Eötvös Loránd Tudományegyetem
Informatikai Kar

Webes alkalmazások fejlesztése

3. fejezet

Objektumrelációs adatkezelés (ASP.NET)

Giachetta Roberto

A jegyzet az ELTE Informatikai Karának 2016. évi jegyzetpályázatának támogatásával készült

Objektumrelációs adatkezelés

Microsoft SQL Server

- A Microsoft rendelkezik saját SQL adatbázis-kezelő megoldással, a *Microsoft SQL Server*rel (*MSSQL*)
  - az *SQL Server Management Studio* az alapvető kliens eszköz, de használható Visual Studio is (*View/Server Explorer*, *Tools/Sql Server*)
  - saját adatkezelő nyelve van (*Transact-SQL*), amely kompatibilis az SQL szabvánnyal
    - tartalmaz pár speciális utasítást/típust is, pl. automatikus sorszámozást az `IDENTITY` utasítással
  - a felhasználó-kezelés támogatja az egyedi fiókokat és Windows authentikációt

Objektumrelációs adatkezelés

Az ADO.NET

- A .NET keretrendszerben az adatbázisokkal kapcsolatos adatelérésért az *ADO.NET* alrendszer biztosítja
  - elődje az *ADO* (*ActiveX Data Objects*)
  - számos lehetőséget ad az adatok kezelésére, az egyszerű SQL utasítások végrehajtásától az összetett objektumrelációs adatmodellekig
  - az egyes adatbázis-kezelőket külön adapterek támogatják, amelyek tetszőlegesen bővíthetőek
  - a közös komponensek a **`System.Data`** névtérben, az adatbázis-függő komponensek külön névterekben helyezkednek el (pl. **`System.Data.SqlClient`**, **`System.Data.OleDb`**)

Objektumrelációs adatkezelés

Adatbázis kapcsolat

- Az adatbázis-kapcsolatot egyben, szöveges formában adjuk meg (*connection string*)
  - általában tartalmazza a szerver helyét, az adatbázis nevét, a kapcsolódó adatait (felhasználónév/jelszó)
  - a pontos tartalom adatbázis-kezelőnként változik
  - pl.:

```
"Server=localhost;Database=myDataBase;
 User Id=myUser;Password=myPassword;"
   // SQL Server standard biztonsággal
"Server=127.0.0.1;Port=5432;Database=myDataBase;
 Integrated Security=true;"
   // PostgreSQL Windows authentikációval
```

Objektumrelációs adatkezelés

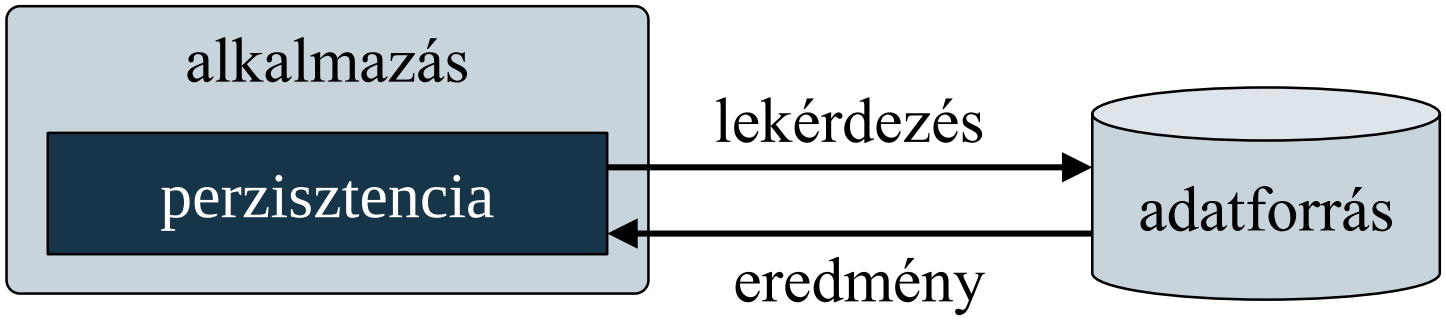
Adatkezelési megoldások

- Az adatbázisok kezelésének több módja adott a .NET keretrendszerben
  - *natív kapcsolat*: direkt SQL utasítások végrehajtása a fizikai adatbázison
  - *logikai relációs modell:* a fizikai adatbázis szerveződésének felépítése és adattárolás a memóriában
  - *egyszerű objektumrelációs modell* (*LINQ to SQL*): az adatbázis-információk leképezése objektumorientált szerkezetre a sémának megfelelően
  - *entitás alapú objektumrelációs modell* (*ADO.NET Entity Framework*): az adatbázis-információk speciális, paraméterezhető leképezése objektumorientált szerkezetre

Objektumrelációs adatkezelés

Natív kapcsolatok

- A *natív (direkt) kapcsolat* lehetővé teszi adatbázis lekérdezések (SQL) végrehajtását a fizikai adatbázison
  - *előnyei*: hatékony erőforrás-felhasználás, közvetlen kommunikáció
  - *hátrányai*: SQL ismerete szükséges, az utasítások a tényleges adatokon futnak (így állandó kapcsolat szükséges az adatbázissal), összetett tevékenységek leírása nehézkes



Objektumrelációs adatkezelés

Natív kapcsolatok

- A kapcsolódást az adatbázishoz az **`SqlConnection`** osztály biztosítja a megfelelő kapcsolati szöveg segítségével, pl.:
  `SqlConnection con = new SqlConnection("…");`
- Az adott kapcsolatban az **`SqlCommand`** osztály segítségével tudunk parancsokat létrehozni
  - a **`CommandText`** tulajdonság tárolja az utasítást
  - a végrehajtás a parancsokra különféleképpen történik
    - az **`ExecuteNonQuery()`** a nem lekérdezés jellegű utasításokat futtatja
    - az **`ExecuteScalar()`** az egy eredményt lekérdező utasításokat futtatja

Objektumrelációs adatkezelés

Natív kapcsolatok

- az **`ExecuteReader()`** az általános lekérdezéseket futtatja, az eredményt egy **`SqlDataReader`** olvasóobjektumba helyezi, amellyel soronként olvasunk

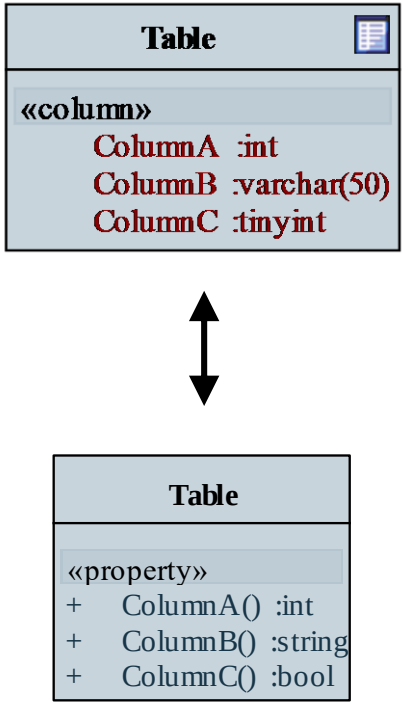
- Pl.:

```
SqlCommand command = con.CreateCommand();
command.CommandText = "select * from MyTable";
SqlDataReader reader = command.ExecuteReader();
while (reader.Read()){
   // amíg tudunk olvasni következő sort
   Console.WriteLine(reader.GetInt32(0) + ", "
      + reader.GetString(1));
   // megfelelően lekérjük az oszlopok tartalmát
};
```

Objektumrelációs adatkezelés

Objektumrelációs adatkezelés

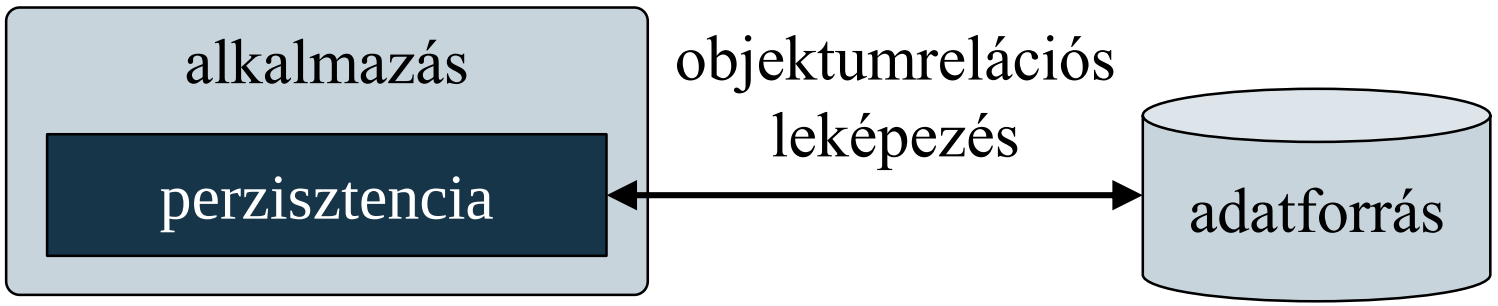
- Az adatkezelő programokat általában objektumorientáltan építjük fel, így célszerű, hogy az adatkezelés is így történjen
- A relációs adatbázisokban
  - az adatokat táblákba csoportosítjuk, amely meghatározza az adatok sémáját, felépítésének módját, azaz *típusát*
  - egy sor tárolja egy adott elem adatait, azaz a sor a típus *példánya*
- Ez a megfeleltetés könnyen átültethető objektumorientált környezetre, a sorok adják az objektumokat, a táblák az osztályokat



Objektumrelációs adatkezelés

Objektumrelációs adatkezelés

- A megfeleltetést *objektumrelációs leképezés*nek (*object-relational mapping*, *ORM*) nevezzük
  - magas szintű transzformációját adja az adatbázisnak, amely a programban könnyen használható
  - ugyanakkor szabályozza az adatok kezelésének módját
  - a létrejött osztályok csak adatokat tárolnak, műveleteket nem végeznek



Objektumrelációs adatkezelés

ADO.NET Entity Framework

- Az *ADO.NET Entity Framework* valósítja meg az adatok összetett, objektumrelációs leképezését
  - alapja az *entitás adatmodell* (*Entity Data Model*, *EDM*), amely leírja az entitások társítását az adatforrás elemeihez
  - általában egy *entitás* egy tábla sorának objektumorientált reprezentációja, de ez tetszőlegesen variálható
  - az entitások között kapcsolatok állíthatóak fel, amely lehet asszociáció, vagy öröklődés
  - támogatja a nyelvbe ágyazott lekérdezéseket (LINQ), a dinamikus adatbetöltést, az aszinkron adatkezelést
  - névtere a **`System.Data.Entity`**

Objektumrelációs adatkezelés

Entitás adatmodellek létrehozása

- A modell létrehozására három megközelítési mód áll rendelkezésünkre:
  - *adatbázis alapján* (*database first*): az adatbázis-szerkezet leképezése az entitás modellre (az adatbázis séma alapján generálódik a modell)
  - *tervezés alapján* (*model first*): a modellt manuálisan építjük fel és állítjuk be a kapcsolatokat (a modell alapján generálható az adatbázis séma)
  - *kód alapján* (*code first*): a modellt kódban hozzuk létre
- A modellben, illetve az adatbázis sémában történt változtatások szinkronizálhatóak, mindkettő könnyen módosítható

Objektumrelációs adatkezelés

Entitás adatmodellek létrehozása

- Pl. (adatbázis):

```
create table Customer( -- tábla létrehozása
    -- tábla oszlopai
    Email VARCHAR(MAX) PRIMARY KEY,
        -- elsődleges kulcs
    Name VARCHAR(50),
    AddressId INTEGER,

    -- idegen kulcs
    CONSTRAINT CustomerToAddress
        FOREIGN KEY (AddressId)
        REFERENCES Address (Id)
);
```

Objektumrelációs adatkezelés

Entitás adatmodellek létrehozása

- Pl. (kód):

```
class Customer // entitástípus létrehozása
{
    [Key] // elsődleges kulcs
    public String Email { get; set; }

    [StringLength(50)] // megszorítás
    public String Name { get; set; }

    [ForeignKey("AddressId")] // idegen kulcs
    public Address Address { get; set; }

    public ICollection<Order> Orders { get; set; }
}
```

Objektumrelációs adatkezelés

Entitás adatmodellek használata

- Az entitásokat egy adatbázis modell (**`DbContext`**) felügyeli, amelyben eltároljuk az adatbázis táblákat (**`DbSet`**)
  - egy aszinkron modellt biztosít, a változtatások csak külön hívásra (**`SaveChanges`**) mentődnek az adatbázisba
  - pl.:

```
public class SalesContext : DbContext {
    // kezelő létrehozása
    public DbSet<Customer> Customers {
        get; set;
    }
        // adatbázisbeli tábla
    …
}
```

Objektumrelációs adatkezelés

Entitás adatmodellek használata

- Az adattábla (**`DbSet`**) biztosítja lekérdezések futtatását, adatok kezelését
  - létrehozás (**`Create`**), hozzáadás (**`Add`**, **`Attach`**), keresés (**`Find`**), módosítás, törlés (**`Remove`**)
  - az adatokat és a lekérdezéseket lusta módon kezeli
    - az adatok csak lekérdezés hatására töltődnek a memóriába, de betölthetjük őket előre (**`Load`**)
    - a LINQ lekérdezések átalakulnak SQL utasítássá, és közvetlenül az adatbázison futnak
  - egy tábla nem tárolja a csatolt adatokat, azok betöltése (**`Include`**)

Objektumrelációs adatkezelés

Entitás adatmodellek használata

- Pl.:

```
SalesContext db = new SalesContext();
IEnumreable<Customer> customer =
    Db.Customers.FirstOrDefault(cust =>
        cust.Email == "groberto@inf.elte.hu");
    // LINQ lekérdezés
if (customer == null)
{
    customer = new Customer { Name = "Roberto",
                    Email = "groberto@inf.elte.hu" };
    db.Customers.Add(customer);
        // entitás létrehozása és felvétele
    db.SaveChanges(); // változások elmentése
}
```

Objektumrelációs adatkezelés

Entitás adatmodellek használata

- Pl.:

```
IQuery<Customer> query = db.Customers
                        .Include(cust => cust.Address);
   // a megadott tulajdonságok (csatolt adatok)
   // is betöltésre kerülnek, hasonlóan
   // táblanévvel: .Include("Address")
Boolean anyBudapest = query
    .Any(cust => cust.Address.City == "Budapest");
   // a lekérdezés az adatbázisban fut

query.Load(); // adatok betöltése
anyBudapest = query
    .Any(cust => cust.Address.City == "Budapest");
   // a lekérdezés a memóriában fut
```

Objektumrelációs adatkezelés

Példa

Feladat: Valósítsuk meg egy utazási ügynökség weblapját, amelyben apartmanok között böngészhetünk.

- a főoldalon (`Index`) az épületek alapvető adatai listázódnak, amit szűrhetünk, a részletek oldalon (`Details`) egy épület apartmanjai listázódnak
- az oldalt egy vezérlő (`HomeController`) irányítja, amely három akciót definiál: minden listázása (`Index`), egy város épületeinek listázása (`List`), egy épület részleteinek lekérése (`Details`)
- a városok listázásához felhasználjuk a `ViewBag` tulajdonságot
- az adatokat adatbázisban (`TravelAgency`) tároljuk

Objektumrelációs adatkezelés

Példa

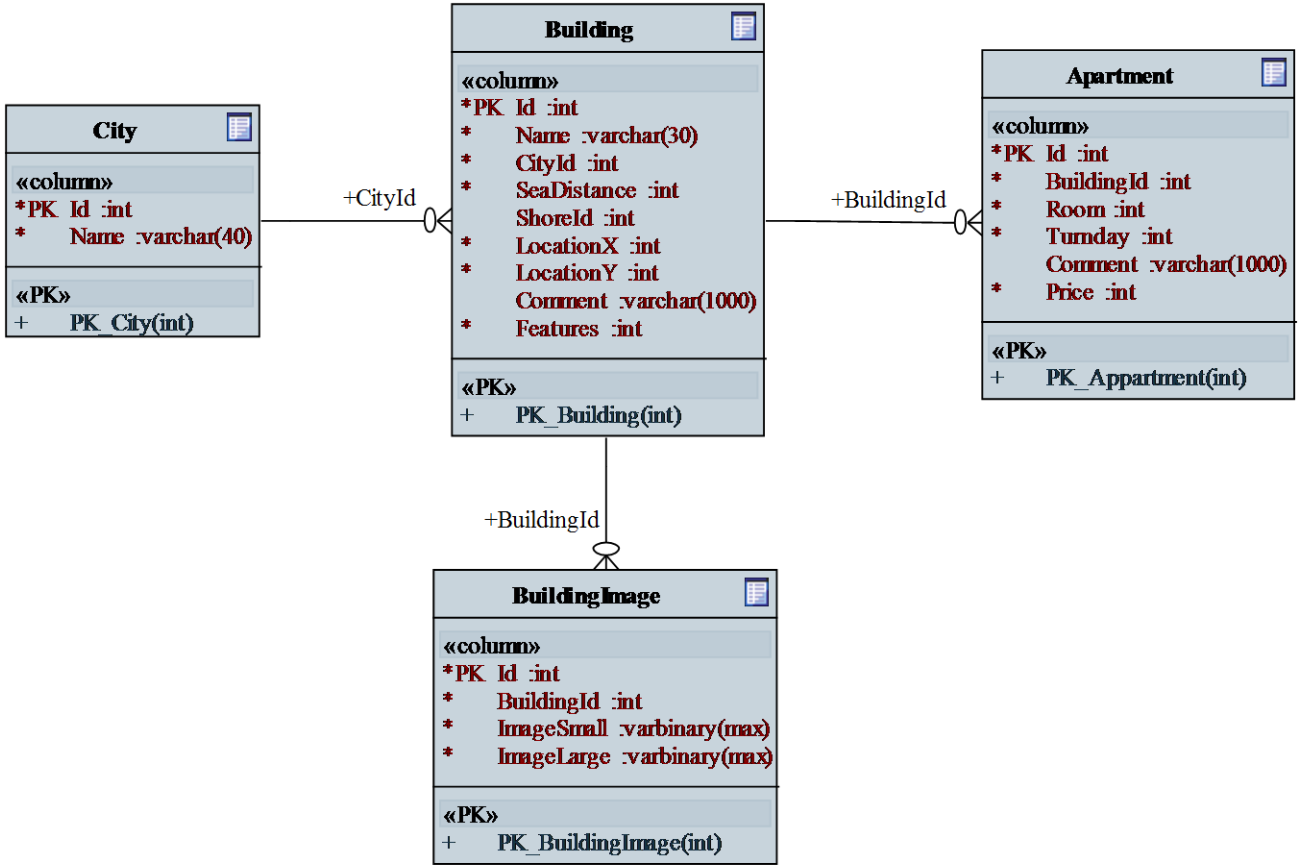
Tervezés (adatbázis):

- a **`City`** tábla tárolja a városok adatait tartalmazza
- a **`Building`** tábla az épületek adatait tartalmazza, benne a város azonosítójával
- az **`Apartment`** tábla az apartman adatokat tárolja, benne az épület azonosítójával
- a **`BuildingImage`** tábla tárolja az épületek bemutató képeit, minden képből egy nagyobb, és egy kisebb változatot, valamint az épület azonosítóját
- az elsődleges kulcsokat automatikusan generáljuk
- az adatbázist entitásmodell segítségével töltjük be az alkalmazásban

Objektumrelációs adatkezelés

Példa

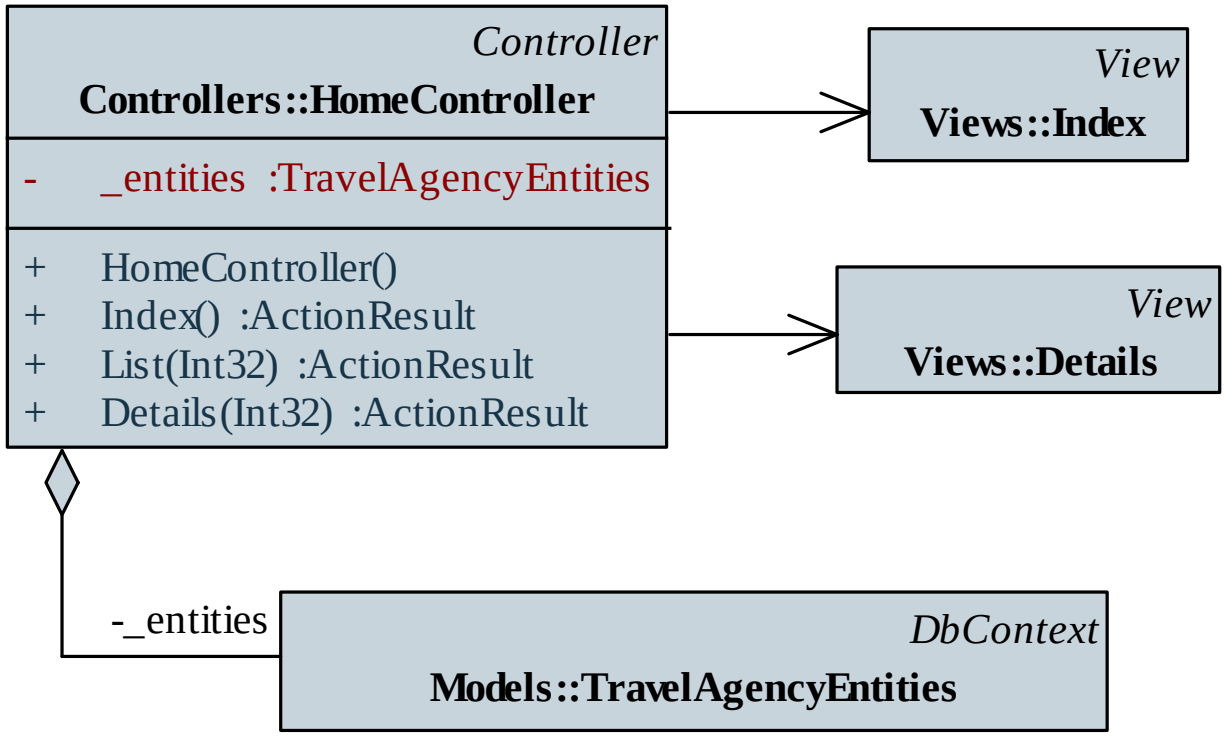
Tervezés (adatbázis):



Objektumrelációs adatkezelés

Példa

Tervezés (alkalmazás):



Objektumrelációs adatkezelés

Példa

Megvalósítás (`HomeController.cs`*):*

```
…
public ActionResult List(Int32 cityId) {
    // ha hibás az azonosító
    if (!_entities.City.Any(c => c.Id == cityId))
        return HttpNotFound();
        // átirányítjuk a nem talált oldalra

    // megkeressük a megfelelő város szonosítókat
    return View("Index", _entities.Building
                    .Include("City")
                    .Where(b => b.CityId == cityId));
}
…
```

Objektumrelációs adatkezelés

Példa

Megvalósítás (`Index.cshtml`):

```
…
<ul>
    @* felsoroljuk a városokat *@
    @foreach (City city in ViewBag.Cities) {
        <li>
        @* létrehozunk egy linket minden városra *@
            @Html.ActionLink(city.Name, "List",
                        new { cityId = city.Id })
        </li>
    }
</ul>
…
```