Haladó Fejlesztési Technikák

Gyakorlati Zárthelyi Dolgozat – 2023.05.31

Az elkészült teljes anyagot ZIP formátumban töltse fel az oktató által megadott helyre.

Előkészületek

Az első 5 percben készítsen elő egy BookstoreApp nevű konzol alkalmazást, és egy BookstoreApp.Test nevű DLL alkalmazást!

- A BookstoreApp alkalmazáshoz telepítse az alábbi NUGET csomagokat:
  `Microsoft.EntityFrameworkCore`
  `Microsoft.EntityFrameworkCore.InMemory`
  `Microsoft.EntityFrameworkCore.Proxies`
- A BookstoreApp.Test alkalmazáshoz telepítse az alábbi NUGET comagokat
  `NUnit`
  `NUnit3TestAdapter`
  `Microsoft.NET.Test.Sdk`

Osztályok elkészítése (*ez a feladat pontot még nem ér*)

Készítse el az alábbi 4 osztályt a felsorolt adattagokkal:

`Author`:

- *Id*: `int` (automatikusan növelt kulcs)
- *Name*: `string`
- *Books*: `Book ICollection` (virtuális adattag, amely nem szerepel az adatbázisban!)

`Book`:

- *Id*: `int` (automatikusan növelt)
- *Title*: `string`
- *Year:* `int`
- *AuthorId*: `int`, idegen kulcs
- *Author*: `Author` (virtuális adattag, amely nem szerepel az adatbázisban)
- *BooksAndBookstores*: `BooksAndBookstores ICollection` (virtuális adattag, amely nem szerepel az adatbázisban!)

`Bookstore`***:***

- *Id*: `int` (automatikusan növelt)
- *Name*: `string`
- *BooksAndBookstores*: `BooksAndBookstores ICollection` (virtuális adattag, amely nem szerepel az adatbázisban!)

`BookAndBookstore` (kapcsolótábla a több-a-többhöz kapcsolathoz)***:***

- *Id*: `int` (automatikusan növelt)
- *BookId*: `int`, idegen kulcs
- *Book*: `Book` (virtuális adattag, amely nem szerepel az adatbázisban!)
- *BookstoreId*: `int`, idegen kulcs
- *Bookstore*: `Bookstore` (virtuális adattag, amely nem szerepel az adatbázisban!)

In-Memory adatbázis elkészítése (14 pont)

Készítsen el egy in-memory database-t a data.txt-ben található seed adatok alapján egy `DbContext` nevű osztály segítségével.

Fontos megjegyzések:

- Összesen 3 darab egy-a-többhöz kapcsolatot kell létrehozni:
  - `Author` és `Book` között egy-a-többhöz kapcsolat áll fent, ezért be kell állítani, hogy egy könyvnek egy szerzője van, melyhez több könyv is tartozhat (a legvégén pedig az idegen kulcsról se feledkezzünk meg!)
  - `Book` és `Bookstore` osztályok között úgy lehet több-a-többhöz kapcsolatot definiálni, hogy a `BookAndBookstore` táblának két idegen kulcsa is van. Ezért a *modelBuilder.Entity<BookAndBookstore>()* esetében a `Book` és `Bookstore` tekintetében is rendelkezni kell egy-a-többhöz kapcsolatról (1 könyv, több BookAndBookstore, valamint 1 könyvesbolt, több BookAndBookstore). Ez két további modelBuilder.Entity() hívás, melyek ugyanúgy épülnek fel, mint az `Author` és `Book` közötti egy-a-többhöz kapcsolat.
- Továbbá, mivel az összes lekérdezés csak a `BookAndBookstore` osztályon keresztül fog történni, ezért elegendő csak ehhez az egy osztályhoz tartozó `DbSet`-et létrehozni.

LINQ lekérdezések (12 pont)

A *BookAndBookstore* `DbSet` alapján a *Main(string[] args)* függvényben írjon LINQ lekérdezéseket az alábbi információk kigyűjtésére:

a. Az összes szerző neve, úgy, hogy ismétlődés ne legyen benne (fontos: mivel nem minden szerzőnek kapható a boltokban könyve a seed adatok szerint, ezért lesz olyan szerző az Author táblából, akinek itt nem szerepel a neve) (2 pont)
b. Az összes olyan könyv címe, melyet HG Wells írt (és valamelyik könyesboltban kapható) (2 pont)
c. Az összes olyan könyv címe, amely nemcsak 1, hanem legalább 2 boltban kapható (4 pont)
d. Az összes olyan könyvesbolt neve, ahol több, mint 2-féle könyv kapható (4 pont)

Attribútum és validáció reflexióval (8 pont)

Hozzon létre egy attribútumot, és valósítsa meg az attribútum alapján történő validációt:

a. Hozzon létre egy argumentum nélküli `YearAttribute` attribútumot – amellyel az `Book` osztályban levő *Year* adattagot ellátva biztosítható, hogy egy könyv megjelenési éve nem lehet nagyobb, mint 2023, és nem lehet kisebb, mint 1000 (valid példa: 2002, invalid példák: 999, -10, 2050) (2 pont)
b. A validáláshoz hozzon létre egy `IValidation` interfészt, melynek *public bool Validate(object instance, PropertyInfo prop)* metódusa nincs kifejtve: ez alkalmas lesz arra, hogy az adott instance-nek lekérdezzük a prop által meghatározott tulajdonságát, és igaz/hamis visszatéréssel megmondjuk, hogy teljesül-e rá az adott feltétel (0 pont)
c. Ezt az interfészt implementálja a `YearValidation(YearAttribute)` osztály, mely konstruktora a kapott *YearAttribute* objektumot eltárolja, és megvalósítja a *Validate* metódust (2 pont)

d. A `Validator` osztálynak a *public bool Validate(object instance)* metódusa végigmegy az instance összes tulajdonságán és lekéri mindegyik attribútumait b.) végigmegy ezeken az attribútumokon, és ha bármelyik kasztolható *YearAttribute* típusra, akkor meghívja annak *Validate* metódusát az adott instance adott property-jére c.) ha ez bármelyik esetben hamis értékkel tér vissza, akkor hamis értéket; ellenkező esetben a ciklusok végén igaz értéket ad vissza. (3 pont)
e. Miután a `Book` osztály *Year* adattagjára applikálta az attribútumot, a *Main()* függvényben hozzon létre 3 *Book* típusú objektumot, melyek közül 1 valid és 2 invalid. A `Validator` osztály példányával validálja ezeket, és írja ki az eredményt a konzolra! (1 pont)

Unit tesztek (6 pont)

Készítsen egy Test DLL-t, melyben az előző feladat e.) pontjában készített példákat unit test keretében validálja (3 teszteset).