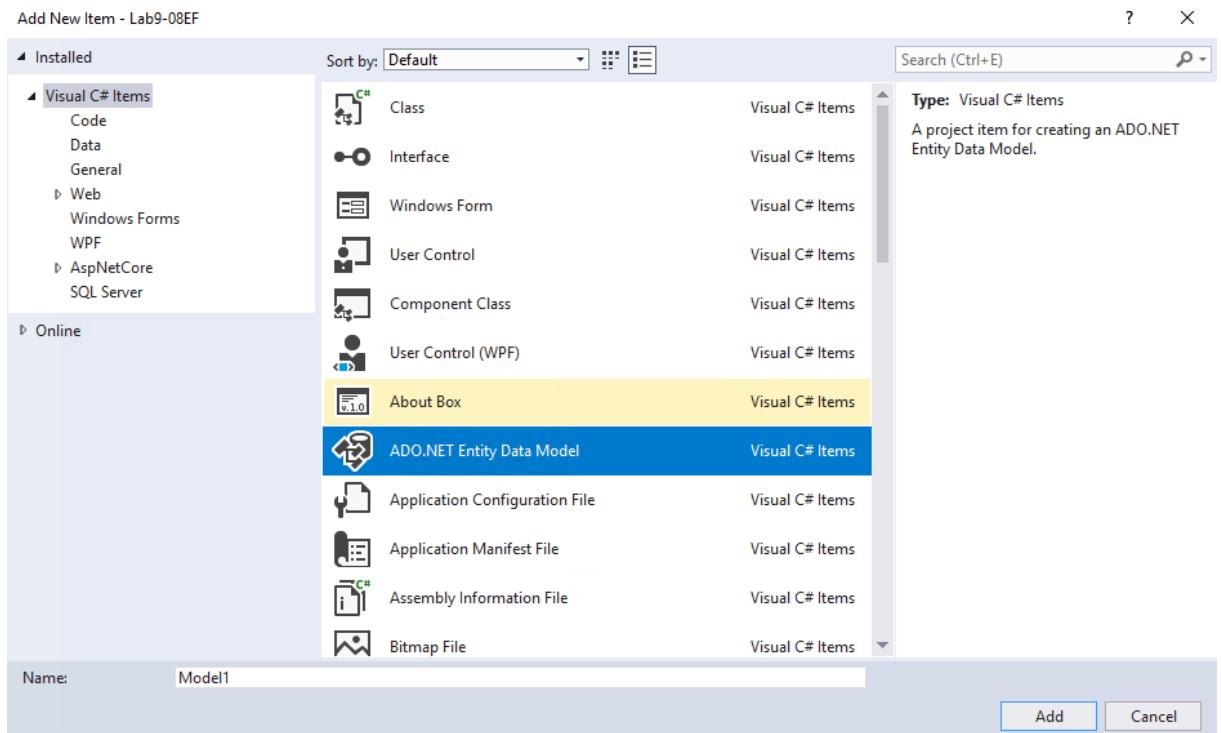
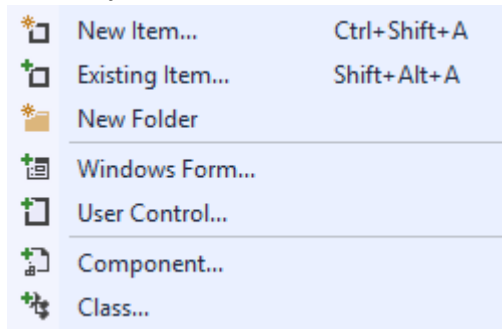


1

1. PPM Projekt-> Add new item ->ADO.NET Entity Data Model



2. EF Designer From Database



Choose Model Contents

What should the model contain?

EF Designer
from
databaseEmpty EF
Designer
modelEmpty Code
First modelCode First
from
database

Creates a model in the EF Designer based on an existing database. You can choose the database connection, settings for the model, and database objects to include in the model. The classes your application will interact with are generated from the model.

< Previous

Next >

Finish

Cancel



Choose Your Data Connection

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

GM.mdf

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://*/Model.csdl|res://*/Model.ssdl|
res://*/Model.msl;provider=System.Data.SqlClient;provider connection string="data source=
(LocalDB)\MSSQLLocalDB;attachdbfilename=C:\LabRepos\lab9\Lab9-05\Database\GM.mdf;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☒ Save connection settings in App.Config as:

GMEntities

< Previous

Next >

Finish

Cancel

Wybieramy tabele które chcemy przenieść do naszego modelu danych EDM

Zaznaczamy pluralize or singularize...

Entity Data Model Wizard



Choose Your Database Objects and Settings

Which database objects do you want to include in your model?

- ☒ Tables
 - ☒ dbo
 - ☒ GM.Magazyn
 - ☒ GM.PRACOWNICY
 - ☐ Views
 - ☐ Stored Procedures and Functions

☐ Pluralize or singularize generated object names

☒ Include foreign key columns in the model

☐ Import selected stored procedures and functions into the entity model

Model Namespace:

GMMModel

< Previous Next > **Finish** Cancel

5. Zmieniamy nazwę Form1 na Entity.

5a. Zmieniamy właściwość Text na Forma Główna.

6. Zmieniamy właściwość WindowsState na Maximized

7. Przeciągamy na płótno obiekt panel

8. Zmieniamy jego nazwę na pGlowny

9. Ustawiamy właściwość Dock dla Panelu na Top.

10. Przeciągamy na płótno obiekt DataGridView

12. Zmieniamy Właściwość obiektu dgPracownicy Dock na Fill.

13. Dodajemy do formy using System.Data.Entity

```
using System.Data.Entity;
```

14. Dodajemy obiekt encji do Formy

```
GMEntities entities = new GMEntities();
```

15. Dodajemy do konstruktora formy wczytanie tabeli pracowników.

```
Entities.GM_PRACOWNICY.Load();
```

16. Przypisujemy pracowników do Grida

```
dgPracownicy.DataSource = Entities.GM_PRACOWNICY.Local.ToBindingList();
```

17. Umieszczamy obiekt Button na płótnie formy.

18. Zmieniamy jego nazwę na btnDodaj.

19. Zmieniamy właściwość Text na Dodaj.

20. Klikając dwukrotnie na przycisk zdefiniujemy zdarzenie onClick.

21. Dodajemy nową formę do projektu PPM Project -> Add->New item-> Windows Forms

22. Zmieniamy Name z na FEdycjaPracownik.

23. Zmieniamy Właściwość Text Na Pracownik.

24. Umieszczamy obiekt Label na płótnie formy.

25. Zmieniamy jego nazwę na Imie.

26. Zmieniamy właściwość Text na Imię.

27 Umieszczamy obiekt TextBox na płótnie formy.

28. Zmieniamy jego nazwę na tbImie.

30. Umieszczamy obiekt Label na płótnie formy.

31. Zmieniamy jego nazwę na lNazwisko.

32. Zmieniamy właściwość Text na Nazwisko.

33. Umieszczamy obiekt TextBox na płótnie formy.

34. Zmieniamy jego nazwę na tbNazwisko.

36. Umieszczamy obiekt Label na płótnie formy.

37. Zmieniamy jego nazwę na lAdres.

38. Zmieniamy właściwość Text na Adres.

39. Umieszczamy obiekt TextBox na płótnie formy.

40. Zmieniamy jego nazwę na tbAdres.

41. Zmieniamy jego nazwę na lPesel.

42. Zmieniamy właściwość Text na Pesel.

43. Umieszczamy obiekt TextBox na płótnie formy.

44. Zmieniamy jego nazwę na tbPesel.

45. Zmieniamy jego nazwę na lIDl.

46. Zmieniamy właściwość Text na ID.

47. Umieszczamy obiekt NumericUpDown na płótnie formy.

48. Zmieniamy jego nazwę na tbID.

49. Umieszczamy obiekt Button na płótnie formy.

50. Zmieniamy jego nazwę na btnWyjscie.

51. Zmieniamy właściwość Text na Wyjscie.

52. Deklarujemy formie FEdycjaPracownik Klasę Encji

```
GMEntities Entities;
```

53. Zmieniamy konstruktor formy edycyjnej aby przyjmował jako parametr obiekt encji

```
public FEdycjaPracownik (GMEntities3 entities3)
{
    InitializeComponent();
    Entities = entities3;
}
```

54. Umieszczamy obiekt Button na płótnie formy.

55. Zmieniamy jego nazwę na btnZapisz.

56. Zmieniamy właściwość Text na Zapisz.

57. Klikając dwukrotnie na przycisk zdefiniujemy zdarzenie onClick.

58. Tworzymy obiekt GM_Pracownicy

```
GM_PRACOWNICY gm_PRACOWNICY = new GM_PRACOWNICY();
```

59. Przypisujemy wartości z formy do obiektu Gm_PRACOWNICY

```
gm_PRACOWNICY.Imie = tbImie.Text;
gm_PRACOWNICY.Nazwisko = tbNazwisko.Text;
gm_PRACOWNICY.Adres = tbImie.Text;
gm_PRACOWNICY.Pesel = tbNazwisko.Text;
```

60. Zaznaczamy kod przypisania i dokonujemy hermetyzacji do metody UstawDane

61. Ustawiamy ID Pracownika

```
gm_PRACOWNICY.Id = Convert.ToInt32(tbID.Text);
```

62. Sprawdzamy czy pracownik o danym ID istnieje już w bazie danych

```
GM_PRACOWNICY testPracownik = Entities.GM_PRACOWNICY.Where(x => x.Id ==
gm_PRACOWNICY.Id)?.FirstOrDefault();
```

63. Jeśli pracownik istnieje już w bazie danych wyświetlamy komunikat o błędzie

```
if (testPracownik != null)
    MessageBox.Show($"Istnieje w bazie już pracownik o numerze
ID=={tbID.Text}");
```

64. W przeciwnym wypadku dodajemy obiekt pracownika do zbioru

```
Entities.GM_PRACOWNICY.Add(gm_PRACOWNICY);
```

65. Zapisujemy zmiany

```
Entities.SaveChanges();
```

66. Zamykamy formę z dialogResult == OK

```
this.DialogResult = DialogResult.OK;
```

67. Klikając dwukrotnie na przycisk zdefiniujemy zdarzenie onClick dla Przycisku Wyjście.

68. Ustawiamy dialogResult na Cancel
`this.DialogResult = DialogResult.Cancel;`

69. Uzupełniamy Zdarzenie Click dla przycisku Dodaj

70. Tworzymy formę dla FEdycjaPracownik

```
FEdycjaPracownik nowy = new FEdycjaPracownik (Entities);
```

71 Wywołujemy formę FEdycjaPracownik

```
nowy.ShowDialog();
```

72. Odświeżamy Grida

```
dgPracownicy.Refresh();
```

73 Modyfikujemy formę FEdycjaPracownik

74. Dodajemy obiekt GM_PRACOWNIK do Formy

```
GM_PRACOWNICY pracownik;
```

75. Dodajemy konstruktor w formie FEdycjaPracownik ustawiający zmienną pracownik

```
public FEdycjaDanych(GMEntities3 entities3, GM_PRACOWNICY pracownik):this(entities3)
```

76. W konstruktorze przypisujemy parametr do zmiennej pracownik

```
this.pracownik= pracownik;
```

77. Ustawiamy kontrolkę tbID jako niedostępna

```
tbID.Enabled = false;
```

78. Przypisujemy do kontrolki ID pracownika

```
tbID.Text = pracownik.Id.ToString();
```

79. Przypisujemy do kontrolki Adres pracownika

```
tbAdres.Text = pracownik.Adres;
```

80. Przypisujemy do kontrolki Imie pracownika

```
tbImie.Text = pracownik.Imie;
```

81. Przypisujemy do kontrolki Nazwisko pracownika

```
tbNazwisko.Text = pracownik.Nazwisko;
```

82. Przypisujemy do kontrolki Pesel pracownika

```
tbPesel.Text = pracownik.Pesel;
```

83. Modyfikujemy konstruktor jedno parametrowy na formie FEdycjaPracownik

```
tbID.Enabled = true;
```

80 Modyfikujemy przycisk zapisz na FEdycjaPracownik

81. Enkapsujemy całość do kodu do metody NowyPracownik

```
private void NowyPracownik()
{
    GM_PRACOWNICY gm_PRACOWNICY = new GM_PRACOWNICY();
    gm_PRACOWNICY.Id = Convert.ToInt32(tbID.Text);
    UstawDane(gm_PRACOWNICY);

    gm_PRACOWNICY.Id = Convert.ToInt32(tbID.Text);

    GM_PRACOWNICY idtest = Entities.GM_PRACOWNICY.Where(x => x.Id ==
gm_PRACOWNICY.Id)?.FirstOrDefault();
    if (idtest != null)
        MessageBox.Show($"Istnieje w bazie już pracownik o numerze
ID=={tbID.Text}");
    else
    {
        Entities.GM_PRACOWNICY.Add(gm_PRACOWNICY);
        Entities.SaveChanges();
        this.DialogResult = DialogResult.OK;
    }
}
```

82 Tworzymy metodę edycji pracownika

```
private void EdycjaPracownika()
```

83. Wywołujemy metodę UstawDane dla zmiennej pracownik

```
UstawDane(pracownik);
```

84. Modyfikujemy stan obiektu pracownik na EntityState.Modified.

```
Entities.Entry(pracownik).State = EntityState.Modified;
```

85. Zapisujemy zmiany.

```
Entities.SaveChanges();
```

86. Ustawiamy dialog Result na ok.

```
this.DialogResult = DialogResult.OK;
```

Uzupełniamy przycisk Edytuj

87. Pobieramy aktualnego pracownika z Grida

```
int idPracownika =
Convert.ToInt32(dgPracownicy.Rows[dgPracownicy.CurrentRow.Index].Cells["ID"].Value);
```

88. Wybieramy obiekt pracownika ze zbioru danych

```
GM_PRACOWNICY gm_pracownik = Entities.GM_PRACOWNICY.Where(x => x.Id ==
idPracownika)?.FirstOrDefault();
```

89. Tworzymy formę dla FEdycjaPracownik

```
FEdycjaPracownik edycja = new FEdycjaPracownik (Entities, gm_pracownik);
```

90. Wywołujemy formę FEdycjaPracownik

```
edycja.ShowDialog();
```

91. Odświeżamy Grida

```
dgPracownicy.Refresh();
```

92. Modyfikujemy przycisk Usun

93. Pobieramy aktualnego pracownika z Grida

```
int idPracownika =  
Convert.ToInt32(dgPracownicy.Rows[dgPracownicy.CurrentRow.Index].Cells["ID"].Value);
```

94. Wybieramy obiekt pracownika ze zbioru danych

```
GM_PRACOWNICY gm_pracownik = Entities.GM_PRACOWNICY.Where(x => x.Id ==  
idPracownika)?.FirstOrDefault();
```

95. Usuujemy pracownika ze zbioru danych

```
Entities.GM_PRACOWNICY.Remove(gm_pracownik);
```

96. Zapisujemy zmiany

```
Entities.SaveChanges();
```

97. Modyfikujemy przycisk Odśwież

98. Odświeżamy Grida

```
dgPracownicy.Refresh();
```