EF Core 8.0 ORM
Code First Programming For RDBMS

SCAN ME

발표자료실 : https://bit.ly/40uiHjR

GitHub 샘플소스: https://github.com/eddykang1074/EFCoreForRDBMS

** 개발가이드 : GitHub 샘플소스 Docs폴더내 Guide.txt 파일 참조 **

Contents

01

EF Core & RDBMS & ORM

1. EF Core 소개
2. EF Core 주요 구성요소
3. Database Provider based EF Core
4. Global RDBMS Trends
5. ORM 이해하기

02

Code First ORM Process

1. EF Core 공통 패키지 설치
2. MS SQL 지원 패키지 설치
3. 모델 및 dbContext 만들기
4. db연결문자열 구성
5. dbContext DI 구성
6. dotnet-ef 명령어기반 테이블 만들기

03

EFCore for PostgreSQL,MySQL

1. EF Core 공통 패키지 설치
2. PostgreSQL,MySQL 패키지 설치
3. 모델 및 dbContext 만들기
4. db연결문자열 구성
5. dbContext DI 구성
6. dotnet-ef 명령어기반 테이블 만들기

01

EF Core & RDBMS & ORM

1. EF Core 소개
2. EF Core 주요 구성요소
3. Database Provider based EF Core
4. Global RDBMS Trends
5. ORM 이해하기

1. EF Core 소개

1) Entity Framework Core (EF Core)의 특징

- 데이터베이스 테이블을 개체로 매핑하는 **ORM 기반 프레임워크**
- ORM 기반 개체 지향적 방식으로 데이터를 관리가능
- **데이터베이스와의 상호작용을 관리하기** 위해 **DbContext 클래스 사용**
- DbContext 클래스는 데이터베이스 세션을 대표하며, 쿼리 및 트랜잭션 처리를 담당
- 코드를 통해 데이터베이스 스키마를 정의하고 관리하는 **Code-First 접근법 지원**.
- 소스 코드의 변경에 따라 데이터베이스 스키마를 자동으로 업데이트하는 마이그레이션 기능제공
- EF Core는 **LINQ를 사용하여 데이터베이스 쿼리를 더 직관적이고 편리하게 작성**할 수 있게 지원

- **2025년 01월 기준 : EFCore8(LTS), EFCore9 제공**

EF Core 8.0 시작하기
https://www.dotnetnote.com/docs/efcore/efcore-getting-started/

ASP.NET CORE MVC for EF Core 8
https://learn.microsoft.com/ko-kr/aspnet/core/data/ef-mvc/?view=aspnetcore-8.0

2. EF Core 주요 구성요소

1) EF Core 주요 구성요소 패키지 소개

- **EF Core 기반 ORM DB Programming시 필수 프로젝트 설치 구성요소**

1.1) Microsoft.EntityFrameworkCore

- .NET용 최신 개체-데이터베이스 매퍼
- LINQ 쿼리, 변경 추적, 업데이트 및 스키마 마이그레이션을 지원

1.2) Microsoft.EntityFrameworkCore.Design

- 모델,DbContext 기반 ORM 설계 지원 도구
- 마이그레이션을 관리하고 데이터베이스 스키마를 역 엔지니어링 처리지원
- DbContext 및 엔터티 유형을 스캐폴딩하는데 사용.
- 명령줄 또는 패키지 관리자 콘솔 기반 작업시 필요
- dotnet-ef 및 Microsoft.EntityFrameworkCore.Tools의 종속성

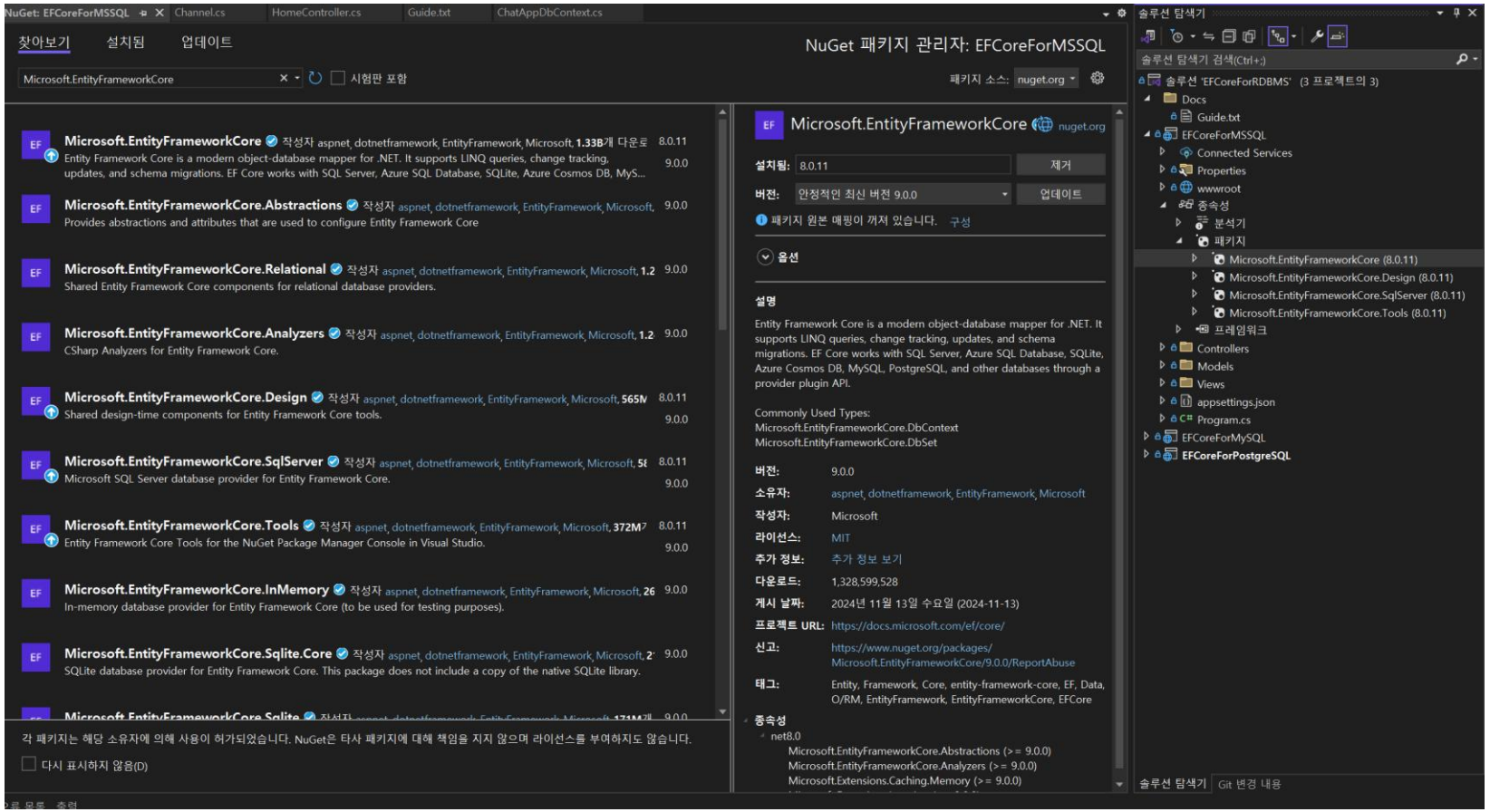
1.3) Microsoft.EntityFrameworkCore.Tools

- Visual Studio의 NuGet 패키지 관리자 콘솔을 위한 Entity Framework Core 도구

2. EF Core 주요 구성요소

2) EF Core8 공용 패키지 설치하기

- 프로젝트 선택 > Nuget 패키지 관리자 선택 > 하기 3개 패키지 프로젝트에 설치하기
- Microsoft.EntityFrameworkCore 8.0.11
- Microsoft.EntityFrameworkCore.Design 8.0.11
- Microsoft.EntityFrameworkCore.Tools 8.0.11



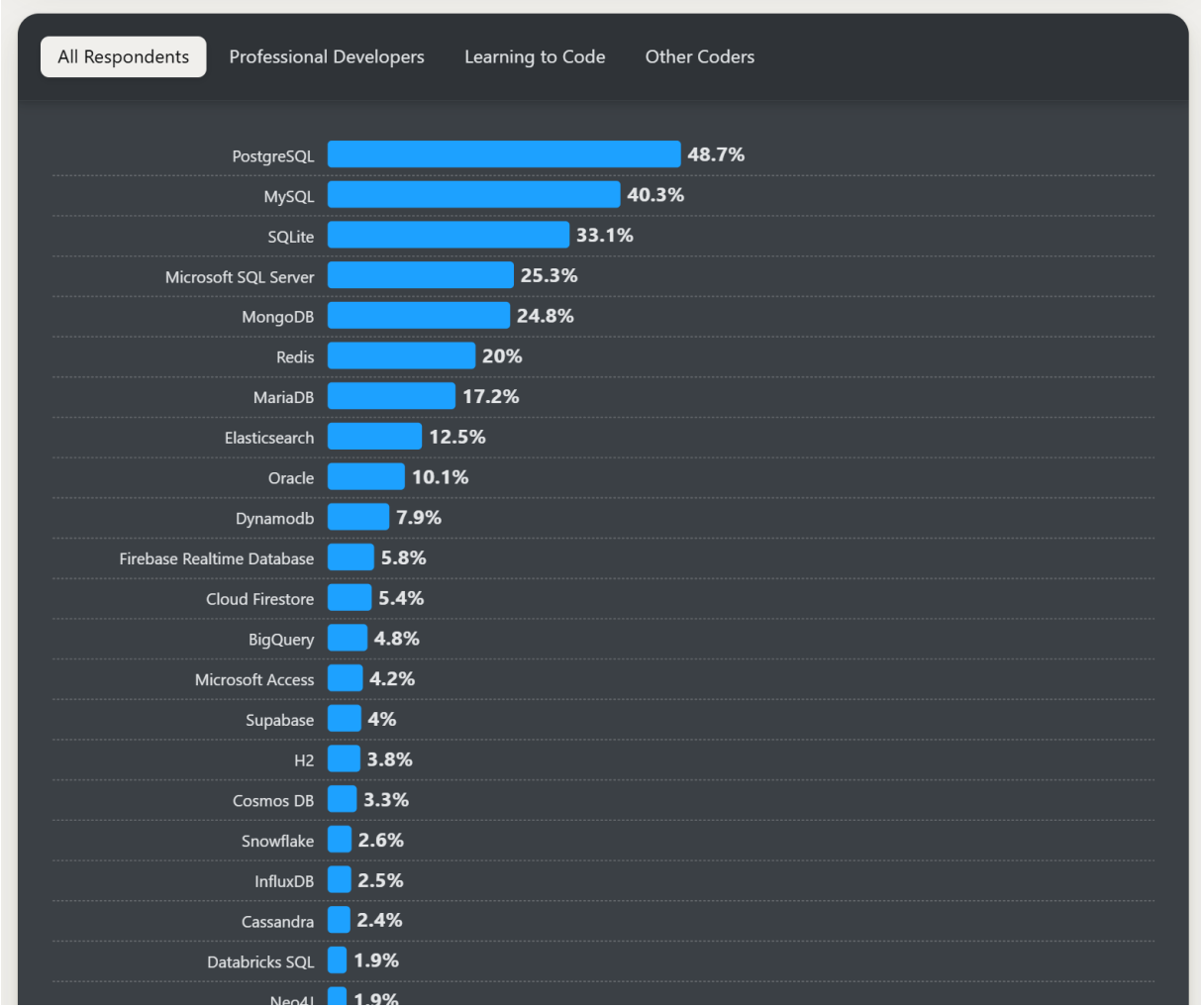
3. Database Provider based EF Core

- 대부분의 글로벌 인기 RDBMS는 모두지원
- **EF Core 주요 구성요소(3개)는 ORM환경 만 지원하며 RDBMS 공급자에 맞는 추가 구성요소 설치필요**
- **EF Core 주요 구성요소 버전별 공급자 지원 구성요소 버전 사용 해야함**

EF CORE DATABASE 공급자 주요 정보
https://learn.microsoft.com/ko-kr/ef/core/providers/?tabs=dotnet-core-cli

4. Global RDBMS Trends

- 글로벌 RDBMS 인기도 순위
- 글로벌 1,2위는 오픈소스 기반 RDBMS 트렌드로 변화
- **포스트그레스큐엘 RDBMS의 사용율,인기도 급 상승중...**
- **관계형 데이터,그래프 데이터, 벡터 데이터 지원 확장**

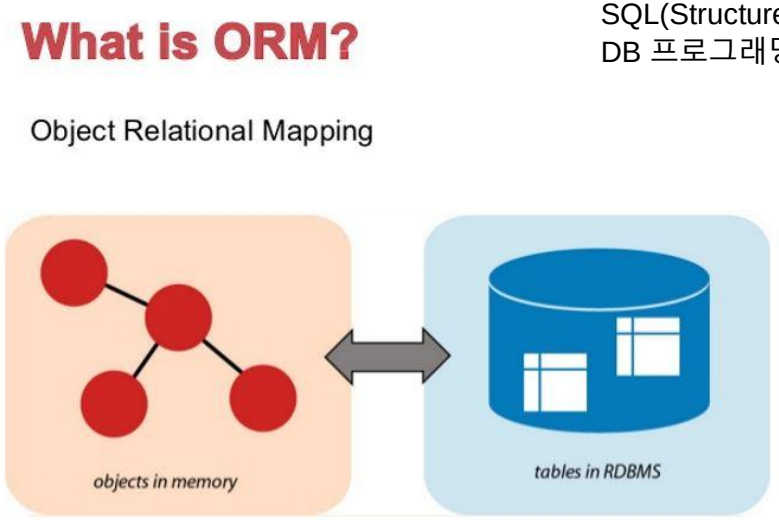


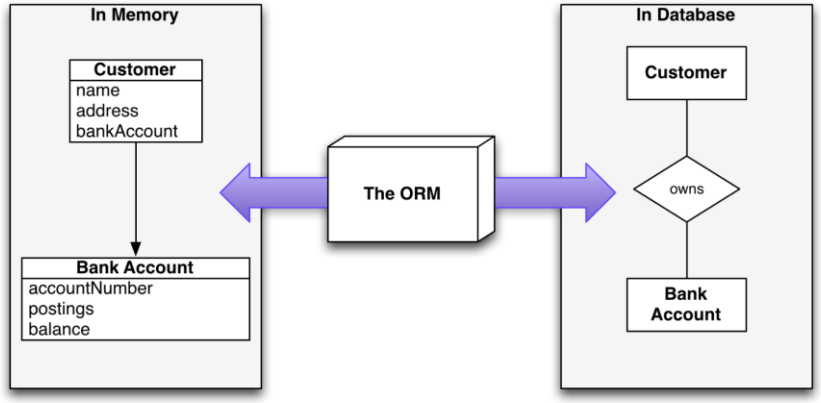
- **StackOverFlow Developer Survey 2024**

https://survey.stackoverflow.co/2024
https://survey.stackoverflow.co/2024/technology

5. ORM 이해하기

ORM은 객체지향 프로그램언어를 이용해 DB 프로그램밍을 할 때 RDBMS의 데이터베이스 구조 및 데이터를 관리하기 위한 SQL(Structured Query Language)를 잘 모르더라도 프로그래밍적 인 방법을 이용 쉽게 데이터 모델 객체 기반으로 DB 프로그래밍을 할 수 있는 환경을 제공합니다.





ORM작동원리:
ORM 메소드를 이용해 SQL구문을 생성 DB서버에 전달
SQL을 사용하지 않는게 아니고
ORM Framework이 SQL구문을 자동생성해서 DB서버에 전달한다.

5. ORM 이해하기

1)Model?

- Model이란 데이터의 구조를 프로그래밍 언어로 표현한 클래스
- 일반적으로 데이터를 저장하는 개별 테이블의 구조와 맵핑 되는 모델 클래스를 생성한다.

2)Model의 유형

- **Data Model : 데이터의 구조를 클래스의 속성으로 표현하고 데이터를 저장하는것이 목적, RDBMS Table과 맵핑**
- **Domain Model : 실제 업무 기준으로 데이터 구조를 표현하며 테이블과는 구조가 다소 상이할수 있다.**
- **View Model : 뷰(화면)에서 데이터 바인딩을 위해 전문적으로 사용하는 뷰 전용 모델**
- **DTO Model : Data Transfer Object ;프로그래밍 계층간 대량(복합) 데이터 전송을 위해 여러 모델을 하나의 전송모델로 재구성한 모델**

3) ORM 장점

- ORM이란 DB의 테이블 과 맵핑되는 프로그램의 데이터 모델 클래스를 이용해 DB프로그램을 구현하는 방법
- **CRUD작업을 위해 SQL 언어를 직접 사용하는것이 아닌 프로그램 언어를 이용해 CRUD 작업이 가능**

- SQL을 몰라도 어플리케이션 개발언어를 이용해 데이터 처리업무를 개발한다(내부적으로는 SQL로 자동변환 RDBMS에서 실행)

- ORM이 나온 배경은 데이터 처리 업무를 위해 별도로 표준 SQL(ANSI-SQL)을 배워야하고 RDBMS마다 표준 SQL(ANSI-SQL)을 사용하지 않는 상황이 많아 RDBMS마다 특화된 SQL사용법을 별도 알아야 하는 개발자들에게 개발자 친숙한 개발언어로 DB프로그래밍을 할수 쉽게 처리할수 있는 환경을 제공코자 함.

- **ORM을 이용하면 RDBMS 종속적인 어플리케이션이 아닌 RDBMS를 변경해도 쉽게 대응이 가능하다.**
- **최소한의 SQL 언어는 알아야 DB에 저장된 데이터에 대한 조작이나 고급 쿼리 작성시 도움된다.**

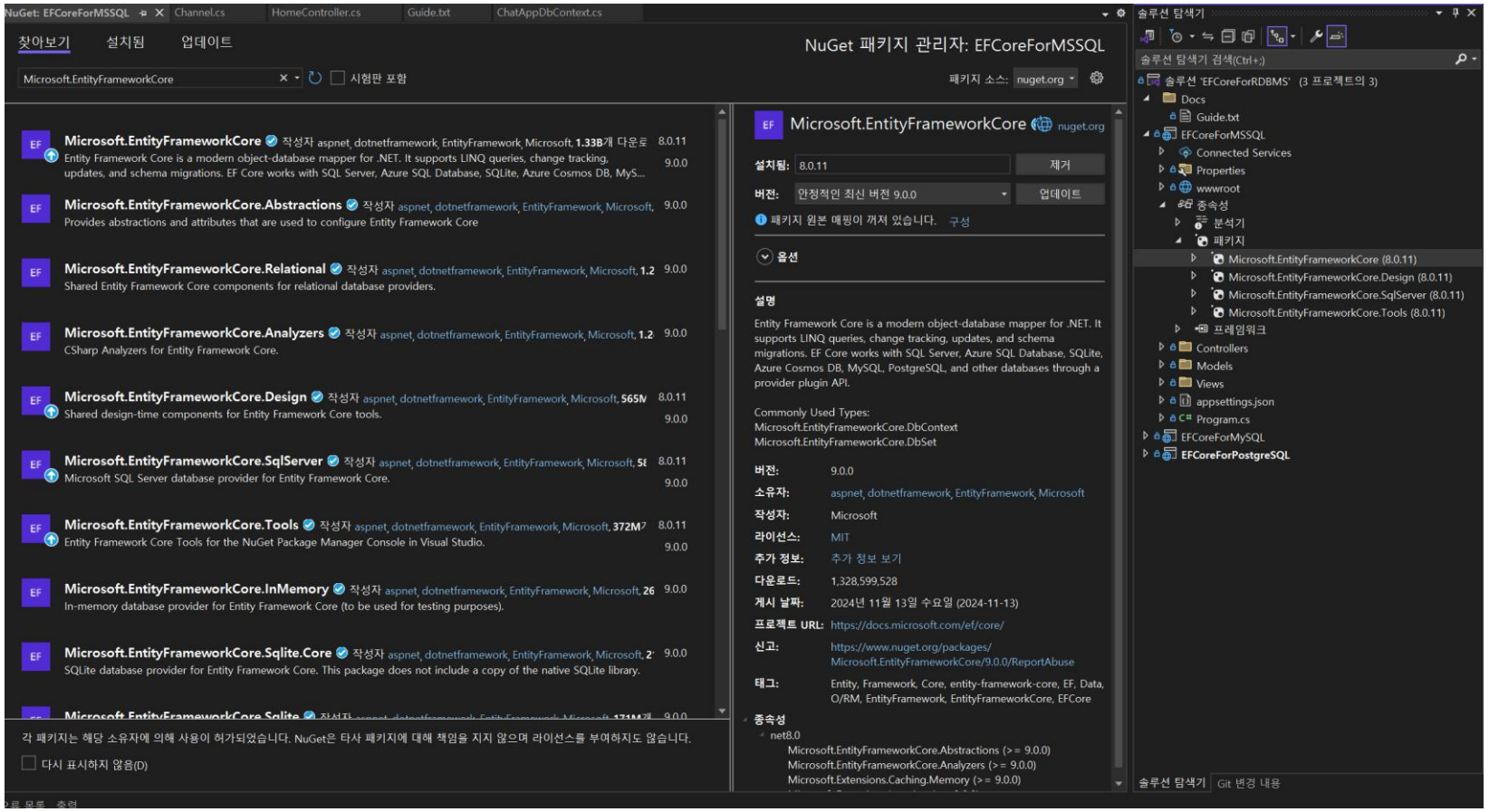
02

Code First ORM Process

1. EF Core 공통 패키지 설치
2. MS SQL 지원 패키지 설치-MS SQL
3. 모델 및 dbContext 만들기
4. db연결문자열 구성
5. dbContext DI 구성
6. dotnet-ef 명령어기반 테이블 만들기

1. EF Core 공통 구성요소 설치

- **프로젝트 선택 > Nuget 패키지 관리자 선택 > 하기 3개 패키지 프로젝트에 설치하기**
- **Microsoft.EntityFrameworkCore 8.0.11**
- **Microsoft.EntityFrameworkCore.Design 8.0.11**
- **Microsoft.EntityFrameworkCore.Tools 8.0.11**



2. MS SQL 지원 패키지 설치

- 프로젝트 선택 > Nuget 패키지 관리자 선택 > 하기 MSSQL 지원 패키지 프로젝트에 설치하기
- Microsoft.EntityFrameworkCore.SqlServer 8.0.11 설치

- MS SQL 설치 가이드

https://servermon.tistory.com/597

3. 모델 및 dbContext 만들기

1) 모델 클래스 만들기

Models/Channel.cs

```
public class Channel
{
    public int ChannelIdx { get; set; }

    public string ChannelName { get; set; }
    public string? ChannelDesc { get; set; }

    ...

    public int? ModifyMemberIdx { get; set; }
    public DateTime? ModifyDate { get; set; }
}
```

2) DbContext 클래스 만들기

-Models/ChatAppDbContext.cs

```
using Microsoft.EntityFrameworkCore;
public class ChatAppDbContext : DbContext {
    public ChatAppDbContext(DbContextOptions<ChatAppDbContext> options) : base(options){}
    public DbSet<Channel> Channels { get; set; }
    protected override void OnModelCreating(ModelBuilder modelBuilder){
        modelBuilder.Entity<Channel>(entity => {
            entity.ToTable("channels");
            entity.Property(e => e.ChannelIdx).UseIdentityColumn();
            entity.Property(e => e.ChannelIdx).HasColumnName("channel_idx");
            entity.HasKey(e => e.ChannelIdx).HasName("channel_pk");
            entity.Property(e => e.ChannelName).HasColumnName("channel_name").HasMaxLength(100);
            entity.Property(e => e.ChannelDesc).HasColumnName("channel_desc").HasMaxLength(500).IsRequired(false);
             .....
            entity.Property(e => e.ModifyMemberIdx).HasColumnName("modify_member_idx").IsRequired(false);
            entity.Property(e => e.ModifyDate).HasColumnName("modify_date").IsRequired(false);
        }); } }
```

4. db연결문자열 구성

- 프로젝트 루트에 appsettings.json 파일내
- DBMS유형별 DB서버 연결문자열 구성하기

appsettings.json

```
"ConnectionStrings": {
    "DefaultConnection": "Data Source=EDDYKANG;Initial Catalog=corechatapp;Integrated Security=false;user id=test;password=eddy524640!;TrustServerCertificate=Yes;Encrypt=False;"
  }
```

.

5. dbContext DI 구성

- 프로젝트 진입점 클래스(Program.cs) 나 Startup.cs파일내에
- DbContext 클래스를 의존성주입(DI)로 서비스 주입처리하여
- 해당 프로젝트 전역에서 dbContext 사용될수 있게 구성

-Program.cs

```
using EFCoreForMSSQL.Models;

using Microsoft.EntityFrameworkCore;


string? rdbConString = builder.Configuration.GetConnectionString("DefaultConnection");

builder.Services.AddDbContext<ChatAppDbContext>(c => c.UseSqlServer(rdbConString));
```

6. dotnet-ef 명령어기반 Code First 테이블 만들기

1) dotnet-ef CLI 툴 설치하기

- Visual Studio > 도구 > Nuget 팩키지 관리자 콘솔 실행

dotnet tool install --global dotnet-ef --version 8.0.11

- dotnet-ef 설치여부확인하기

dotnet ef

2) dotnet-ef CLI 명령어 기반 Code First 기반 모델기반 테이블 만들기

- cd ls
- cd 프로젝트로 이동

- **dotnet ef migrations add InitialCreate -o Models/Migrations**
  ㄴ실행계획수립 : Models/Migrations폴더를 만들고 실행계획을 미리 준비

- **dotnet ef database update**
  ㄴ실행계획에 따라 물리적 DB에 모델 기반 물리 테이블 생성: 물리적 Database는 미리 생성 해두어야함

dotnet-ef CLI Tool 소개

- https://learn.microsoft.com/en-us/ef/core/cli/dotnet

ORM

Code First Vs DB First

- Model기반 물리 테이블 생성기법
- 신규 프로젝트에서 주로 사용
- 백엔드 앱, DBMS 새로 개발

기존 물리 테이블기반 모델 생성기법
기존 DBMS는 그대로 사용하고
백엔드앱만 새로 개발하는 경우

7. dbContext 활용하기

- 어플리케이션 시작시 의존성 주입으로 생성된 DbContext 객체를 필요한 영역(Controller,Service,Repository) 계층에서 생성자에서 DbContext 객체를 호출하여 사용함

```
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    private readonly ChatAppDbContext _db;

    public HomeController(ChatAppDbContext context, ILogger<HomeController> logger)
    {
        _logger = logger;
        _db = context;
    }


    public async Task<IActionResult> Index()
    {
        Channel channel = new Channel() { ChannelName = "샘플채널1", ChannelTypeCode = 1, LimitCnt = 100, StateCode = 1, RegistMemberIdx = 1, RegistDate = DateTime.Now };
        var registedChannel = await _db.Channels.AddAsync(channel);
        await _db.SaveChangesAsync();


        var channels = await _db.Channels.ToListAsync();

        return View();
    }
}
```

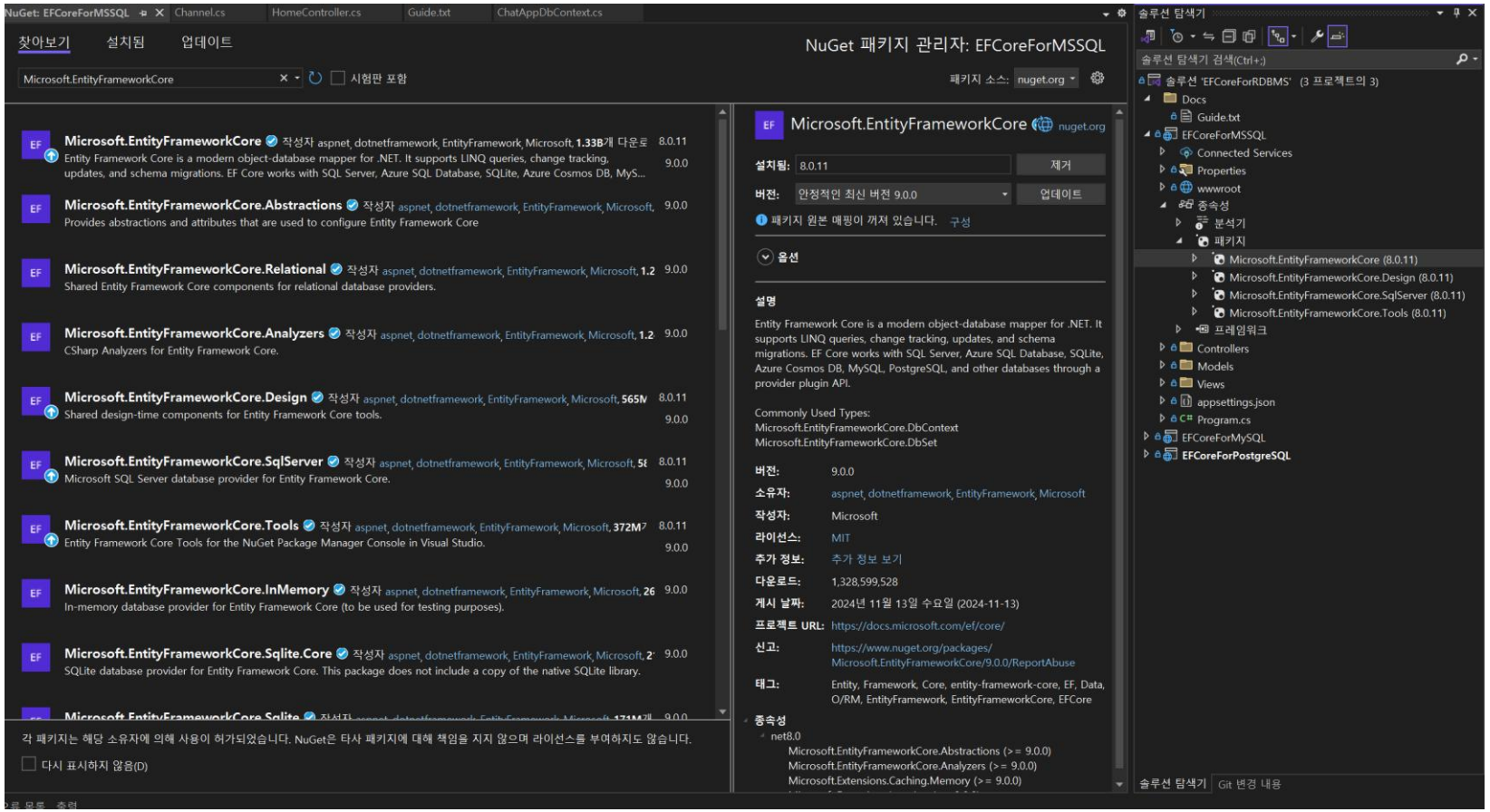
03

EFCore for PostgreSQL,MySQL

1. EF Core 공통 패키지 설치
2. PostgreSQL,MySQL 지원 패키지 설치
3. 모델 및 dbContext 만들기
4. db연결문자열 구성
5. dbContext DI 구성
6. dotnet-ef 명령어기반 테이블 만들기

1. EF Core 공통 구성요소 설치

- **프로젝트 선택 > Nuget 패키지 관리자 선택 > 하기 3개 패키지 프로젝트에 설치하기**
- **Microsoft.EntityFrameworkCore 8.0.11**
- **Microsoft.EntityFrameworkCore.Design 8.0.11**
- **Microsoft.EntityFrameworkCore.Tools 8.0.11**



2. PostgreSQL 또는 MySQL 지원 패키지 설치

1) PostgreSQL 구성요소 설치하기

- 프로젝트 선택 > Nuget 패키지 관리자 선택 > 하기 패키지 프로젝트에 설치하기
- Npgsql.EntityFrameworkCore.PostgreSQL 8.0.11 설치

- PostgreSQL 소개 및 설치 동영상 가이드
https://mixedcode.com/blog/83

2) MySQL 구성요소 설치하기

- 프로젝트 선택 > Nuget 패키지 관리자 선택 > 하기 패키지 프로젝트에 설치하기
- Pomelo.EntityFrameworkCore.MySql 8.0.2 설치

- MySQL 소개 및 설치 동영상 가이드
https://mixedcode.com/blog/52

3. 모델 및 dbContext 만들기

1) 모델 클래스 만들기

Models/Channel.cs

```
public class Channel
{
    public int ChannelIdx { get; set; }

    public string ChannelName { get; set; }
    public string? ChannelDesc { get; set; }

    ...

    public int? ModifyMemberIdx { get; set; }
    public DateTime? ModifyDate { get; set; }
}
```

2) DbContext 클래스 만들기 -PostgreSQL

-Models/ChatAppDbContext.cs

```
using Microsoft.EntityFrameworkCore;
public class ChatAppDbContext : DbContext {
    public ChatAppDbContext(DbContextOptions<ChatAppDbContext> options) : base(options){}
    public DbSet<Channel> Channels { get; set; }
    protected override void OnModelCreating(ModelBuilder modelBuilder){
        modelBuilder.Entity<Channel>(entity => {
            entity.ToTable("channels");
            entity.Property(e => e.ChannelIdx).UseIdentityColumn();
            entity.Property(e => e.ChannelIdx).HasColumnName("channel_idx");
            entity.HasKey(e => e.ChannelIdx).HasName("channel_pk");
            entity.Property(e => e.ChannelName).HasColumnName("channel_name").HasMaxLength(100);
            entity.Property(e => e.ChannelDesc).HasColumnName("channel_desc").HasMaxLength(500).IsRequired(false);
             .....
            entity.Property(e => e.ModifyMemberIdx).HasColumnName("modify_member_idx").IsRequired(false);
            entity.Property(e => e.ModifyDate).HasColumnName("modify_date").IsRequired(false).HasColumnType("timestamp without time zone");
```

3. 모델 및 dbContext 만들기

1) 모델 클래스 만들기

Models/Channel.cs

```
public class Channel
{
    public int ChannelIdx { get; set; }

    public string ChannelName { get; set; }
    public string? ChannelDesc { get; set; }

    ...

    public int? ModifyMemberIdx { get; set; }
    public DateTime? ModifyDate { get; set; }
}
```

2) DbContext 클래스 만들기-MySQL

-Models/ChatAppDbContext.cs

```
using Microsoft.EntityFrameworkCore;
public class ChatAppDbContext : DbContext {
    public ChatAppDbContext(DbContextOptions<ChatAppDbContext> options) : base(options){}
    public DbSet<Channel> Channels { get; set; }
    protected override void OnModelCreating(ModelBuilder modelBuilder){
        modelBuilder.Entity<Channel>(entity => {
            entity.ToTable("channels");
            entity.Property(e => e.ChannelIdx).UseIdentityColumn();
            entity.Property(e => e.ChannelIdx).HasColumnName("channel_idx");
            entity.HasKey(e => e.ChannelIdx).HasName("channel_pk");
            entity.Property(e => e.ChannelName).HasColumnName("channel_name").HasMaxLength(100);
            entity.Property(e => e.ChannelDesc).HasColumnName("channel_desc").HasMaxLength(500).IsRequired(false);
             .....
            entity.Property(e => e.ModifyMemberIdx).HasColumnName("modify_member_idx").IsRequired(false);
            entity.Property(e => e.ModifyDate).HasColumnName("modify_date").IsRequired(false);
        }); } }
```

4. db연결문자열 구성

- 프로젝트 루트에 appsettings.json 파일내
- DBMS유형별 DB서버 연결문자열 구성하기

appsettings.json for PostgreSQL

```
"ConnectionStrings": {
    "DefaultConnection": "Host=127.0.0.1;Database=corechatapp;Username=postgres;Password=eddy524640!“
}
```

appsettings.json for MySQL

```
"ConnectionStrings": {
    "DefaultConnection": "Server=127.0.0.1;Database=corechatapp;User=root;Password=eddy524640!;“
 }
```

.

5. dbContext DI 구성

- 프로젝트 진입점 클래스(Program.cs) 나 Startup.cs파일내에
- DbContext 클래스를 의존성주입(DI)로 서비스 주입처리하여
- 해당 프로젝트 전역에서 dbContext 사용될수 있게 구성

-Program.cs For PostgreSQL

```
using EFCoreForMSSQL.Models;
using Microsoft.EntityFrameworkCore;
string? rdbConString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ChatAppDbContext>(c => c.UseNpgsql(rdbConString));
```

-Program.cs For MySQL

```
using EFCoreForMSSQL.Models;
using Microsoft.EntityFrameworkCore;
string? rdbConString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ChatAppDbContext>(c => c.UseMySql(rdbConString, serverVersion).LogTo(Console.WriteLine, LogLevel.Information)
        .EnableSensitiveDataLogging()
        .EnableDetailedErrors());
```

6. dotnet-ef 명령어기반 Code First 테이블 만들기

1) dotnet-ef CLI 툴 설치하기

- Visual Studio > 도구 > Nuget 팩키지 관리자 콘솔 실행

dotnet tool install --global dotnet-ef --version 8.0.11

- dotnet-ef 설치여부확인하기

dotnet ef

2) dotnet-ef CLI 명령어 기반 Code First 기반 모델기반 테이블 만들기

- cd ls
- cd 프로젝트로 이동

- **dotnet ef migrations add InitialCreate -o Models/Migrations**

- **dotnet ef database update**

ORM

Code First Vs DB First

dotnet-ef CLI Tool 소개

- https://learn.microsoft.com/en-us/ef/core/cli/dotnet

7. dbContext 활용하기

- 어플리케이션 시작시 의존성 주입으로 생성된 DbContext 객체를 필요한 영역(Controller,Service,Repository) 계층에서 생성자에서 DbContext 객체를 호출하여 사용함

```
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    private readonly ChatAppDbContext _db;

    public HomeController(ChatAppDbContext context, ILogger<HomeController> logger)
    {
        _logger = logger;
        _db = context;
    }


    public async Task<IActionResult> Index()
    {
        Channel channel = new Channel() { ChannelName = "샘플채널1", ChannelTypeCode = 1, LimitCnt = 100, StateCode = 1, RegistMemberIdx = 1, RegistDate = DateTime.Now };
        var registedChannel = await _db.Channels.AddAsync(channel);
        await _db.SaveChangesAsync();


        var channels = await _db.Channels.ToListAsync();

        return View();
    }
}
```

발표자 소개

글 쓰고
강의 하고
사업 하는 시니어 개발자 **강창훈** 입니다.
멘토링/진로상담 : 010-2760-5246 ceo@msoftware.co.kr

-주요 이력

현) 실시간 메시징 솔루션 기업 엠소프트웨어 대표(CEO/ CTO)
현) 글로벌 융합기술정보제공 플랫폼 믹스드코드닷컴 운영
현) 비영리 IT교육 단체 지니공공아카데미 공동설립자/운영자
현) SW 아키텍트,풀 스택개발자
현) Naver Cloud Master –AI/Dev

전) 모두의연구소 & 고용노동부 **자바스크립트 풀스택 교육과정 주강사**
전) 모두의연구소 전략기획팀 SW 교육사업 총괄PM
전) Microsoft MVP 어워드 수상 AI/DEV(2020년~2024년)
전) 스마트한 인공지능 챗봇 /ASP.NET MVC5 개발 및 서비스 외 1권 서적 저자
전) 마이크로소프트 골드파트너 **필라넷 선임 기술 컨설턴트**

- 충북대학교 AI 풀스택 개발자 여름방학 특강 강사- Next.JS/NodeExpress/LangChainJS/Cloud
- 전주 정보문화산업 진흥원-클라우드/서버리스/인공지능서비스 개발 강사 : 2020~2022년
- 서울시 남부여성발전센터 풀 스택 강사: 2019~2021:Node.js,React.js,Vue.js,Python
- 대림대학교 겸임교수 - 1,2학년-HTML/CSS/JavaScript/Node.js & Cloud
- 질병관리본부-클라우드/AI서비스과정(2019년) 강사
- KT/SKT/LG/여성가족부/특허청/병무청/기업은행 다수 프로젝트 수행
- 삼성 반도체 다수 프로젝트 수행(Application Architect)
- 분당 서울대 병원 차세대 TA(Technical Architect)

감사합니다.

EF Core 8.0 ORM Code First Programming For RDBMS

강창훈 | 믹스드코드닷컴

010-2760-5246
ceo@msoftware.co.kr
https://mixedcode.com
https://jiny.kr