Previous Next

Querying in Entity Framework Core

Querying in Entity Framework Core remains the same as in EF 6.x, with more optimized SQL queries and the ability to include C#/VB.NET functions into LINQ-to-Entities queries.

Visit the LINQ-to-Entities chapter to learn more about the basics of querying in Entity Framework.

Here, you will learn the new features of querying introduced in Entity Framework Core.

C#/VB.NET Functions in Queries

EF Core has a new feature in LINQ-to-Entities where we can include C# or VB.NET functions in the query. This was not possible in EF 6.

```
private static void Main(string[] args)
{
    var context = new SchoolContext();
    var studentsWithSameName = context.Students
                                      .Where(s => s.FirstName == GetName())
                                      .ToList();
}

public static string GetName() {
    return "Bill";
}
```

In the above L2E query, we have included the `GetName()` C# function in the Where clause. This will execute the following query in the database:

```
exec sp_executesql N'SELECT [s].[StudentId], [s].[DoB], [s].[FirstName],
    [s].[GradeId], [s].[LastName], [s].[MiddleName]
FROM [Students] AS [s]
WHERE [s].[FirstName] = @__GetName_0',N'@__GetName_0 nvarchar(4000)',
    @__GetName_0=N'Bill'
Go
```

Eager Loading

Entity Framework Core supports eager loading of related entities, same as EF 6, using the `Include()` extension method and projection query. In addition to this, it also provides the `ThenInclude()` extension method to load multiple levels of related entities. (EF 6 does not support the `ThenInclude()` method.)

Include

Unlike EF 6, we can specify a lambda expression as a parameter in the `Include()` method to specify a navigation property as shown below.

```
var context = new SchoolContext();

var studentWithGrade = context.Students
                            .Where(s => s.FirstName == "Bill")
                            .Include(s => s.Grade)
                            .FirstOrDefault();
```

In the above example, `.Include(s => s.Grade)` passes the lambda expression `s => s.Grade` to specify a reference property to be loaded with `Student` entity data from the database in a single SQL query. The above query executes the following SQL query in the database.

```
SELECT TOP(1) [s].[StudentId], [s].[DoB], [s].[FirstName], [s].[GradeId],[s].
[LastName],
           [s].[MiddleName], [s.Grade].[GradeId], [s.Grade].[GradeName], [s.Grade].
[Section]
FROM [Students] AS [s]
LEFT JOIN [Grades] AS [s.Grade] ON [s].[GradeId] = [s.Grade].[GradeId]
WHERE [s].[FirstName] = N'Bill'
```

We can also specify property name as a string in the `Include()` method, same as in EF 6.

```
var context = new SchoolContext();

var studentWithGrade = context.Students
                        .Where(s => s.FirstName == "Bill")
                        .Include("Grade")
                        .FirstOrDefault();
```

The example above is not recommended because it will throw a runtime exception if a property name is misspelled or does not exist. Always use the `Include()` method with a lambda expression, so that the error can be detected during compile time.

The `Include()` extension method can also be used after the `FromSql()` method, as shown below.

```
var context = new SchoolContext();

var studentWithGrade = context.Students
                        .FromSql("Select * from Students where FirstName ='Bill'")
                        .Include(s => s.Grade)
                        .FirstOrDefault();
```

Note: The `Include()` extension method cannot be used after the `DbSet.Find()` method. E.g. `context.Students.Find(1).Include()` is not possible in EF Core 2.0. This may be possible in future versions.

Multiple Include

Use the `Include()` method multiple times to load multiple navigation properties of the same entity. For example, the following code loads `Grade` and `StudentCourses` related entities of `Student`.

```
var context = new SchoolContext();

var studentWithGrade = context.Students.Where(s => s.FirstName == "Bill")
                        .Include(s => s.Grade)
                        .Include(s => s.StudentCourses)
                        .FirstOrDefault();
```

The above query will execute two SQL queries in a single database round trip.

```
SELECT TOP(1) [s].[StudentId], [s].[DoB], [s].[FirstName], [s].[GradeId], [s].[LastName],
           [s].[MiddleName], [s.Grade].[GradeId], [s.Grade].[GradeName], [s.Grade].[Section]
FROM [Students] AS [s]
LEFT JOIN [Grades] AS [s.Grade] ON [s].[GradeId] = [s.Grade].[GradeId]
WHERE [s].[FirstName] = N'Bill'
ORDER BY [s].[StudentId]
Go

SELECT [s.StudentCourses].[StudentId], [s.StudentCourses].[CourseId]
FROM [StudentCourses] AS [s.StudentCourses]
INNER JOIN (
    SELECT DISTINCT [t].*
    FROM (
        SELECT TOP(1) [s0].[StudentId]
        FROM [Students] AS [s0]
        LEFT JOIN [Grades] AS [s.Grade0] ON [s0].[GradeId] = [s.Grade0].[GradeId]
        WHERE [s0].[FirstName] = N'Bill'
        ORDER BY [s0].[StudentId]
    ) AS [t]
) AS [t0] ON [s.StudentCourses].[StudentId] = [t0].[StudentId]
ORDER BY [t0].[StudentId]
Go
```

ThenInclude

EF Core introduced the new `ThenInclude()` extension method to load multiple levels of related entities. Consider the following example:

```
var context = new SchoolContext();

var student = context.Students.Where(s => s.FirstName == "Bill")
                        .Include(s => s.Grade)
                            .ThenInclude(g => g.Teachers)
                        .FirstOrDefault();
```

In the above example, `.Include(s => s.Grade)` will load the `Grade` reference navigation property of the `Student` entity. `.ThenInclude(g => g.Teachers)` will load the `Teacher` collection property of the `Grade` entity. The `ThenInclude` method must be called after the `Include` method. The above will execute the following SQL queries in the database.

```
SELECT TOP(1) [s].[StudentId], [s].[DoB], [s].[FirstName], [s].[GradeId], [s].
[LastName],
            [s].[MiddleName], [s.Grade].[GradeId], [s.Grade].[GradeName], [s.Grade].
[Section]
FROM [Students] AS [s]
LEFT JOIN [Grades] AS [s.Grade] ON [s].[GradeId] = [s.Grade].[GradeId]
WHERE [s].[FirstName] = N'Bill'
ORDER BY [s.Grade].[GradeId]
Go

SELECT          [s.Grade.Teachers].[TeacherId],          [s.Grade.Teachers].[GradeId],
[s.Grade.Teachers].[Name]
FROM [Teachers] AS [s.Grade.Teachers]
INNER JOIN (
    SELECT DISTINCT [t].*
    FROM (
        SELECT TOP(1) [s.Grade0].[GradeId]
        FROM [Students] AS [s0]
        LEFT JOIN [Grades] AS [s.Grade0] ON [s0].[GradeId] = [s.Grade0].[GradeId]
        WHERE [s0].[FirstName] = N'Bill'
        ORDER BY [s.Grade0].[GradeId]
    ) AS [t]
) AS [t0] ON [s.Grade.Teachers].[GradeId] = [t0].[GradeId]
ORDER BY [t0].[GradeId]
go
```

Projection Query

We can also load multiple related entities by using the projection query instead of `Include()` or `ThenInclude()` methods. The following example demonstrates the projection query to load the `Student`, `Grade`, and `Teacher` entities.

```
var context = new SchoolContext();

var stud = context.Students.Where(s => s.FirstName == "Bill")
                        .Select(s => new
                        {
                            Student = s,
                            Grade = s.Grade,
                            GradeTeachers = s.Grade.Teachers
                        })
                        .FirstOrDefault();
```

In the above example, the `.Select` extension method is used to include the `Student`, `Grade` and `Teacher` entities in the result. This will execute the same SQL query as the above `ThenInclude()` method.

Lazy Loading

Lazy loading is not supported in Entity Framework Core 2.0. Track lazy loading issue on github.

Explicit Loading

Explicit loading works the same way as in EF 6. Learn about it here.

Previous

Next

ENTITYFRAMEWORKTUTORIAL

Learn Entity Framework using simple yet practical examples on EntityFrameworkTutorial.net for free. Learn Entity Framework DB-First, Code-First and EF Core step by step. While using this site, you agree to have read and accepted our terms of use and privacy policy.

---

feedback@entityframeworktutorial.net

TUTORIALS

E-MAIL LIST

Subscribe to EntityFrameworkTutorial email list and get EF 6 and EF Core Cheat Sheets, latest updates, tips & tricks about Entity Framework to your inbox.

Email address GO

We respect your privacy.