedju Biznis i DataAkses sloja.

To se radi na sledeci nacin:

Prvo pravimo jedno privatno readonly polje: read polje noze biti i obrisano ako pravi gresku

```
public class StudentBusiness
{
    private readonly StudentRepository studentRepository;
}
```

U slucaju da ne prepozna i javlja gresku otklanjamo je na sledeci nacin: misem dodamo referencu.

Ovim pravimo referencu od Biznis lejera ka Data Akses lejeru.

U konstruktoru ove klase inicijalizujemo ovaj repozitorijum.

```
public StudentBusiness ()
{
    this.studentRepository = new StudentRepository();
}
```

Kreiramo metodu koja vraca listu svih studenata.

```
public List<Student> GetAllStudents()
{
    return this.studentRepository.GetAllStudents();
}
```

U slucaju da se javljaju greske, otkloniti ih prilazom misa na <Student> i odabrati :

Dalje kreiramo metodu koja omogućava unos studenata u bazu:

```
public bool InsertStudent (Student s)
{
    if (this.studentRepository.InsertStudent(s) > 0)
    {
        return true;
    }
    return false;
}
```

Dalje, kreirati decimalnu metodu koja vraća studente iz baze koji imaju veću ocenu od prosledjene.

```
public List<Student> GetStudentsGTAvgMark (decimal averageMark)
{
    return this.studentRepository.GetAllStudents()
        .Where(s => averageMark > s.AverageMark).ToList();
}
```

PRIKAZ DOSADASNJEG KODA U OKVIRU METODE **StudentBusiness**, RADI ORIJENTACIJE I PROVERE ISTOG

```
1 reference
public class StudentBusiness
{
    private readonly StudentRepository studentRepository;

    0 references
    public StudentBusiness ()
    {
        this.studentRepository = new StudentRepository();
    }

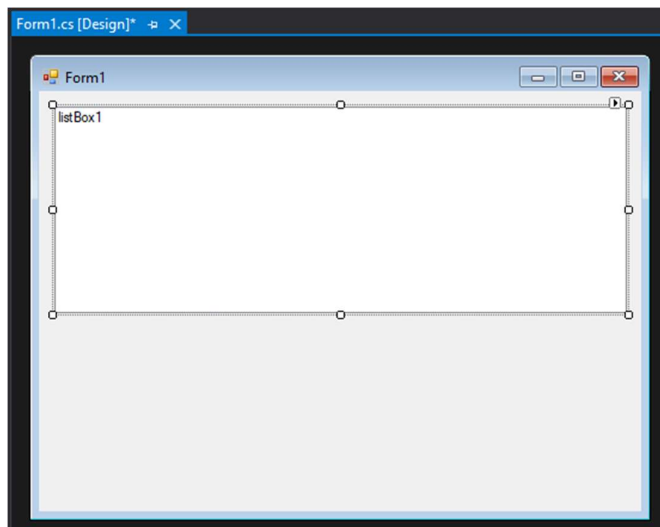
    0 references
    public List<Student> GetAllStudents()
    {
        return this.studentRepository.GetAllStudents();
    }

    0 references
    public bool InsertStudent (Student s)
    {
        if (this.studentRepository.InsertStudent(s) > 0)
        {
            return true;
        }
        return false;
    }

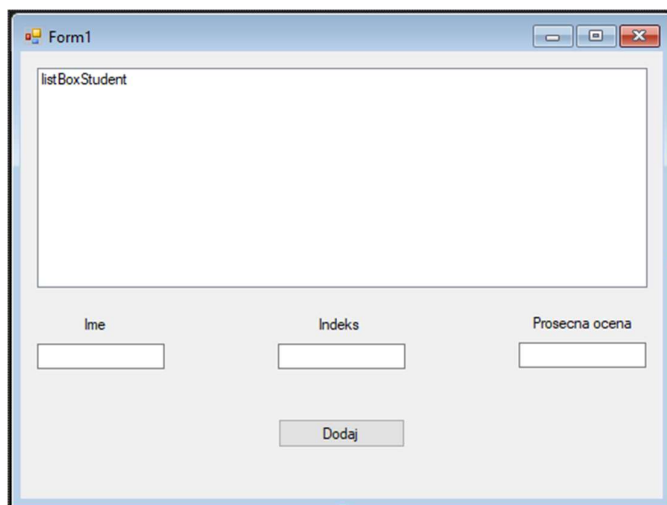
    0 references
    public List<Student> GetStudentsGTAvgMark (decimal averageMark)
    {
        return this.studentRepository.GetAllStudents()
            .Where(s => averageMark > s.AverageMark).ToList();
    }
}
```

Sada se prelazi na prezentacioni sloj tj Formu

Dodamo jedan listBox na formu, promenimo mu ime kakvo nam odgovara desnim klikom u delu propertis



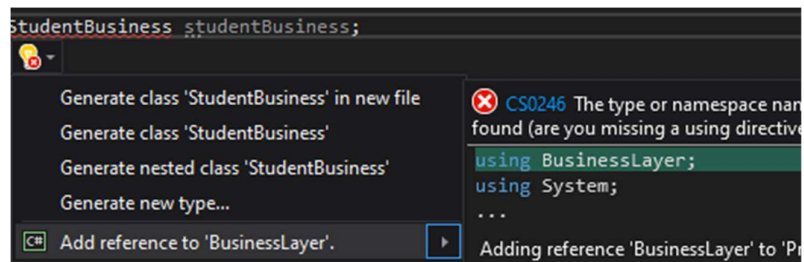
Isti postupak ponoviti i za labele, tekst boksove i dugme tj button.



Nakon toga moramo izvršiti povezivanje sa BusinessLogic lejerom> kliknemo dvoklik na formu i kucamo>

```
private readonly StudentBusiness studentBusiness;
```

Ako javi gresku predjemo misom i dodamo using:



Popunimo konstruktor od Forme:

```
public Form1()
{
    this.studentBusiness = new StudentBusiness();
    InitializeComponent();
}
```

Dalje popunjavamo metodu RefreshData:

```
private void RefreshData ()
{
    List<Student> students = this.studentBusiness.GetAllStudents();
    listBoxStudent.Items.Clear();

    foreach (Student s in students)
    {
        listBoxStudent.Items.Add(s.Id + " " + s.Name + " "
            + s.IndexNumber + " " + s.AverageMark + " ");
    }
}
```

Pozovemo RefreshData metodu unutar Form1_Load metode:

```
private void Form1_Load(object sender, EventArgs e)
{
    RefreshData();
}
```

Kada to završimo idemo dvoklik na button Dodaj i kucamo sledece:

```
private void buttonInsertStudent_Click(object sender, EventArgs e)
{
    Student s = new Student();
    s.Name = textBoxName.Text;
    s.IndexNumber = textBoxIndexNumber.Text;
    s.AverageMark = Convert.ToDecimal(textBoxAverageMark.Text);

    if (this.studentBusiness.InsertStudent(s))
    {
        RefreshData();
    }
    else
    {
        MessageBox.Show("Greska");
    }
}
```

Praznjenje polja nakon unosa: obicno dadavanje praznih navodnika.

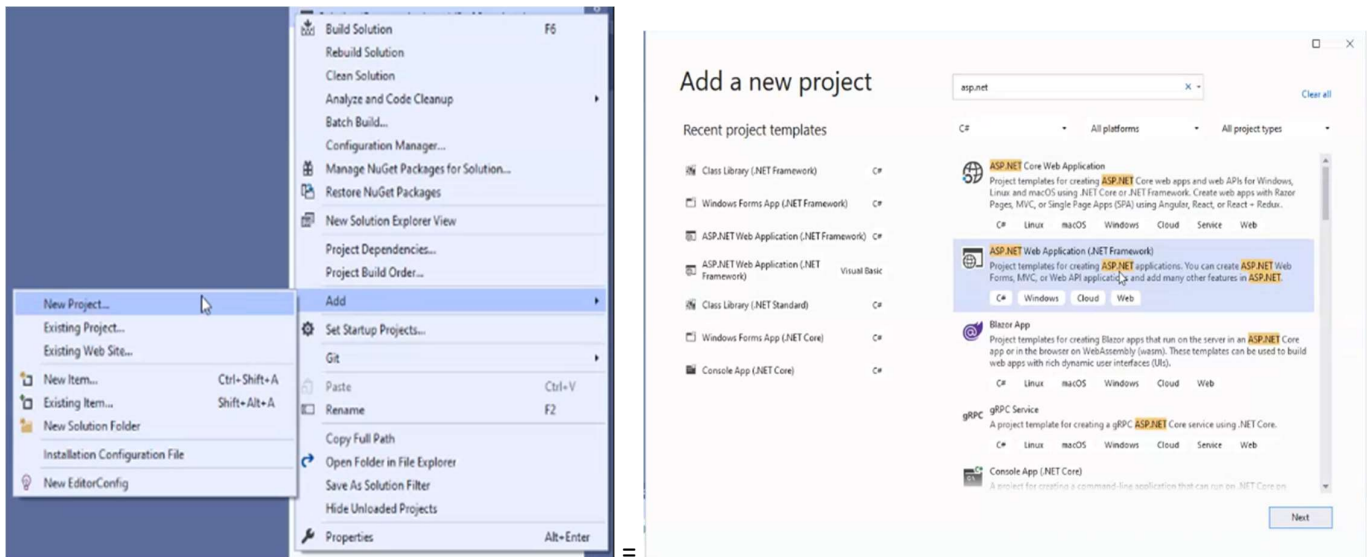
```
if (this.studentBusiness.InsertStudent(s))
{
    RefreshData();
    textBoxName.Text = "";
    textBoxIndexNumber.Text = "";
    textBoxAverageMark.Text = "";
}
```

Vraca prosledjene vrednosti u opsegu minimalne i maksimalne. //Ovde je cisto ako zatreba ova metoda

```
public List<MenuItems>ListMinMaxPrice(decimal min,decimal max)
{
    List<MenuItems> list = new List<MenuItems>();
    foreach(MenuItems menu in List())
    {
        if (menu.Price >= min && min <= max)
            list.Add(menu);
    }
    return list;
}
```

WEB FORME:

Web formu kreiramo tako idemo desni klik na solution explorer, pa new projet l biamo asp.net framework

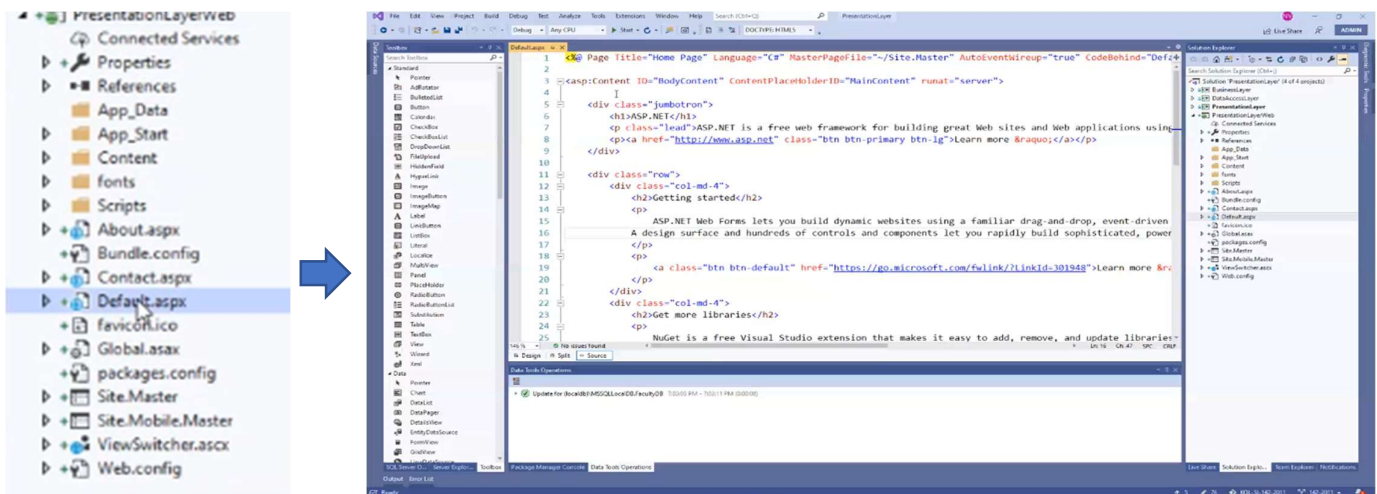


Ime koje mu dodeljujemo moze biti na primer: PresentationLayerWeb l biamo Web Forms

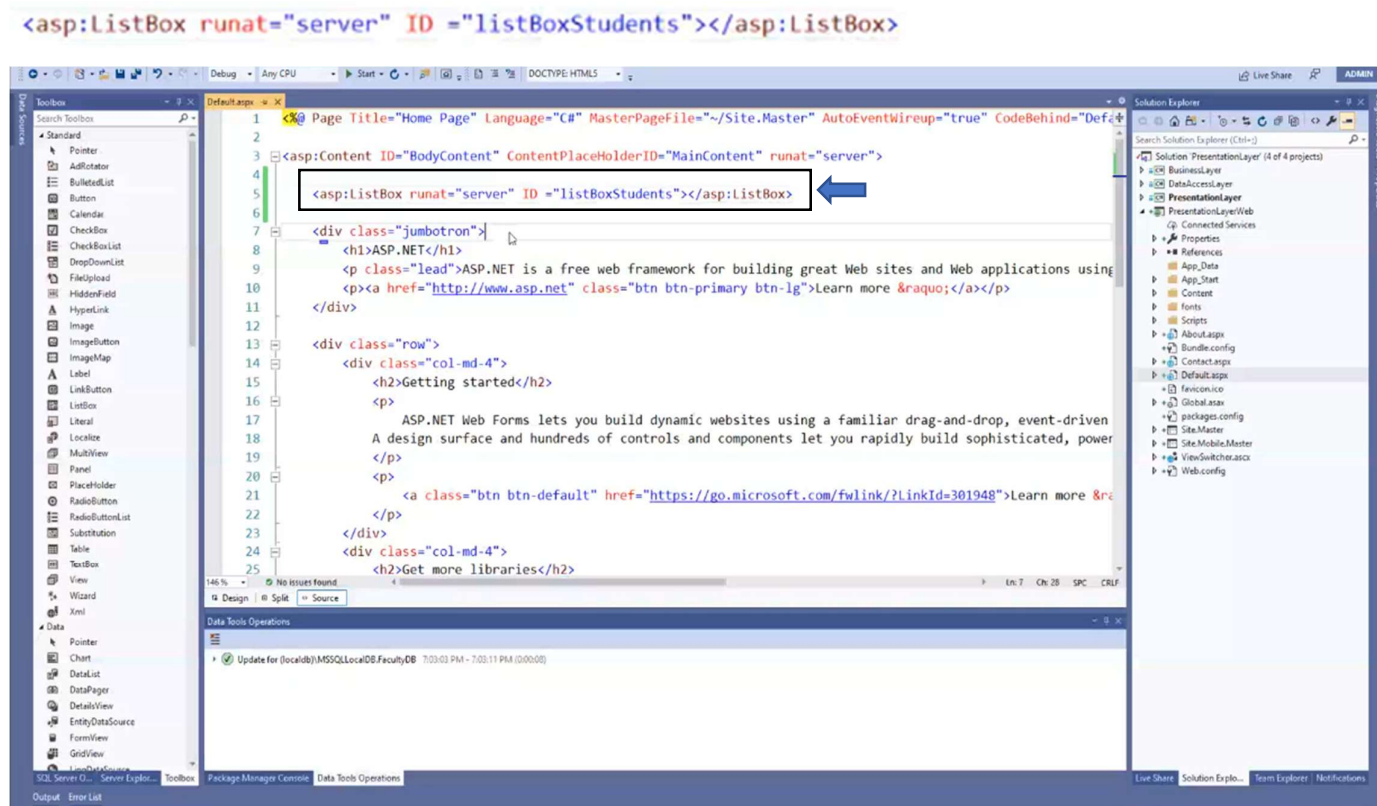
Create a new ASP.NET Web Application



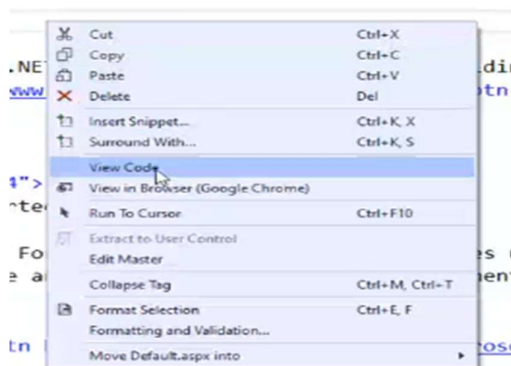
Nakon kreiranja projekta biamo **default** solution expolorer-u, nakon čega dobijamo sledeci prikaz:



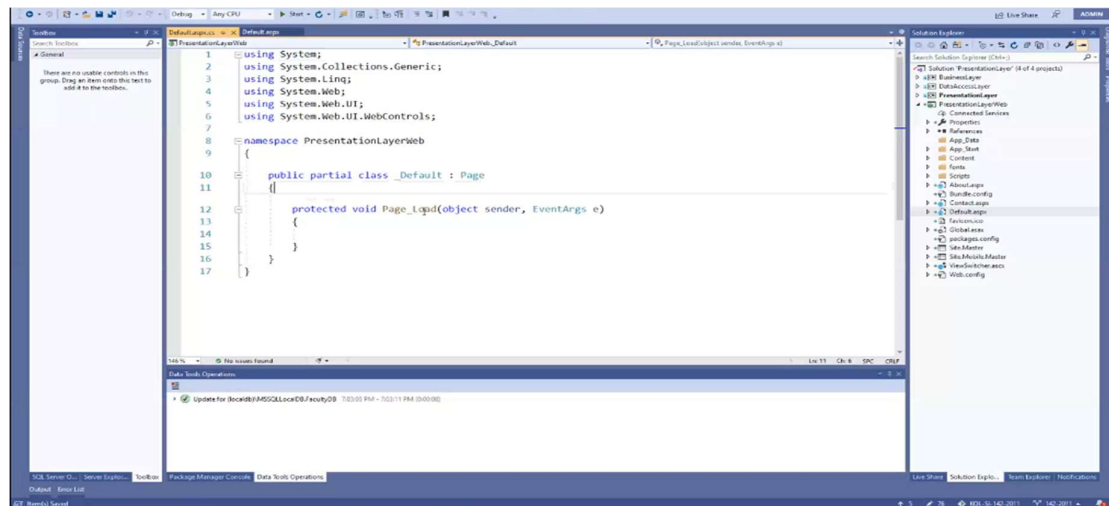
Pri vrhu na pocetnoj stranici cemo prikazati listu studenta to postizemo sledecim kodom I to ce biti to sto se tice html dela.



Sada je potrebno precu u c# kod, to radimo tako sto bilo gde kliknemo desni klik pa view code:



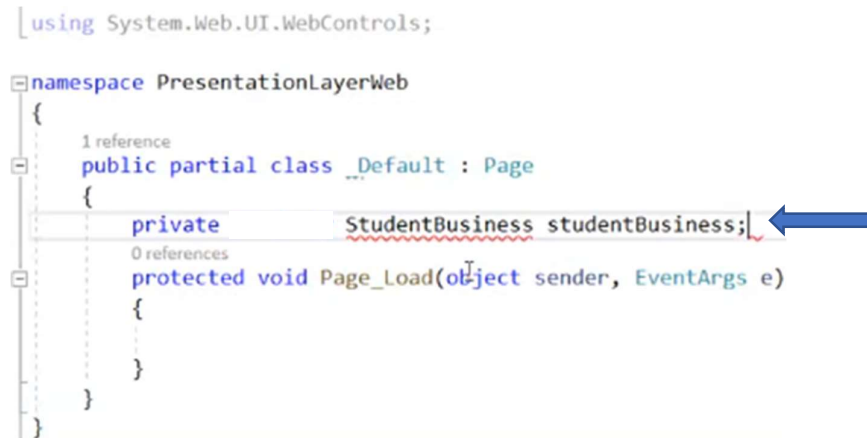
Nakon cega dobijamo sledeci prozor:



Nakon toga vršimo povezivanje na StudentBusiness:

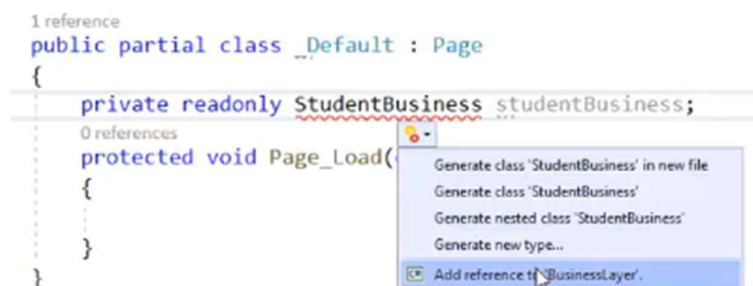
```
using System.Web.UI.WebControls;

namespace PresentationLayerWeb
{
    1 reference
    public partial class _Default : Page
    {
        private StudentBusiness studentBusiness;
        0 references
        protected void Page_Load(object sender, EventArgs e)
        {
        }
    }
}
```



Ako se javlja greska onda samo pridjemo misom I dodamo referencu:

```
1 reference
public partial class _Default : Page
{
    private readonly StudentBusiness studentBusiness;
    0 references
    protected void Page_Load(
    {
    }
}
```

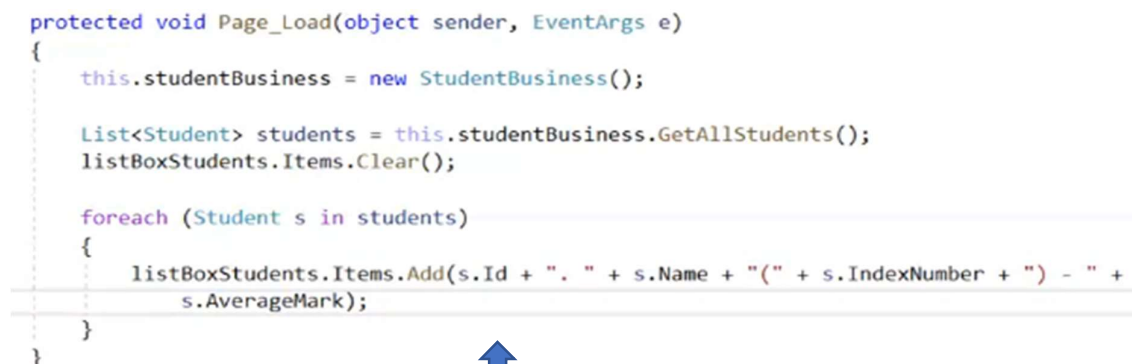


Zatim instanciramo I dovucemo listu studenata u metodi Page_Load:

```
protected void Page_Load(object sender, EventArgs e)
{
    this.studentBusiness = new StudentBusiness();

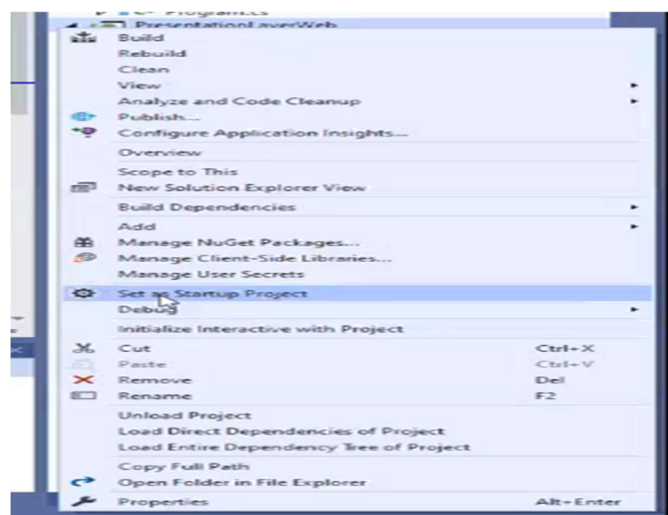
    List<Student> students = this.studentBusiness.GetAllStudents();
    listBoxStudents.Items.Clear();

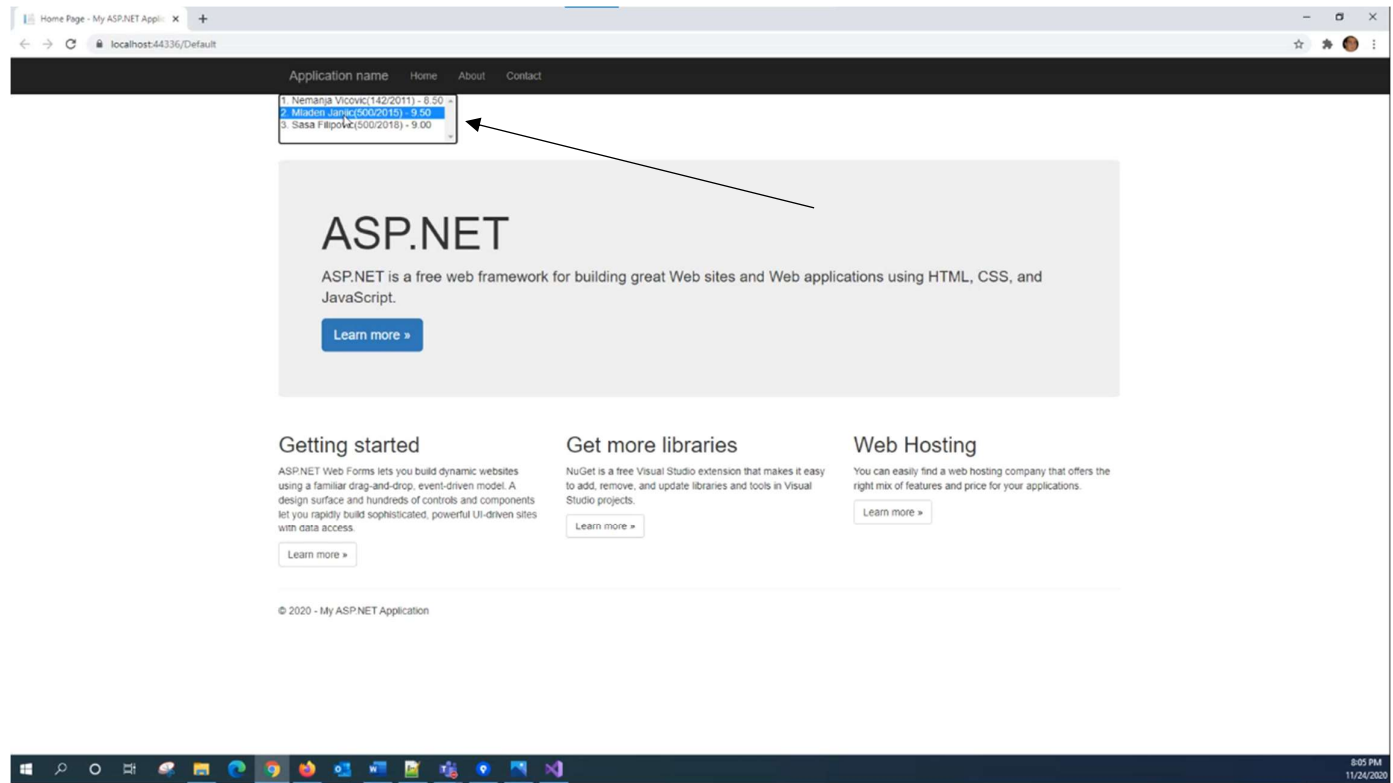
    foreach (Student s in students)
    {
        listBoxStudents.Items.Add(s.Id + ". " + s.Name + "(" + s.IndexNumber + ") - " +
            s.AverageMark);
    }
}
```



Ako se javi neka greska samo pridjemo misom I dodamo odgovarajuci using.

Na sledeci nacin pokrecemo projekat: desni klik na PresentationLayerWeb pa setan Startup Project





Kraj