

A MyDbContext az EF Core “kapuja” az adatbázishoz: ezen keresztül érdeklőd a táblákat (DbSet), és ez kezeli az **egységnyi munkát** (unit of work): objektumok követése (tracking) → beszúrás/módosítás/törlés → SaveChanges()-sel véglegesítés.

Mit reprezentál a MyDbContext?

- A MyDbContext : DbContext egy “session” az adatbázissal: tartja a konfigurációt, a kapcsolatot, és a változáskövetést.
- A public virtual DbSet<user> users { get; set; } olyan, mintha a user tábla/entitás-gyűjteményed lenne, amin LINQ-olhatsz és amibe új entitást adhatsz.
- Az OnConfiguring(...) => optionsBuilder.UseMySQL(...) mondja meg, hogy melyik adatbázishoz és milyen providerrel (Pomelo MySQL/MariaDB) csatlakozzon a context, ha nincs máshonnan (pl. DI-ből) konfigurálva.

Mi történik a CreateUser-ben lépésről lépésre?

1. `using (var context = new MyUserContext())`
Létrehozol egy DbContext példányt egy rövid élettartamra, majd a `using` a végén automatikusan `Dispose()`-olja (felszabadítja az erőforrásokat).
2. `var newUser = new user { ... }`
Ez még csak egy sima C# objektum, ekkor még nem “adatbázis rekord”.
3. `context.users.Add(newUser);`
Ezzel az objektumot “trakkelteted” a contexttel, és az állapotát `Added`-re teszi (azaz beszúrandó).
4. `context.SaveChanges();`
Itt történik a lényeg: az EF Core végignézi a trackelt változásokat, és lefuttatja az SQL INSERT-et az adatbázisban.
5. `currentId = newUser.id;`
Az `id` azért lesz kitöltve `SaveChanges()` után, mert az adatbázis generálja (`AUTO_INCREMENT`), és az EF Core a mentés után visszaírja az értéket az entitás példányba.

“Mit hívok” pontosan a MyUserContextből?

- `new MyUserContext()` → példányosítod a contextet (konzol appban ez gyakori, web appban inkább DI-val adják).
- `context.users` → a user entitások készlete (tábla leképezése).
- `Add(...)` → jelez a contextnek, hogy ezt be kell szúrni.
- `SaveChanges()` → tranzakció-szerűen végrehajtja a felgyűlt módosításokat a DB-ben.

select esetén az EF Core a háttérben végigköveti a „lekérdezési folyamatot”, de a valódi SQL lekérdezés csak akkor fut le, amikor materializálod az eredményt (pl. ToList()-al). Ez a lazys (inkább: deferred) execution EF Core jellegzetessége C# LINQ kifejezésekkel.

Ami a context.users.ToList() sorban történik:

```
using (var context = new MyUserContext())  
{  
    List<user> users = context.users.ToList();  
    // ...  
}
```

- context.users egy DbSet<user>, ami valójában egy IQueryable<user> interfészt implementál. Ez még csak „leírja” a lekérdezést, nem futtatja le.
- A ToList() meghívásakor az EF Core veszi ezt az IQueryable-t, lefordítja SQL SELECT lekérdezésre (SELECT id, user_name, first_name, ... FROM user), elküldi a MySQL/MariaDB szervernek, egész táblát beolvassa, majd visszaadják az adatokat C# objektumokként (user példányok listáját).
- A users lista most már teljesen az applikáció memóriájában van, és a using végén a context el is kezd felszabadulni (nem „él” tovább).