Jegyzőkönyv

# Programozás az Entity Framework használatával

|  |  |
| --- | --- |
| Gyakorlat adatok | |
| Név | **Csatlós Tamás Péter** |
| Neptun kód | **TALXCE** |
| Hallgató tanszéke/szakiránya | **IIT, Rendszerfejlesztő** |
| Gyakorlatvezető |  |
| Gyakorlat helye | **IL206** |
| Gyakorlat időpontja | **2016.04.18.** |
| Gyakorlat sorszáma | **2.** |

|  |  |
| --- | --- |
| Feladatlista | |
| 1. feladat |  |
| 2. feladat |  |
| 3. feladat |  |
| 4. feladat |  |
| 5. feladat |  |
| 6. feladat |  |
| 7. feladat |  |
| 9. feladat |  |
| 10. feladat |  |
| 11. feladat |  |
| 12. feladat |  |
| 14. feladat |  |

# Feladatok

### 2.feladat

**Feladat neve és leírása:**

Készíts egy függvényt TermekListazas() néven, mely kiírja a termékek neveit.

**Megoldás:**

static void TermekListazas()

{

using (var context = new LABOREntities())

{

foreach (var termek in context.Termek)

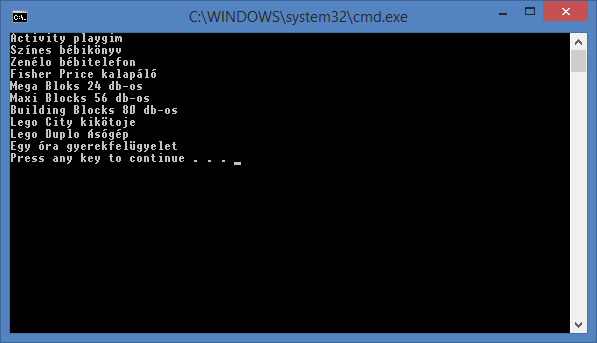
{

Console.WriteLine("{0}", termek.Nev);

}

}//Lezarja a kapcsolatot

}



**Indoklás:**

Az adatbázis kapcsolatot egy LABOREntities tipusú objektum reprezentálja. Using blokban használva biztosított a kapcsolat lezárása. A függvény kiírja az összes termék nevét. Képesek vagyunk egy egyszerű ciklussal lekérni az összes rekordot és azon belül a név attribútumot. Először a foreach első iterációjakor nyúlunk az adatbázishoz a Lazy loading miatt. Megfigyelhető az is, hogy a táblát úgy használom mint egy kollekció. Egy rekord ebből a kollekcióból egy elem, és elérem az attríbutumait mint tagváltozók.

### 3.feladat

**Feladat neve és leírása:**

Készíts egy előzőhöz hasonló függvényt HatekonyTermekListazas() néven, mely csak a készleten lévő termékeket írja ki, a termékeket betűrendbe rendezi, illetve hogy az adatbázisból csak a megjelenítendő adatokat kéri le!

**Megoldás:**

static void HatekonyTermekListazas()

{

using (var context = new LABOREntities())

{

IOrderedQueryable<Termek> query = from t in context.Termek

where t.Raktarkeszlet > 0

orderby t.Nev

select t;

foreach (var termek in query)

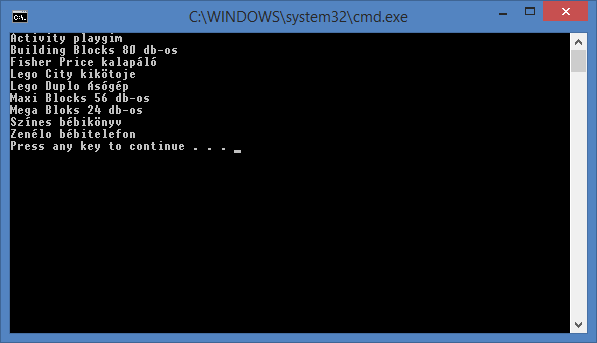
{

Console.WriteLine("{0}", termek.Nev);

}

}

}



**Indoklás:**

Ebben a feladatban használtam a Linq által nyújtott másik módszert az adatok eléréséhez és szűréséhez. A query lekérdezés adatbázis oldalon egy SQL utasításként fog lefutni. A típusa a kódban látható és leszármazik az IEnumerable interfészből is.

### 4.feladat

**Feladat neve és leírása:**

Készíts egy függvényt TermekListazasBrutto() néven mely a termékek neveit és bruttó árait írja ki.

**Megoldás:**

static void TermekListazasBrutto()

{

using (var context = new LABOREntities())

{

var query = from t in context.Termek

orderby t.Nev

select new

{

Nev = t.Nev,

Brutto = t.NettoAr \* (100 + t.AFA.Kulcs) / 100

};

foreach (var termek in query)

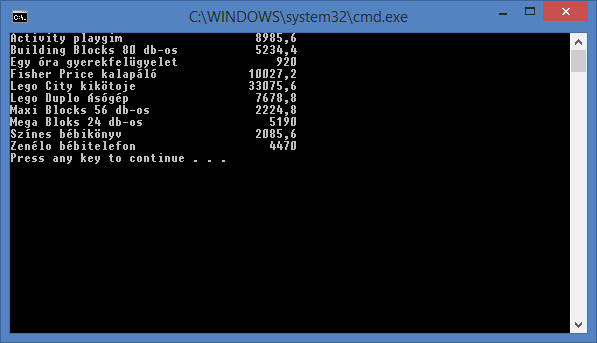
{

Console.WriteLine("{0, -25} {1, 15}", termek.Nev, termek.Brutto);

}

}

}



**Indoklás:**

Feladat megoldásához egy anonymus osztályt használtam. Ebben két tagváltozót hozok létre és ezeket felhasználom a kiíratásnál. Ki tudom számolni a bruttó árát egy terméknek, úgy hogy az attribútumait használom a fent látható módon.

### 5.feladat

**Feladat neve és leírása:**

Készíts egy függvényt Megrendelesek() néven, mely felsorolja a termékeket és minden termék mellett feltünteti a megrendelések összegét! üÜyelj rá, hogy a listázás egyetlen SQL utasítás végrehajtásával történjen

**Megoldás:**

static void Megrendelesek()

{

using (var context = new LABOREntities())

{

var query = from t in context.Termek

orderby t.Nev

select new

{

Nev = t.Nev,

Ossz = t.MegrendelesTetel.Sum(m => m.Mennyiseg)

};

foreach (var termek in query)

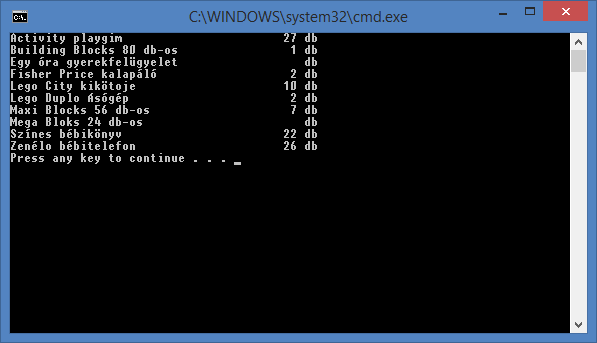
{

Console.WriteLine("{0, -25} {1, 15} db", termek.Nev, termek.Ossz);

}

}

}



**Indoklás:**

Mivel az IEnumerable interfészből származik a query típusa, ezért alkalmazhatóak a lambda kifejezések is.

### 6.feladat

**Feladat neve és leírása:**

Készíts függvényt Eladas() néven mely a 3-as azonosítójú termék készletét 1-gyel csökkenti, majd kiírja az új készletet.

**Megoldás:**

static void Eladas()

{

using (var context = new LABOREntities())

{

//SQL: update utasítás lenne

Termek termek = context.Termek.Single(t => t.ID == 3);

Console.WriteLine(termek.Raktarkeszlet);

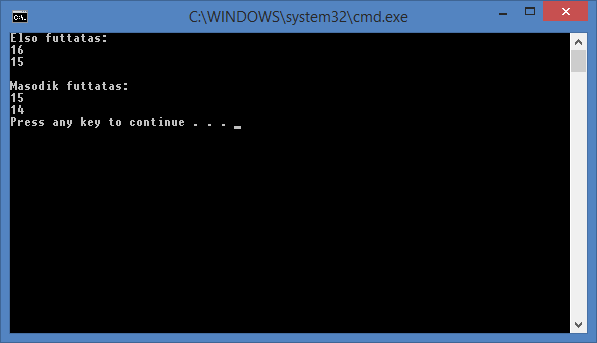
termek.Raktarkeszlet--;

Console.WriteLine(termek.Raktarkeszlet);

context.SaveChanges();// explicit nekünk kell megadni!

}

}



**Indoklás:**

Feladathoz kétszer egymás után hívtam meg a függvényt és látható, hogy a raktárkészlet csökken minden hívás után. SQL utasításként ez UPDATE parancsnak felel meg. A végén nekünk kell gondoskodni a változások mentéséről. Továbbá a Single() metódus hibát dob, ha nincs mivel visszatérnie.

### 7.feladat

**Feladat neve és leírása:**

Készíts egy függvényt Beszallitas() néven, mely felvesz egy új kategóriát és egy abba tartozó új terméket az adatbázisba, végül pedig kiírja az új sorok azonosítóit. A művelet során ne használjuk ki, hogy ismerhetjük az új kategória azonosítóját

**Megoldás:**

static void Beszallitas()

{

using (var context = new LABOREntities())

{

Kategoria kategoria = new Kategoria();

kategoria.Nev = "Csonthejasok";

Termek termek = new Termek();

termek.Nev = "Nagy kakas";

termek.NettoAr = 420;

termek.Kategoria = kategoria;

context.Termek.Add(termek);

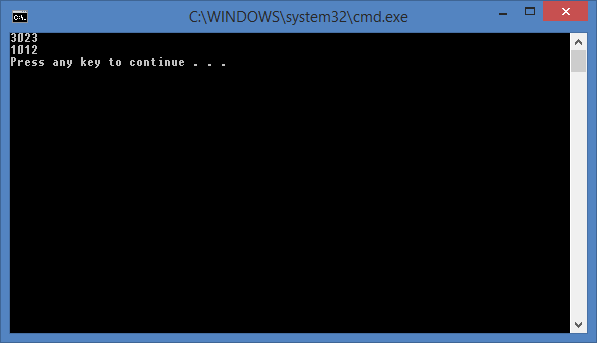
context.SaveChanges();

Console.WriteLine(kategoria.ID);

Console.WriteLine(termek.ID);

}

}



**Indoklás:**

Beszúrás és az adatok egyéb módú módosítása nem nehéz feladat az EF segítségével. A kategóriát el lehetne külön is menteni, de a navigációs property-k miatt elég értékül adni a terméknek és a terméket visszaírni az adatbázisba.

### 8.feladat

**Feladat neve és leírása:**

Az Entity Framework segítségével válaszd szét az ÁFA mentes számlatételket a többitől osztály szinten is. Ennek érdekében vezess be egy AfaMentesSzamlaTetel entitástípust, mely a SzamlaTetel típussal származási relációban van! Megoldásod tesztelésére készíts egy függvényt AfaMentesTetelek() néven, mely kilistázza az összes számlatételt, megjelölve azok osztályát! (Tipp: az alapértelmezett adatbázisban nincs 1-es kódú tétel, a megfelelő teszteléshez új számlatételeket kell felvenned, vagy meglévőket módosítani. Ezt megteheted közvetlenül az SQL Server Management Konzol használatával is.)

**Megoldás:**

static void AfaMentesTetelek()

{

using (var context = new LABOREntities())

{

foreach (var tetel in context.SzamlaTetel)

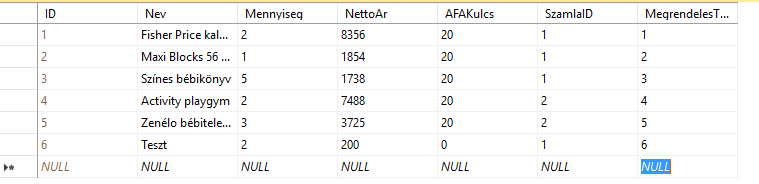
{

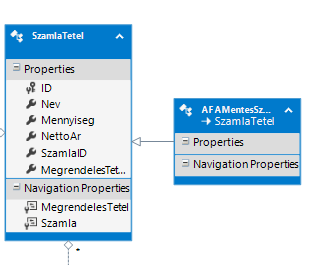
Console.WriteLine("{0}\t{1}", tetel.Nev, tetel.GetType());

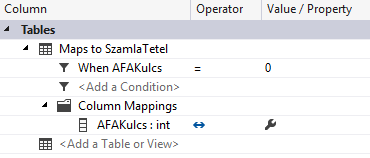
}

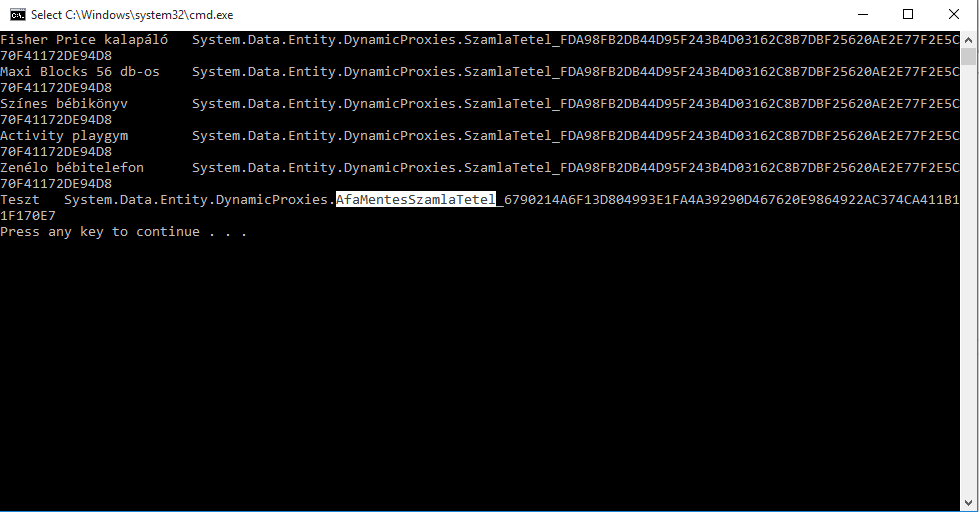
}

}









**Indoklás:**

A feladat megoldásához az első képen látható Teszt nevű rekordot szúrtam be a számla tétel táblába (1.kép). Továbbá leszármaztattam a Számla Tétel entitásból egy Áfa mentes számla tétel entitást (2.kép) és beállítottam a leképezést is (3. kép). Végül futattam a metódust és látható, hogy a Teszt nevű Számla Tétel típusa más, mint a többié (4.kép).

### 9.feladat

**Feladat neve és leírása:**

Készíts egy függvényt VevoListazas() néven, mely kiírja a vevők neveit

**Megoldás:**

static void VevoListazas()

{

using (var context = new LABOREntities())

{

foreach (var vevo in context.Vevo)

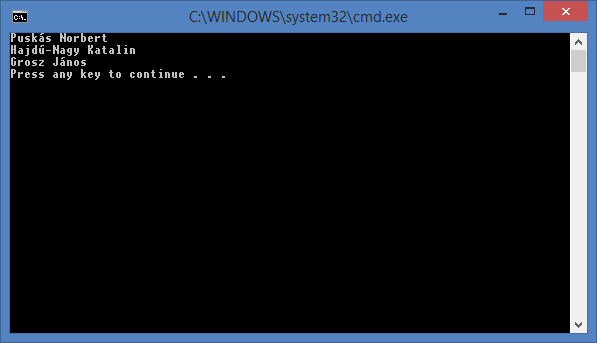
{

Console.WriteLine("{0}", vevo.Nev);

}

}

}



**Indoklás:**

A 2.feladathoz nagyon hasonló ez a feladat, csak a vevőket kellett kilistázni.

### 10.feladat

**Feladat neve és leírása:**

Készíts egy előzőhöz hasonló függvényt HatekonyVevoListazas() néven, mely csak az „o” betűt tartalmazó nevű vevőket listázza ki, a vevőket betűrendbe rendezi, illetve az adatbázisból csak a megjelenítendő adatokat kéri le.

**Megoldás:**

static void HatekonyVevoListazas()

{

using (var context = new LABOREntities())

{

var query = from v in context.Vevo

where v.Nev.Contains("o")

orderby v.Nev

select v.Nev;

foreach (var vevo in query)

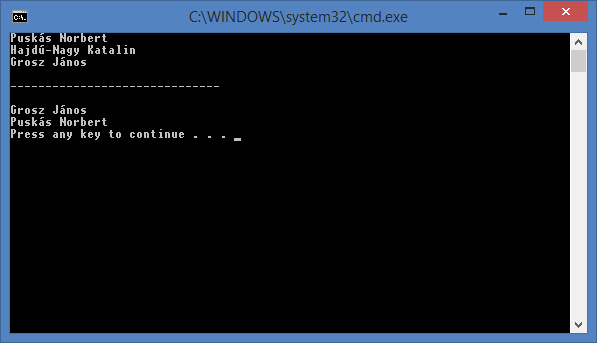
{

Console.WriteLine("{0}", vevo);

}

}

}



**Indoklás:**

A lekérdezés where feltételében vizsgálom, hogy van-e a névben „o” betű. A lekérdezés az „o” betűt tartalmazó neveket adja vissza. Összehasonlításképp lefuttattam a vevők listázását is, hogy ellenőrizzem a működését a függvénynek.

### 11.feladat

**Feladat neve és leírása:**

Készíts egy függvényt VevoTelephelyListazas() néven mely a vevők neveit és központi telephelyeinek címeit listázza ki. Az EDMX varázsló a KözpontiTelephelyhez tartozó tulajdonságokat nem tudja automatikusan jól elnevezni. Az egyszerűbb munka érdekében módosítsd az edmx szerkesztőben a KozpontiTelephely tulajdonság nevét KozpontiTelephelyID-re, majd a Telephely1 navigációs tulajdonságot KozpontiTelephelyre.

**Megoldás:**

static void VevoTelephelyListazas()

{

using (var context = new LABOREntities())

{

var query = from v in context.Vevo

orderby v.Nev

select new

{

Nev = v.Nev,

Telep = v.KozpontiTelephely.IR + " " +

v.KozpontiTelephely.Varos + ", " +

v.KozpontiTelephely.Utca

};

foreach (var vevo in query)

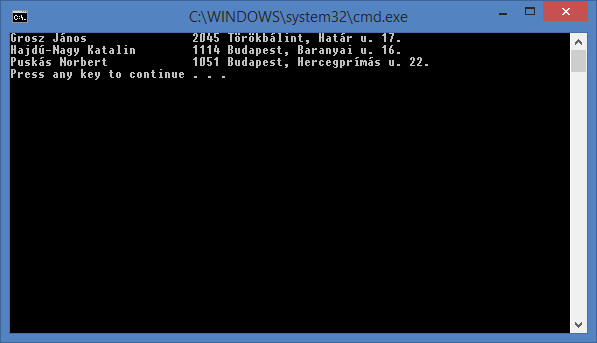
{

Console.WriteLine("{0, -25} {1, 15}", vevo.Nev, vevo.Telep);

}

}

}



**Indoklás:**

Korábbi feladatban is látott anonymus osztályt használok, aminek név és cím tulajdonsága van.

### 12.feladat

**Feladat neve és leírása:**

Készíts egy függvényt VevoMegrendelesek() néven, mely felsorolja a vevőket és minden vevő mellett feltünteti az eddigi, központi telephelyről feladott megrendeléseinek számát! Ügyelj rá, hogy a listázás egyetlen SQL utasítás végrehajtásával történjen.

**Megoldás:**

static void VevoMegrendelesek()

{

using (var context = new LABOREntities())

{

var query = from v in context.Vevo

orderby v.Nev

select new

{

Nev = v.Nev,

Ossz = v.KozpontiTelephely.Megrendeles.Count()

};

foreach (var vevo in query)

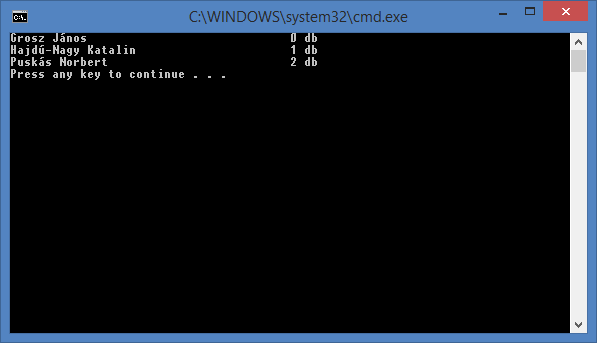
{

Console.WriteLine("{0, -25} {1, 15} db", vevo.Nev, vevo.Ossz);

}

}

}



**Indoklás:**

Az osztályban létrehozok egy attríbutumot, amiben megszámolom a megrendeléseket.

### 14.feladat

**Feladat neve és leírása:**

Készíts egy függvényt Kiszallitas() néven, mely minden megrendelés státuszát 5-re („Kiszállítva”) állítja. A módosítás során a megrendelés tételeinek a státuszát is módosítani kell.

**Megoldás:**

static void Kiszallitas()

{

using (var context = new LABOREntities())

{

var query = from m in context.Megrendeles

where m.StatuszID != 5

select m;

foreach (var rendeles in query)

{

Console.WriteLine("{0}:\t{1}", rendeles.ID, rendeles.StatuszID);

rendeles.StatuszID = 5;

foreach(var tetel in rendeles.MegrendelesTetel)

{

tetel.StatuszID = 5;

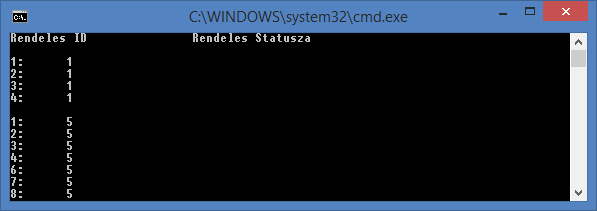
}

}

context.SaveChanges();

}

}



**Indoklás:**

A Kiszállítva státusz ID-ja 5-ös. Lekérdeztem minden olyan rekordot, ami még nincs ksizállítva és átállítom az azonosítót 5-re. Kiírom az elején a megrendelés ID-t és a statusz ID-t, majd miután lefutott a metódus, akkor az összes kiszállított megrendelést is.

### Vélemény

1. **Milyennek értékeled a gyakorlat nehézségét?**
2. **Mennyire érzed hasznosnak a gyakorlat anyagát?**
3. **Milyen javaslataid lennének a gyakorlat felépítésével kapcsolatban?**