PEMROGRAMAN WEB LANJUTAN

ALOYSIUS ARI WICAKSONO

Topik

- EF Core
- Code First Approach
- Database First Approach
- Query, Saving Data

Pre-Test

- Apa fungsi Model dalam arsitektur MVC?
- Apa perbedaan code first vs database first pada entity framework?

EF Core

- Entity Framework (EF) Core is a lightweight, extensible, open source and cross-platform version of the popular Entity Framework data access technology.
- EF Core can serve as an object-relational mapper (O/RM), which:
  - Enables .NET developers to work with a database using .NET objects.
  - Eliminates the need for most of the data-access code that typically needs to be written.

Model

- With EF Core, data access is performed using a model. A model is made up of entity classes and a context object that represents a session with the database. The context object allows querying and saving data.

Code First Approach

- In code first approach we will first create entity classes with properties defined in it. Entity framework will create the database and tables based on the entity classes defined. So database is generated from the code. When the dot net code is run database will get created.

Code First Approach

Advantage

- You can create the database and tables from your business objects.
- You can specify which related collections are to be eager loaded, or not be serialized at all.
- Database version control.
- Good for small applications.

Disadvantage

- You have to write everything related to database in the visual studio code.
- For stored procedures you have to map stored procedure using Fluent API and write Stored Procedure inside the code.
- If you want to change anything in the database tables you to make changes in the entity classes in the code file and run the update-database from the package manager console.

Database First Approach

- Database and tables are created first. Then you create entity Data Model using the created database.
- Simple to create the data model
- Mapping and creation of keys and relationships are easy as you need not have to write any code .
- Preferred for large applications

Reverse Engineering (Scaffolding)

- Reverse engineering is the process of scaffolding entity type classes and a DbContext class based on a database schema. It can be performed using the Scaffold-DbContext command of the EF Core Package Manager Console (PMC) tools or the dotnet ef dbcontext scaffold command of the .NET Command-line Interface (CLI) tools.

Query

```
using (var db = new BloggingContext())
{
    var blogs = db.Blogs
        .Where(b => b.Rating > 3)
        .OrderBy(b => b.Url)
        .ToList();
}
```

Saving Data

```
using (var db = new BloggingContext())
{
    var blog = new Blog { Url = "http://sample.com" };
    db.Blogs.Add(blog);
    db.SaveChanges();
}
```

- .Add lalu SaveChanges akan menjalankan syntax INSERT di database

CRUD

```
using var db = new BloggingContext();

// Note: This sample requires the database to be created before running.
Console.WriteLine($"Database path: {db.DbPath}.");

// Create
Console.WriteLine("Inserting a new blog");
db.Add(new Blog { Url = "http://blogs.msdn.com/adonet" });
db.SaveChanges();

// Read
Console.WriteLine("Querying for a blog");
var blog = db.Blogs
    .OrderBy(b => b.BlogId)
    .First();

// Update
Console.WriteLine("Updating the blog and adding a post");
blog.Url = "https://devblogs.microsoft.com/dotnet";
blog.Posts.Add(
    new Post { Title = "Hello World", Content = "I wrote an app using EF Core!
db.SaveChanges();

// Delete
Console.WriteLine("Delete the blog");
db.Remove(blog);
db.SaveChanges();
```

Commit & Rollback

- Jika ada error, akan kerollback

```
using (var transaction = context.Database.BeginTransaction())
{
    try
    {
        Mahasiswa mahasiswa2 = new Mahasiswa();
        mahasiswa2.NIM = "123";
        mahasiswa2.Nama = "Nadia";
        context.Mahasiswa.Add(mahasiswa2);
        context.SaveChanges();

        Mahasiswa mahasiswa3 = new Mahasiswa();
        mahasiswa3.NIM = "123";
        mahasiswa3.Nama = "Dinda";
        context.Mahasiswa.Add(mahasiswa3);
        context.SaveChanges();

        transaction.Commit();
    }
    catch (Exception ex)
    {
        transaction.Rollback();
        //result = false;

    }
}
```

Pemrograman Web Lanjutan

SEKIAN DAN TERIMA KASIH