STEPS FOR CONNECTING FROM ASP.NET CORE MVC APP TO DATABASE AND FOR MIGRATIONS WITH CODE FIRST MODEL IN VS CODE

1. Create a class that inherits from dbcontext class. (dbcontext class is available in EntityFrameworkCore namespace). Read more about dbcontext from here:

<https://docs.microsoft.com/en-us/dotnet/api/system.data.entity.dbcontext?view=entity-framework-6.2.0>

This class will also list the dbset entities.

1. Create your connectin string in appsettings.json. During development phase connection string can go in appsettings.json file. For your connection string , the one to use in hp lab, will mostly be:

“Server=localhost\SqlExpress;Database=<yourdbname>;Trusted\_Connection=true;MultipleActiveResultSets=True”

We use the sqlexpress database in our lab exercise.

1. For dependency injection we will configure the database connection information in program.cs to get the connection object on our controller files directly via Dependency Injector class.

builder.Services.AddDbContext<ApplicationDbContext> (options => options.UseSqlServer (builder.Configuration.GetConnectionString(connstringname)))

1. You will need to install the following packages
2. Microsoft.EntityFrameworkCore
3. Microsoft.EntityFrameworkCore.SqlServer
4. Microsoft.EntityFrameworkCore.Tools.Dotnet
5. Microsoft.EntityFrameworkCore.Design

In real world projects, data models change as features get implemented: new entities or properties are added and removed, and database schemas need to be changed accordingly to be kept in sync with the application. The migrations feature in EF Core provides a way to incrementally update the database schema to keep it in sync with the application's data model while preserving existing data in the database.

At a high level, migrations function in the following way:

* When a data model change is introduced, the developer uses EF Core tools to add a corresponding migration describing the updates necessary to keep the database schema in sync. EF Core compares the current model against a snapshot of the old model to determine the differences, and generates migration source files; the files can be tracked in your project's source control like any other source file.
* Once a new migration has been generated, it can be applied to a database in various ways. EF Core records all applied migrations in a special history table, allowing it to know which migrations have been applied and which haven't.

To manage migrations you must first install the EF Core command line tools

1. Dotnet tool install –global dotnet-ef, to install the ef tool for cli in .net
2. Dotnet ef migrations add InitialCreate
3. Dotnet ef database update

the above updates your database to the latest migration.

Read more details about migrations from here:

<https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli>

<https://docs.microsoft.com/en-us/ef/core/managingschemas/migrations/managing?tabs=dotnet-core-cli>

https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/applying?tabs=dotnet-core-cli