# EF Core

## 목차



---

## 하위 문서

# 개요

## 문서의 목적

- SQL-Server 을 지원하는 .Net Framework 의 공식 ORM(Entity Framework Core) 을 도입하기 위한 방법을 연구한다.
- Entity Framework Core (이하 EF Core) 의 기능 중 어떤 주요 기능이 필요한지, 기존 개발 방식 보다 주의해야 할 점은 무엇인지 정의한다.

## 레퍼런스

- https://learn.microsoft.com/ko-kr/ef/core/

## 실습 프로젝트

- https://gitlab.mrblue.com/dhyoo23/toy-efcore-learning

# 버전 비교

## 목적

- 현재 사용가능한 버전, 특징, 문제점을 파악하여 현 프로젝트에 적절한 버전을 찾는것을 목표로 합니다.

## 버전별 비교

| | 6.0 | 7.0 |
|---|---|---|
| .Net Version | 6.0 | 6.0 |
| LTS | YES | No |
| 지원 만료일 | 2024. 11. 12 | 2024.05.14 |
| 주요 추가 기능 | 주요 기능 페이지<br>• 임시 테이블<br>• 기록 데이터 쿼리<br>  ◦ 데이터 복원이 가능하도록 데이터 변경 기록을 저장하는 기능<br>• 기록 데이터 복원<br>  ◦ 데이터 복원 테이블을 통해 특정 시점으로 데이터를 변경하는 기능 | 주요 기능 페이지<br>• JSON 열 기능<br>  ◦ 컬럼에 JSON 으로 저장된 데이터도 테이블 처럼 조회, 쿼리가 가능<br>• 대량 업데이트<br>  ◦ 일괄 처리 기능 (업데이트, 삭제)<br>• Save Changes 성능향상<br>  ◦ 필요한 경우에만 트랜잭션 사용<br>• 테이블 트리거 확인 필요<br>  ◦ 테이블 트리거가 있을시 별도의 코드 추가 필요<br>• 리버스 엔지니어링<br>  ◦ 데이터베이스의 컬럼및 기능들을 코드화 해 주는 기능 |

| | | |
|---|---|---|
| 주요 호환성 손상 | 현재 기준 항목 | 주요 호환성 손상<br>• Encrypt 기본값이 true<br>  ◦ 서버의 인증서가 유효하게 설정 되어야 함<br>  ◦ 클라이언트가 인증서를 신뢰 해야함<br>  ◦ 명시적으로 Encrypt false<br>• 일부 경고가 예외 발생 추가<br>  ◦ 어플리케이션 오류 확인 필요<br>• 테이블 트리거 정의 필요<br>  ◦ 주요 기능 |
| 기타 | | MS 의 목표 Performance Edition 으로 명명 하면서, 성능향상에 집중됨,<br><br>대부분의 개선사항은 MS SQL Server 와 관련 |
| 선택시 이점 | • LTS 이기 때문에 지원 기간이 김<br>• 새로운 기능들에 대한 이슈가 적음 | • 사용한다면 유용한 기능<br>  ◦ 리버스 엔지니어링<br>  ▪ 실제로 현재 DB에 사용할 수 있다면 어플리케이션에서 현재 DB 를 공유하는 방식에서 개발 속도의 유리함을 가져갈 수 있을것으로 보임<br>  ◦ JSON 열 기능<br>  ▪ JSON 형태로 데이터를 저장하는 경우가 발생할 수 있다, 위와 같이 지원해 준다면, 좀더 테이블 정의가 작아지고 편리해질 수 있다고 생각됨. |

# 이슈 목록

## EF Core 7

| # | 이슈 타입 | 해설 | 기타 | 링크 |
|---|---|---|---|---|
| 1 | 성능 | Cartesian Explosion 문제 (JOIN 시 ) | EF Core 공통의 문제일 가능성이 있음 | https://www.thinktecture.com/en/entity-framework-core/ef-core-7-performance-cartesian-explosion/ |
| 2 | 성능 | N+1 문제, 지연 로드에서만 발생 | EF Core 공통의 문제일 가능성이 있음 | https://www.thinktecture.com/en/entity-framework-core/entity-framework-core7-n1-queries-problem/ |

## 공통 이슈

| # | 이슈 타입 | 해설 | 기타 | 링크 |
|---|---|---|---|---|
| 1 | 성능 | SQLClient 에서 대용량 데이터 사용시 비동기 API 사용하면 매우 느려지는 현상 | 발생 이슈는 SqlClient 5.0.1 (EF Core 7)<br><br>• EF Core 8 : SqlClient : 5.1.1<br>• EF Core 7 : SqlClient : 5.0.1<br>• EF Core 6 : SqlClient : 2.1.4<br><br>issue 593에 따르면 대용량 데이터를 처리할때 이슈가 발생하는건 1년내내 끌고 있는 부분 (=계속해서 발생됨) 이고 이는 SqlClient 의 문제로 추정됩니다.<br><br>이는 상당히 특수한 상황으로 생각됩니다. (EF Core 팀에서도 정확한 문제원인을 진단하고 있지 못해 보입니다.) | • https://github.com/dotnet/efcore/issues/21147<br>• https://github.com/dotnet/SqlClient/issues/1837<br>• https://github.com/dotnet/SqlClient/issues/593 |

# 작업 진행 방안

1. 버전과 관계 없는 내용, 필수적인 기능부터 처리
2. 버전 결정되면 부가 기능, 버전의 오류 사항들에 대한 진행 시작

---

# 연구 예정 내용

## 필수 기능

- EF 에서 제공하는 데이터베이스 와의 상호 작용들을 예시와 함께 정의한다.

| 항목 | 예시 | 레퍼런스 |
| --- | --- | --- |
| 테이블 관리 | • CRUD<br>• 관계<br>• 인덱싱<br>• 생성된 값 | https://learn.microsoft.com/ko-kr/ef/core/modeling/ |
| 저장 프로시저 | • CRUD | https://learn.microsoft.com/ko-kr/ef/ef6/modeling/code-first/fluent/cud-stored-procedures |
| 이벤트 | | https://learn.microsoft.com/ko-kr/ef/core/logging-events-diagnostics/events |
| 쿼리 | | https://learn.microsoft.com/ko-kr/ef/core/querying/sql-queries |

## 부가 기능

### 마이그레이션

- DB 를 코드 단에서 관리할 수 있는 전략을 가져 간다.
- 차후 Micro Service 화 될때, 코드단에서 관리할 수 있도록 돕는다.

| 항목 | 예시 | 레퍼런스 |
| --- | --- | --- |
| 데이터베이스 생성 / 삭제 | | https://learn.microsoft.com/ko-kr/aspnet/core/data/ef-rp/intro?view=aspnetcore-7.0&tabs=visual-studio#create-the-database |
| 마이그레이션<br><br>(현재 고려 사항 X) | | https://learn.microsoft.com/ko-kr/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli |
| CI/CD 를 이용한 데이터베이스 마이그레이션<br><br>(현재 고려 사항 X) | • 데이터베이스 Migration 을 진행하는 Console, 이 콘솔이 마이그레이션 및 테스트 진행. 실패시 롤백 | |

### 리버스 엔지니어링

- 실제 데이터베이스에 접속하여 자동생성된 코드들이 실제로 동작 하는지 확인한다.

# 기타 기능

## 테스트

- EF Core 을 사용하여 통합 테스트 및 유닛 테스트 전략에 대해서 정의한다.

| 항목 | 예시 | 레퍼런스 |
| --- | --- | --- |
| 통합 테스트 전략 | • 통합 테스트시 개발 DB 를 사용하며 테스트가 완료되면 삽입한 데이터를 모두 삭제할 수 있도록 한다. | |
| 유닛 테스트 전략 | • 임시 데이터베이스를 생성하여 테스트를 진행하며, 종료 후에는 임시 데이터베이스를 삭제한다. | |

# 이슈 사항

## 주요 이슈

- ORM 사용시 현재 발생하는 문제점은 무엇이고, 어떻게 해결 / 전략을 논의 해야하는지 정의한다.

## 성능

- ORM 사용시 성능적 이슈로 문제가 발생할 수 있는것을 최소화 한다.

| 항목 | 예시 | 레퍼런스 |
| --- | --- | --- |
| 효율적 쿼리<br><br>(현재 고려 사항 X) | | https://learn.microsoft.com/ko-kr/ef/core/performance/efficient-querying |
| 효율적 업데이트<br><br>(현재 고려 사항 X) | | https://learn.microsoft.com/ko-kr/ef/core/performance/efficient-updating?tabs=ef7<br><br>기타 고려 사항 (비동기 이슈)<br>• https://github.com/dotnet/SqlClient/issues/593<br>• https://github.com/dotnet/SqlClient/issues/601 |
| 고급 성능 토픽<br><br>(현재 고려 사항 X) | | https://learn.microsoft.com/ko-kr/ef/core/performance/advanced-performance-topics?tabs=with-di%2Cexpression-api-with-constant |
| Cartesian Explosion<br><br>(현재 고려 사항 X) | | https://www.thinktecture.com/en/entity-framework-core/ef-core-7-performance-cartesian-explosion/ |
| N + 1<br><br>(현재 고려 사항 X) | | https://www.thinktecture.com/en/entity-framework-core/entity-framework-core7-n1-queries-problem/ |

### 보안

- 잘 알려진 SQL 공격 문제를 방어하는 전략을 수립한다.

| 항목 | 예시 | 레퍼런스 |
| --- | --- | --- |
| SQL Injection<br><br>(현재 고려 사항 X) | • FromSqlRaw 등의 직접 쿼리문에서 발생할 수 있는 문제이다.<br>• 해결 방안<br>  ◦ Overposting/Mass Assignment Model Binding | |

---

# 결정 사항

프로젝트에서는 EF Core 6버전을 사용하도록 한다. 사유는 아래와 같다.

- EF 7 은 현재도 버그 픽스와 업데이트가 일어나고 있는 버전이다.
- EF 7 Join 관련 성능이슈가 해결 되지 않았다.
- EF 6 은 현재 거의 변경이 없는 버전으로, 안정화가 된 버전이다.