Alkalmazásfejlesztés-II

Adatbáziskezelés ADO.NET technolóiával

University of Szeged

2022. október 26.– 11:30

Az *ADO.NET*-et adatforrások kezelésére használhatjuk (adatok elérésére és manipulálására) online és offline módon is. Egy réteget képez, amely segítségével közel azonos módon kezelhetünk erőforrásokat pl. Sqlite vagy MS SQL adatbázist.

Felépítését tekintve az alapvető metódusai, adattagjai interfészekként definiáltak, ezeket *providerek* implementálják, melyekkel dolgozunk a gyakorlatban. A providerek esetén elnevezési konvenció a megfelelő prefixelés pl. Sqlite esetén minden használt osztály a SqliteXYZASD néven fut. Ezek megvalósítják az IDisposable interfészt - használhatók a `using` kulcsszóval, - minden esetben valamilyen kivételkezelés javallott a használatukkor, nem menedzselt kód lévén ezt a szemétgyűjtő nem fogja eltakarítani utánunk.

Objektumok

- **Adatbázis kapcsolat**
  - `DbConnection`: Rendelkezik `Open` és `Close` metódusokkal, Java-val ellentétben az adatbáziskapcsolat létrehozásakor nem nyílik meg a kapcsolat automatikusan, így azt minden esetben manuálisan az `Open()` segítségével kell megnyitnunk használat előtt,
  - Vár egy `connectionString`-et, ami tartalmazza az adatbázis elérési útvonalát és egyéb információkat (pl. felhasználónév, jelszó),
  - A `CreateCommand(...)` segítségével tudunk SQL lekérdezést létrehozni
- **DbCommand** - lekérdezések
  - A `CommandText` tulajdonságával tudunk egy lekérdezést megadni. Adatlekérdezéseket és módosításokat is a `CommandText` segítségével lehet elvégezni,
  - Amennyiben paraméterekre van szükség (pl. `where ...`), ezt a `CommandText`-től külön, az objektum nevesített paraméterlistájában – `Parameters` – lehet megtenni,
  - Minden nevesített paraméter használatakor, a paraméterek megadása `@xyzasd` formában lehetséges,
  - Hasonló konstrukció, mint az Alkalmazásfejlesztés-I-ből ismert `PreparedStatement`,
  - Egy `DbCommand` az `ExecuteNonQuery()`, az `ExecuteReader()`, és az `ExecuteScalar()` metódusok segítségével futtatható le,

    * `ExecuteNonQuery` - érintett sorok számát adja vissza
    * `ExecuteReader` - readert ad vissza - SELECT esetén használandó
    * `ExecuteScalar` - első találat első celláját
- **DataReader**
  - Tároló objektum, a `SELECT` során a lekért adatok tárolására (az `ExecuteReader` visszatérési értéke),
  - `Close`, `Read`, `NextResult` metódusok használhatók,
  - A `Read` segítségével lehet rajta végigiterálni,
  - Az egyes értékek `GetType`, `GetOrdinal` metódusokkal érhetők el

Heroes alkalmazás kiegészítése

Bevezetés

Néhány alap lépés az adatbázisműveletek megírása előtt:

- Sqlite ismeretek felelevenítése - Alkalmazásfejlesztés I. anyag
- Megfelelő adatbázis fájl elkészítése (.db kiterjesztésű fájl), datbázistáblák létrehozása ld. Sqlite how to leírás
- A megfelelő névterek használata az Sqlite providerek elérésére
  - `System.Data.Sqlite.Core` NuGet package telepítése szükséges (ezt felajánlja a VS, az utolsó stabil verziót telepítsük).

Adatbázis létrehozása

Kezdjük egy adatbázisfájl elkészítésével. Készítsünk a projekt mappában egy `DB` nevű almappát. Az adatbázis létrehozásához az sqlite3 shell-t használjuk. Először szükség lesz egy scriptre, ami a tábla létrehozásához szükséges SQL-t tartalmazza:

heroes.sql fájl:

```
CREATE TABLE Heroes (
  ID integer NOT NULL PRIMARY KEY AUTOINCREMENT,
  Name text NOT NULL,
  HeroName text NOT NULL,
  Power text NOT NULL,
  Age integer NOT NULL,
  UNIQUE(HeroName)
);
```

Nyissunk egy terminált/command propt-ot a DB mappában, és hozzuk létre az sqlite3 segítségével az adatbázis fájlt:

1. `sqlite3 heroes.db`: ha még nincs ilyen fájl, az sqlite3 létrehozza, ha pedig már van, akkor megnyitja (Linuxon: `./sqlite3 heroes.db`, amennyiben nincs jog a futtatásra, akkor `chmod +x sqlite3`),
2. `.read heroes.sql`: beolvassa és lefuttatja a scriptet,

3. `.tables`: ellenőrzés, kilistázza az adatbázis táblákat, ha visszakapjuk a Heroes-t, akkor kész is az adatbázis,
4. +1 ellenőrzés: nézzük meg, hogy létrejött-e a parancsok hatására a db fájl a `DB` mappában, és hogy nagyobb-e a mérete 0KB-nál.

Megfelelő NuGet Package telepítése

A *Solution Explorerben* jobb klikk a projektre, majd `Manage Nuget Packages...` menüpont, a megnyíló Nuget Package Manager ablakban válasszuk ki a `Browse` tabot. Keressük meg a `Microsoft.Data.Sqlite` csomagot és telepítsük a legfrissebb stabil verziót.

DAO

Készítsünk egy új osztályt a DAO mappába HeroesAdoDao néven.

```
namespace Heroes_Core.DAO {
  class HeroesAdoDao { /*...*/ }
}
```

Az osztálynak meg kell valósítani a korábban létrehozott IHeroesDao interfészt. Az IDE generálja a metódusok fejlécét, csak a hasznos kód írásával kell foglalkozni:

```
namespace Heroes_Core.DAO {
  class HeroesAdoDao : IHeroesDao
  {
    public bool AddHero(Hero hero) {
        throw new NotImplementedException();
    }
    /* ... */
  }
}
```

Adjuk meg az adatbázis elérési útvonalát egy connection stringben:

```
private static readonly string conn_string =
    @"Data Source=../../../DB/heroes.db;";
```

Windowson, Visual Studioban az elérési útvonalat a `./bin/Debug/net5/` mappától visszaszámolva kell megadni, ezt igazítsátok a saját igényeitekhez.

AddHero

A hős hozzáadás logikája teljes mértékben hasonlítani fog a már Java-ban látott logikához. `using` blokkban dolgozunk, így az erőforrás `Dispose` metódusa automatikusan le fog futni, amint a blokkból kilép a futás. Az adatbáziskapcsolathoz kell az adatbázis elérési útvonalát tartalmazó connection string.

Mielőtt elkezdünk az adatbázistól kérni adatot, az `Open` metódussal meg kell nyitni a kapcsolatot. Paraméterezett commandra/statementre lesz szükségünk a hozzáadáshoz. A paraméterekhez értéket rendelünk, majd futtatjuk a parancsot. Update/insert commandoknál az `ExecuteNonQuery` metódust használjuk, mely az érintett sorok számával tér vissza.

```
public bool AddHero(Hero hero)
{
    using SqliteConnection conn = new SqliteConnection(conn_string);
    conn.Open(); //kapcsolat nyitása

    //paraméterezett command
    SqliteCommand command = conn.CreateCommand();
    command.CommandText = "INSERT INTO Heroes " +
        "(Name, HeroName, Power, Age) VALUES " +
        "(@name, @hname, @power, @age)";

    //paraméterek megadása
    command.Parameters.Add("name", System.Data.DbType.String).Value = hero.Name;
    command.Parameters.Add("hname", System.Data.DbType.String).Value =
        ↪ hero.HeroName;
    command.Parameters.Add("power", System.Data.DbType.String).Value = hero.Power;
    command.Parameters.Add("age", System.Data.DbType.Int32).Value = hero.Age;

    if (command.ExecuteNonQuery() != 1)
        return false;

    return true;
}
```

GetHeroes

A futtatáshoz az `ExecuteReader`-t használjuk, ami egy readert ad vissza. A reader értékein kell végigiterálni, és az egyes sor értékeket egy lokális listához adjuk hozzá.

```
public IEnumerable<Hero> GetHeroes()
{
    List<Hero> heroes = new List<Hero>();
    using SqliteConnection conn = new SqliteConnection(conn_string);
    conn.Open();
    SqliteCommand command = conn.CreateCommand();
    command.CommandText = "SELECT * FROM Heroes";

    using SqliteDataReader reader = command.ExecuteReader();
    while (reader.Read())
    {
        heroes.Add(
            new Hero
            {
                ID = reader.GetInt32(reader.GetOrdinal("ID")),
                Name = reader.GetString(reader.GetOrdinal("Name")),
                HeroName = reader.GetString(reader.GetOrdinal("HeroName")),
                Power = reader.GetString(reader.GetOrdinal("Power")),
                Age = reader.GetInt32(reader.GetOrdinal("Age"))
            }
        );
    }
    return heroes;
}
```

ModifyHero

A szerkesztésnél a hozzáadáshoz hasonló módon járunk el.

```
public bool ModifyHero(Hero hero)
{
    using (SqliteConnection conn = new SqliteConnection(conn_string);
    conn.Open();
    SqliteCommand command = conn.CreateCommand();

    command.CommandText = "UPDATE Heroes SET "+
        "Name=@name, HeroName=@heroName, Power=@power, Age=@age "+
        "WHERE ID=@id";

    command.Parameters
        .Add("name", System.Data.DbType.String).Value = hero.Name;
    command.Parameters
        .Add("heroName", System.Data.DbType.String).Value = hero.HeroName;
    command.Parameters
        .Add("power", System.Data.DbType.String).Value = hero.Power;
    command.Parameters
        .Add("age", System.Data.DbType.Int32).Value = hero.Age;
    command.Parameters
        .Add("id", System.Data.DbType.Int32).Value = hero.ID;

    if (command.ExecuteNonQuery() != 1)
        return false;

    return true;
}
```

A DAO réteg kicserélése az alkalmazásban

Ahhoz hogy az új DAO réteg legyen felhasználva a memóriás megvalósítást ki kell cseréljük az újra a Controller példányosításánál.

Form1: (főablak)

```
controller = new HeroesController(new HeroesAdoDao());
```