This cheat sheet is for the course [Learn C# Full Stack Development with Angular and ASP.NET] by Jannick Leismann.

ENTITY FRAMEWORK CORE

Entity Framework

Is an object-relational mapper (ORM) for .NET applications supported by Microsoft. It allows developers to work with a database using .NET objects, removing the need for most of the data-access code that developers usually need to write.

Entity Framework Core

A contemporary, lightweight, and expandable version of the well-known Entity Framework data access technology . It supports a large variety of relational and non-relational database sources and is made to function on several operating systems, including Windows, Linux, and macOS.

Features

Object-Relational Mapping

EF associates properties of.NET classes with database columns and maps.NET classes to database tables.

LINQ Queries

Language Integrated Query (LINQ) allows you to design strongly-typed queries that EF converts into SQL for the underlying database.

Change Monitoring

EF records any modifications performed to objects, including adds, updates, and deletions, and permits these modifications to be stored in the database.

Code-First Approach

This method lets you use C# classes to design your model, while EF builds the database around these classes.

Database-First Approach

Based on an already-existing database schema, this approach enables you to create C# classes.

Support for Migration

EF offers a way to handle modifications to database schemas as an application progresses.

Steps

1. Define your data model using C# classes.

```
public class Employee

{

    public int EmployeeId { get; set; }

    public string FirstName { get; set; }

    public string LastName { get; set; }

    public string Position { get; set; }

}
```

2. Create a DbContext class that manages the model and handles database connections.

```
public class EmployeeContext : DbContext

{

    public DbSet<Employee> Employees { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)

{


optionsBuilder.UseSqlServer(@"Server=[YourServer];Database=[YourData
base];Trusted_Connection=True;");

    }

}
```

3. Use LINQ to query and manipulate data through the DbContext.

```
using (var context = new EmployeeContext())

{

    var employee = new Employee

    {

        FirstName = "John",

        LastName = "Doe",

        Position = "Manager"

    };

    context.Employees.Add(employee);

    context.SaveChanges();

}
```

4. Apply migrations to update the database schema.

```
dotnet ef migrations add InitialCreate

dotnet ef database update
```