Entity Framework + LINQ oktatási segédlet

Török János Zsolt

2.0. változat

Célok:

- Ismerkedés az Entity Framework –el
- Adatbázis lekérdezések létrehozása LINQ segítségével

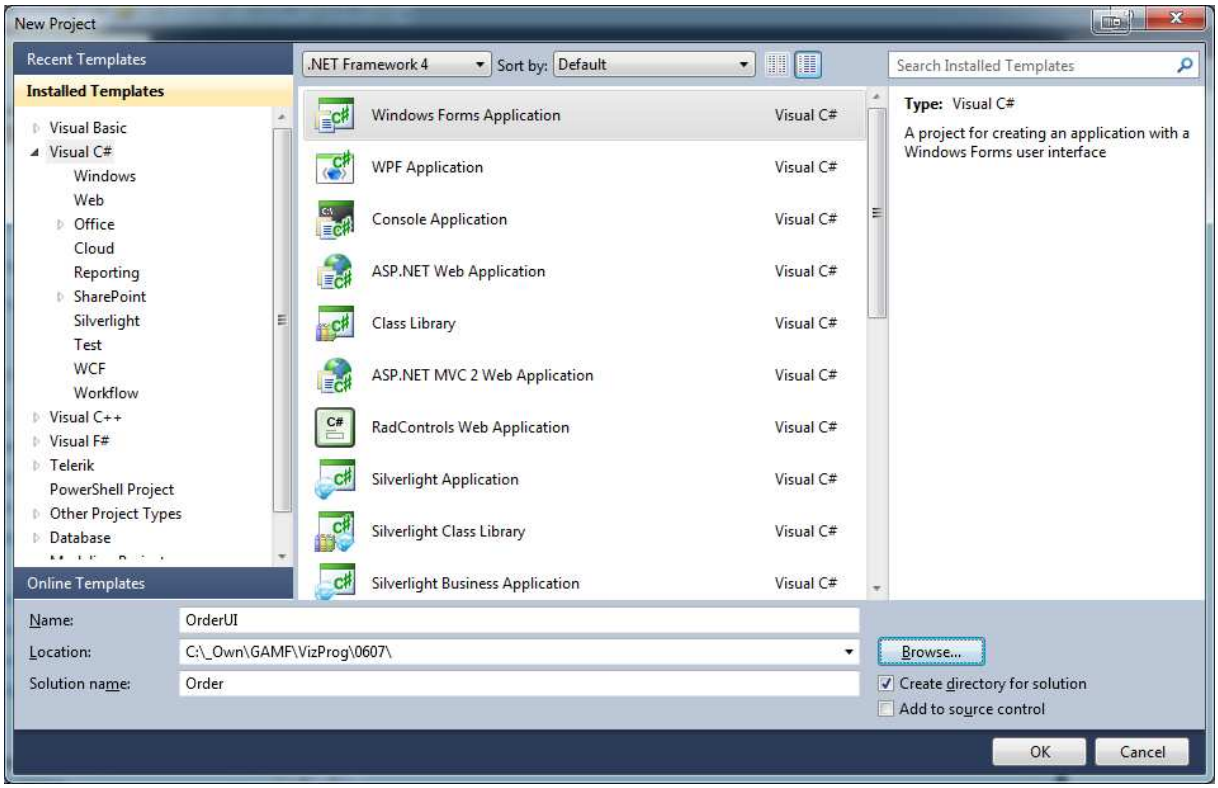
Feladat:

Megrendeléseket kezelő kétrétegű (UI + DAL) alkalmazás készítése.

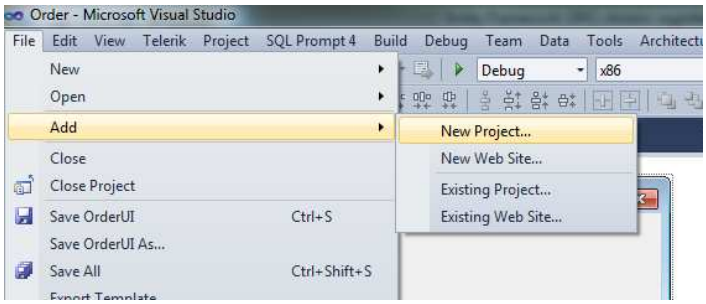
Lehetséges megvalósítás:

1. Projektek létrehozása

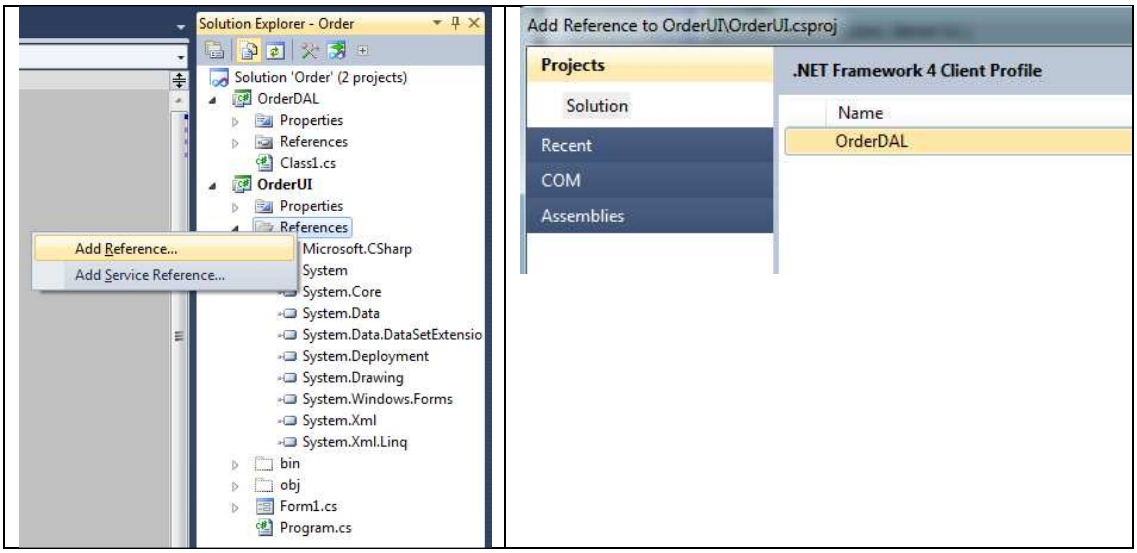
Első lépésként hozzuk létre a projekteket és hozzuk létre közöttük a kapcsolatot. A UI egy Windows Forms alkalmazás lesz, amely az adatok megjelenítéséért, módosításáért felel, a projekt neve OrderUI.



Ezután adjunk hozzá még egy projektet a solution-höz, amely legyen Class Library projekt és kapja a OrderDAL nevet. Ez a projekt lesz a felelős az adatbázis elérésért.



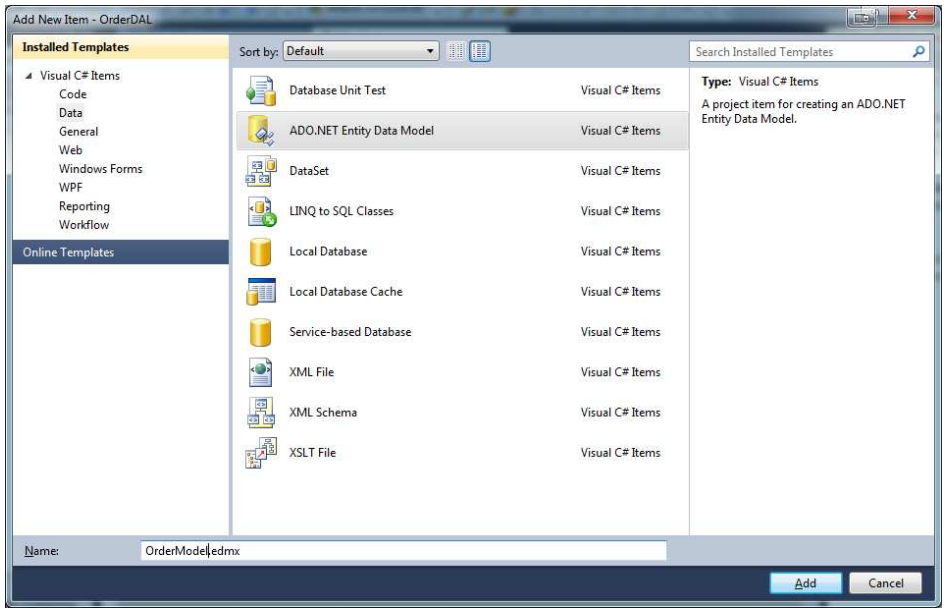
Végezetül adjuk hozzá a UI projekthez a DAL projekt referenciáját. Ezzel biztosítjuk, hogy a UI felől elérhetőek lesznek a DAL-on publikusként kivezetett objektumok és metódusok.



2. Entitás modell elkészítése Model First megközelítés segítségével

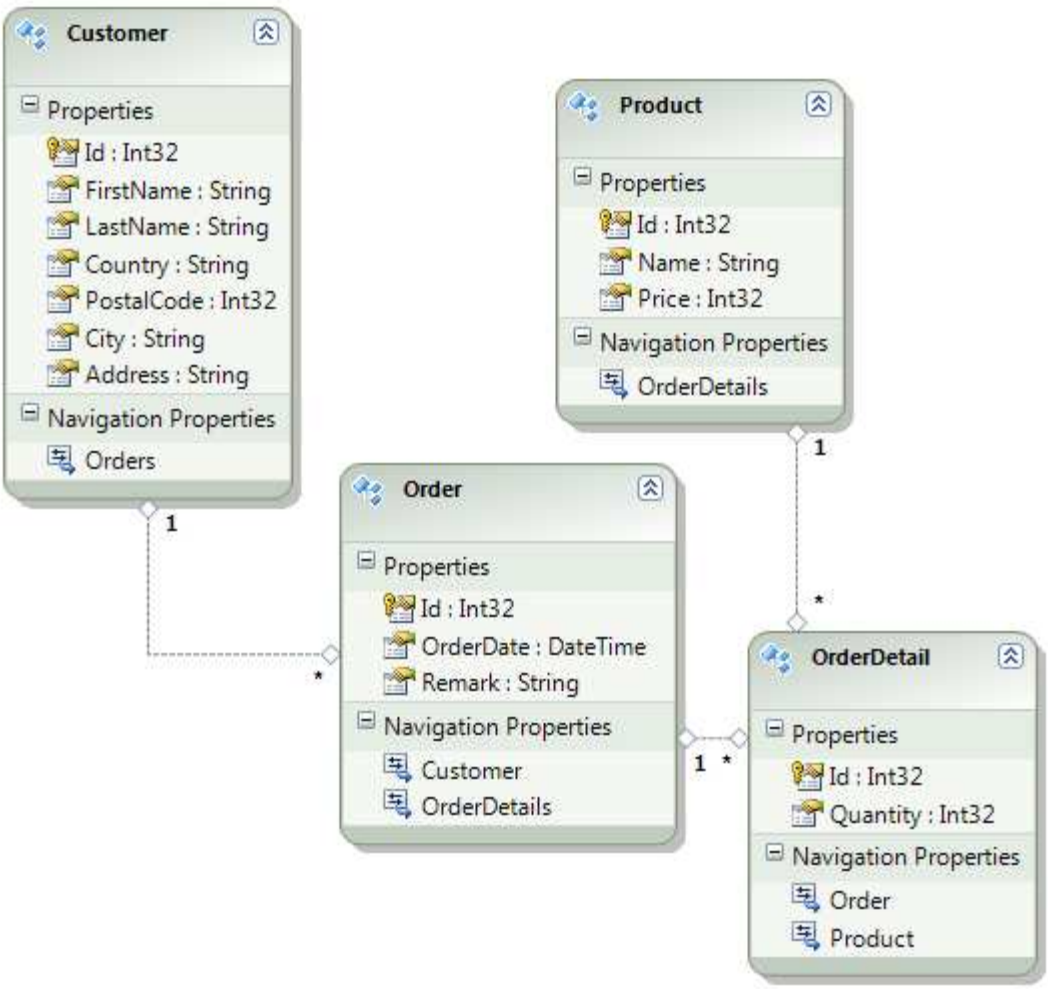
Tervezzük meg az adatokat tároló objektumokat, és ezzel a relációs adatbázist is a Model First megközelítés segítségével.

a. Az OrderDAL projekthez adjunk hozzá egy ADO.NET Entity Data Model elemet, amely az Entity Framwork-ös adatelérés központi eleme, magát a modellt fogja tartalmazni, azaz a objektumok és relációs adatok közötti összerendeléseket.



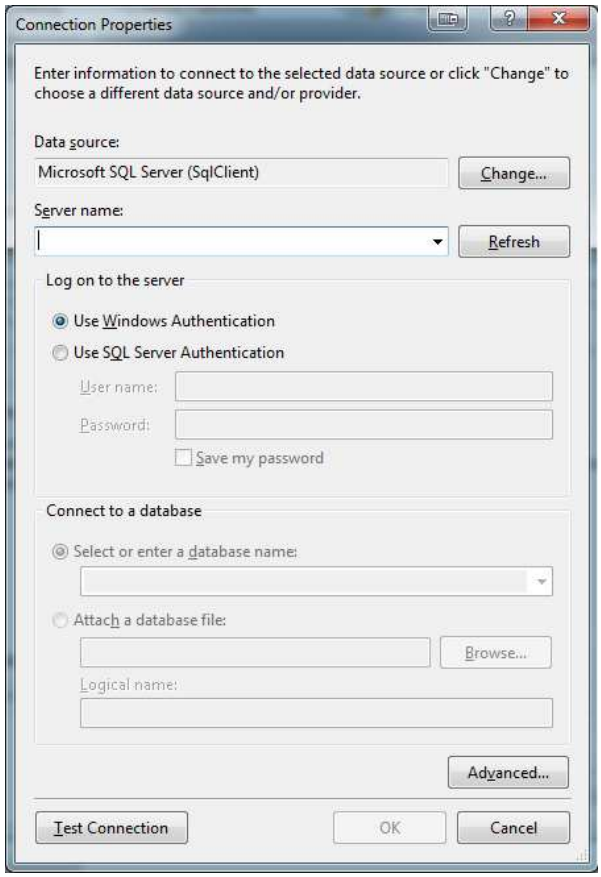
b. A „What should the model contain?” kérdés megjelenésekor válasszuk az Empty Model-t (Model First Approach). Ekkor megjelenik a model designer, amelyen meg lehet tervezni az adatmodellt.
c. A modell kialakítása előtt tegyünk meg még két dolgot: egyrészt a modell tulajdonságlapján állítsuk a „Pluralize new objects” tulajdonságot true-ra, ezzel biztosítva az entitás kollekciók megfelelő elnevezését. Másrészt kapcsoljuk be a skaláris tulajdonságok típusainak mutatását: jobb gomb a modell-en -> Scalar property format -> Display name and type.
d. A feladat megvalósításához szükségünk van a következő entitások tárolására-visszaolvasására:
    i. Vevők alapadatai (név, cím)
    ii. Termékek alapadatai (megnevezés, ár)
    iii. Rendelések (ki a megrendelő, rendelés ideje)
    iv. Rendelések részletei (mely termékeket és hány darabot tartalmaz a megrendelés?

Ezek alapján rajzoljuk meg az entitásokat (toolbox - entity), valamint a közöttük lévő kapcsolatokat (toolbox - association):

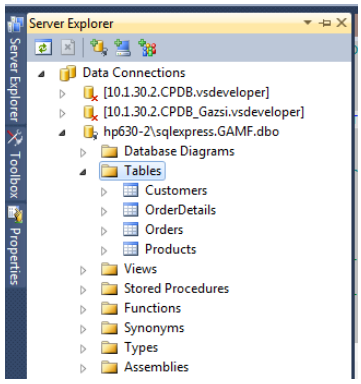


A modell felépítésekor figyeljünk oda a entity property-k helyes adattípusának beállítására, valamint a navigation property-k szingularizálására (ha beállítjuk a modellen a „Pluralize new objects” tulajdonságot true-ra, ezt a designer elvégzi helyettünk). További információk a model-first tervezéshez: http://msdn.microsoft.com/en-us/data/ff830362

e. Végezetül generáljuk le az adatbázis szkriptet, és futtassuk le az adatbázison: jobb egérgomb a modellen és válasszuk a Generate Database from Model… menüpontot. Első lépésként az adatbázishoz történő csatlakozás paramétereit kell beállítanunk. Amennyiben nincs a legördülő listában megfelelő connection string, úgy a New Connection gombra kattintva tudjuk beállítani a SQL szerver paramétereit. A legfontosabb a szerver nevét és a login metódust beállítani, ezután pedig az adatbázist kiválasztani.



f. A Next gombra kattintva létrejön az adattáblákat és a köztük lévő kapcsolatokat leíró adatbázis szkript, amely a Finish-re kattintva megnyitásra is kerül a Visual Studio-ban az DAL projekt részeként. Ezzel a lépéssel az adatbázis táblák és oszlopok automatikusan hozzárendelődnek a Model Designer-ben megtervezett objektuomokhoz és azok property-jeihez. Az Execute SQL ikonra kattintva a szkript futtatásra kerül a connection string-ben meghatározott adatbázison. A Server Explorer Data Connections szekciójában leellenőrizhetjük a létrejött adatbázis struktúrát:



3. Az adatelérő osztály implementálása

Ahhoz, hogy a modellünkön deklarált objektumokat el tudjuk érni a UI-on, ki kell vezetni őket. A kivezetéshez írjuk át a DAL projekt létrehozásakor automatikusan létrejött Class1 osztályunkat: az osztály neve legyen OrderDALC, tagváltozója legyen a példányosított OrderModelContainer, amelyen keresztül elérhetőek az entitásaink és a köztük lévő kapcsolatok, valamint definiáljunk két metódust.

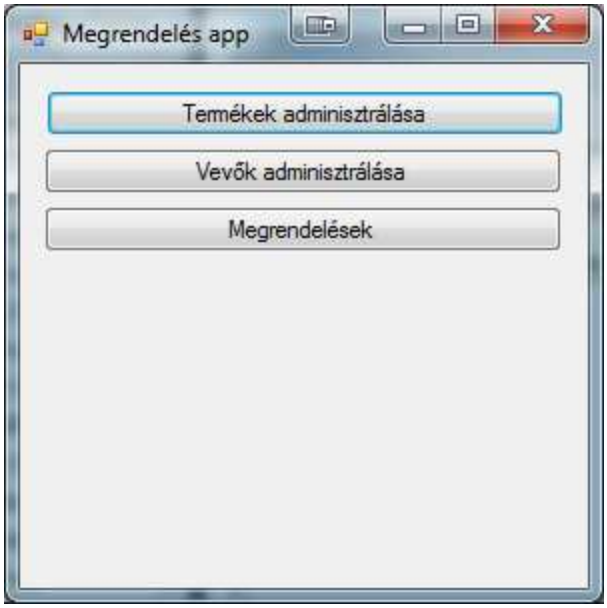
Az egyik az adatbázisban található összes terméket adja vissza, a másik pedig visszamenti az adatbázisba a modelcontainer-en történt változtatásokat:

```
public class OrderDALC
{
    OrderModelContainer omc = new OrderModelContainer();

    public IEnumerable<Product> GetProducts()
    {
        return omc.Products.OrderBy(p => p.Name);
    }

    public int SaveChanges()
    {
        return omc.SaveChanges();
    }
}
```

A későbbiekben ezt az osztályt fogjuk bővíteni metódusokkal, amikor további adatokra lesz szükségünk az adatbázisból.

4. A fő ablak megtervezése

Először alakítsuk át a fő form-ot, amely a projekt létrehozásakor a Form1 nevet. Nevezzük át frmMain-re és design-oljuk meg a következőképpen:



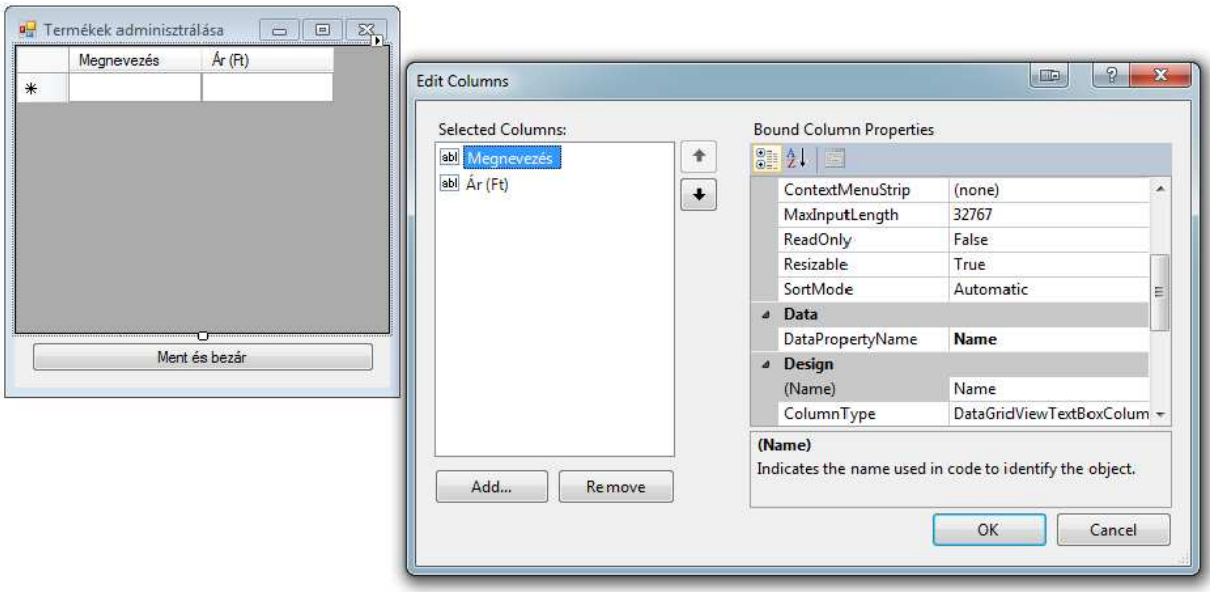
Ahhoz hogy az alkalmazás elérhesse az adatokat az OrderDAL dll-en keresztül két dolgot kell még beállítani: a System.Data.Entity referenciát is hozzá kell adni az OrderUI projekthez, valamint az App.config-ot át kell másolni az OrderDAL projektből, mert az OrderDALC példányosításakor szükség van a connectionstring-re.

A Termékek adminisztrálása gombra kattintva ugorjon fel egy új form, amelyen a termékeket lehet feltölteni / módosítani:

```
private void btnProducts_Click(object sender, EventArgs e)
{
    frmProducts productsForm = new frmProducts(dalc);
    productsForm.Show();
}
```

5. Termékek adminisztrálása

A termékek form-ot tervezzük meg a következőképp: az adatokat datagridview kontroll segítségével menedzseljük. A kontrollon jobb gombbal kattintva nyílik lehetőség az oszlopok szerkesztésére. Itt vegyük fel a megjelenítendő oszlopokat figyelve arra, hogy az összrendeléshez az oszlopok DataPropertyName property-jét is kitöltsük a termék entitás megfelelő property-jével.



Az frmProducts osztály code behind-ja:

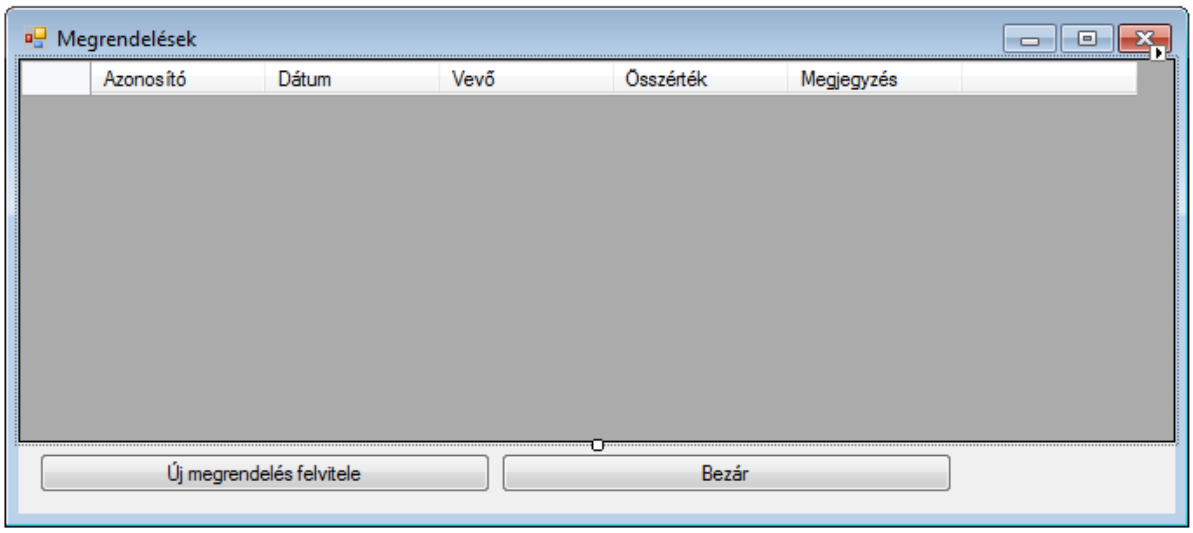
```
public partial class frmProducts : Form
{
    OrderDALC dalc;

    public frmProducts()
    {
        InitializeComponent();
    }

    public frmProducts(OrderDALC paramDalc)
    {
        InitializeComponent();

        dalc = paramDalc;
        dgvProducts.AutoGenerateColumns = false;
        dgvProducts.DataSource = dalc.GetProducts();
    }

    private void btnSave_Click(object sender, EventArgs e)
```

```
        {
            dalc.SaveChanges();
            this.Close();
        }
    }
```

Ugyanilyen módon implementáljuk a vevők adminisztrálását is.

6. Megrendelések form implementálása

Az frmOrders szolgál a meglévő megrendelések listázására és ezen a form-on helyezzük el azt a gombot, amely segítségével új megrendelést fogunk tudni felvenni:



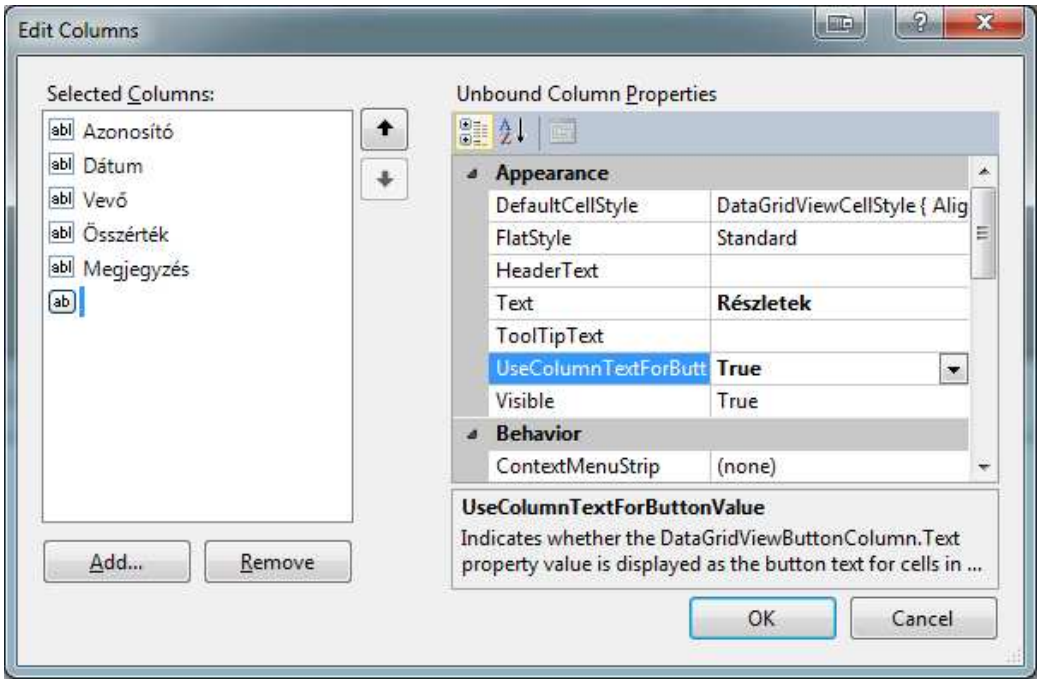
Az összérték oszlop egy lehetséges megvalósítása mutatja igazán az erejét az Entity Framework-nek: tekintve, hogy az Entity Framework a létrehozott entitás osztályokat parciálisként implementálja, nekünk is lehetőségünk van új tulajdonságokat és metódusokat rendelni az entitáshoz. Adjunk hozzá egy „új” class-t az OrderDALC projektünkhöz, amelyben egy új TotalValue tulajdonságban nyerjük ki az adott rendelés összértékét:

```
public partial class Order
{
    public int TotalValue
    {
        get
        {
            return this.OrderDetails.Sum(d => d.Quantity * d.Product.Price);
        }
    }
}
```

Az összértékhez hasonló módon implementáljuk a vevő oszlopot is (itt érdemes két lépésben gondolkodni: először egy FullName property-t hozzáadni a Customer osztályhoz, majd az Order osztályt kiegészíteni CustomerName property-vel).

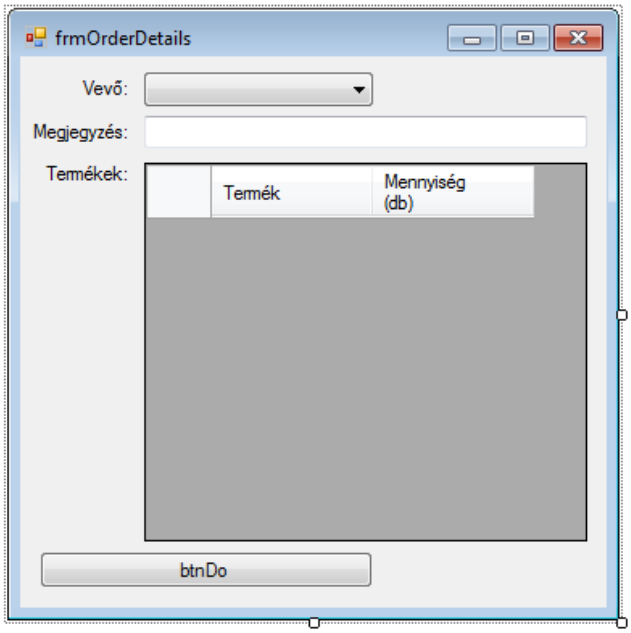
A helyes működéshez állítsuk false-ra a datagridview-n az AllowUserToAddRows és az AllowUserToDeleteRows tulajdonságokat.

A rendelés részleteinek megtekintéséhez adjunk egy DataGridViewButtonColumn-t a grid-hez:



7. Új megrendelés valamint meglévő megrendelés részletei

Az új megrendelésekhez, valamint a meglévő megrendelések részleteinek a megtekintéséhez ugyanazt a form-ot fogjuk használni. Ehhez hozzuk létre az frmOrderDetails nevű form-ot a következő design-al:



Az frmOrders form-on a következőképp szabályozzuk az frmOrderDetails form megjelenítését:

```
private void btnNew_Click(object sender, EventArgs e)
{
    frmOrderDetails orderDetailsForm = new frmOrderDetails(dalc, null);
    if (orderDetailsForm.ShowDialog() == DialogResult.OK)
    {
        // Because of lazy loading ToList() forces to reload data from db
        dgvOrders.DataSource = dalc.GetOrders().ToList();
```

```
        }
    }

    private void dgvOrders_CellClick(
        object sender, DataGridViewCellEventArgs e)
    {
        int orderId = int.Parse(
            dgvOrders.Rows[e.RowIndex].Cells[0].Value.ToString());
        Order order = dalc.GetOrder(orderId);
        frmOrderDetails orderDetailsForm = new frmOrderDetails(dalc, order);
        orderDetailsForm.ShowDialog();
    }
```

Meglévő megrendelés megjelenítéséhez szükség van magára a megrendelésre, ehhez kiegészítjük az OrderDALC osztályt a következő metódussal:

```
    public Order GetOrder(int orderId)
    {
        return omc.Orders.Single(o => o.Id == orderId);
    }
```

Új megrendelés és a részleteinek elmentéséhez további metódusokkal egészítjük ki az adatelérő komponensünket:

```
    public Order AddOrder(string remark, Customer customer)
    {
        Order order = new Order()
        {
            OrderDate = DateTime.Now,
            Remark = remark,
            Customer = customer
        };
        omc.Orders.AddObject(order);
        return order;
    }

    public void AddOrderDetail(Order order, Product product, int quantity)
    {
        OrderDetail detail = new OrderDetail()
        {
            Order = order,
            Product = product,
            Quantity = quantity
        };
        omc.OrderDetails.AddObject(detail);
    }
```

Ezek után az frmOrderDetails code behind-ja a következőképp alakul:

```
public partial class frmOrderDetails : Form
{
    OrderDALC dalc;
    Order order;

    public frmOrderDetails()
    {
```

```
        InitializeComponent();
    }

    public frmOrderDetails(OrderDALC paramDalc, Order paramOrder)
    {
        InitializeComponent();

        dalc = paramDalc;
        order = paramOrder;

        cbCustomers.DataSource = dalc.GetCustomers();
        cbCustomers.ValueMember = "Id";
        cbCustomers.DisplayMember = "FullName";

        dgvOrderDetails.AutoGenerateColumns = false;
        IEnumerable<Product> products = dalc.GetProducts();
        foreach (Product product in products)
            dgvOrderDetails.Rows.Add(new object[] {product, product.Name, 0});

        if (order == null)
        {
            this.Text = "Új megrendelés felvitele";
            btnDo.Text = "Ment és bezár";
        }
        else
        {
            this.Text = order.Id + ". számú megrendelés részletei";

            cbCustomers.SelectedItem = order.Customer;
            cbCustomers.Enabled = false;

            txtRemark.Text = order.Remark;
            txtRemark.ReadOnly = true;

            dgvOrderDetails.Enabled = false;
            foreach (OrderDetail detail in order.OrderDetails)
            {
                foreach (DataGridViewRow row in dgvOrderDetails.Rows)
                {
                    if (row.Cells["Product"].Value == detail.Product)
                        row.Cells["Quantity"].Value = detail.Quantity;
                }
            }

            btnDo.Text = "Bezár";
        }
    }



    private void btnDo_Click(object sender, EventArgs e)
    {
        if (order == null)
        {
            Order newOrder = dalc.AddOrder(
                txtRemark.Text, (Customer)cbCustomers.SelectedItem);

            foreach (DataGridViewRow row in dgvOrderDetails.Rows)
            {
```

```
                if (row.Cells["Quantity"].Value.ToString() != "0")
                {
                    dalc.AddOrderDetail(
                        newOrder, (Product)row.Cells["Product"].Value,
                        int.Parse(row.Cells["Quantity"].Value.ToString()));
                }
            }

            dalc.SaveChanges();
        }

        this.Close();
    }
}
```

További feladatok

- A program ne engedjen törölni olyan vevőket illetve termékeket, amelyek már szerepelnek valamely megrendelésben!
- Alakítsa át az frmOrderDetails form-ot úgy, hogy az csak annyi terméket mutasson a grid-ben, ahány a megrendeléshez tartozik! Ne engedjen termék nélküli megrendeléseket felvinni! Figyeljen arra is, hogy amennyiben mentéskor ugyanaz a termék többször is szerepel, akkor dobjon fel figyelmeztető ablakot és biztosítson lehetőséget az ablakban a végleges darabszám beírására.