< Previous Next >

Eager Loading in Entity Framework

Eager loading is the process whereby a query for one type of entity also loads related entities as part of the query, so that we don't need to execute a separate query for related entities. Eager loading is achieved using the **Include()** method.

In the following example, it gets all the students from the database along with its standards using the `Include`() method.

LINQ Query Syntax:

```
using (var context = new SchoolDBEntities())
{
    var stud1 = (from s in context.Students.Include("Standard")
                where s.StudentName == "Bill"
                select s).FirstOrDefault<Student>();
}
```

LINQ Method Syntax:

```
using (var ctx = new SchoolDBEntities())
{
    var stud1 = ctx.Students
                   .Include("Standard")
                   .Where(s => s.StudentName == "Bill")
                   .FirstOrDefault<Student>();
}
```

The above LINQ queries will result in following SQL query:

```
SELECT TOP (1)
[Extent1].[StudentID] AS [StudentID],
[Extent1].[StudentName] AS [StudentName],
[Extent2].[StandardId] AS [StandardId],
[Extent2].[StandardName] AS [StandardName],
[Extent2].[Description] AS [Description]
FROM  [dbo].[Student] AS [Extent1]
LEFT OUTER JOIN [dbo].[Standard] AS [Extent2] ON [Extent1].[StandardId] = [Extent2].[StandardId]
WHERE 'Bill' = [Extent1].[StudentName]
```

Use Lambda Expression

You can also use the LINQ lambda expression as a parameter in the `Include` method. For this, take a reference of `System.Data.Entity` namespace and use the lambda expression as shown below:

```
using System;
using System.Data.Entity;

class Program
{
    static void Main(string[] args)
    {
        using (var ctx = new SchoolDBEntities())
        {
            var stud1 = ctx.Students.Include(s => s.Standard)
                                .Where(s => s.StudentName == "Bill")
                                .FirstOrDefault<Student>();
        }
    }
}
```

Load Multiple Entities

You can also eagerly load multiple levels of related entities. The following example query eagerly loads the `Student`, `Standard` and `Teacher` entities:

```
using (var ctx = new SchoolDBEntities())
{
    var stud1 = ctx.Students.Include("Standard.Teachers")
                    .Where(s => s.StudentName == "Bill")
                    .FirstOrDefault<Student>();
}
```

Or use the lambda expression as below:

```
using (var ctx = new SchoolDBEntities())
{
    var stud1 = ctx.Students.Include(s => s.Standard.Teachers)
                    .Where(s => s.StudentName == "Bill")
                    .FirstOrDefault<Student>();
}
```

The above query will execute the following SQL query in the database:

```sql
SELECT [Project2].[StudentID] AS [StudentID],
[Project2].[StudentName] AS [StudentName],
[Project2].[StandardId] AS [StandardId],
[Project2].[StandardName] AS [StandardName],
[Project2].[Description] AS [Description],
[Project2].[C1] AS [C1],
[Project2].[TeacherId] AS [TeacherId],
[Project2].[TeacherName] AS [TeacherName],
[Project2].[StandardId1] AS [StandardId1]
FROM ( SELECT
    [Limit1].[StudentID] AS [StudentID],
    [Limit1].[StudentName] AS [StudentName],
    [Limit1].[StandardId1] AS [StandardId],
    [Limit1].[StandardName] AS [StandardName],
    [Limit1].[Description] AS [Description],
    [Project1].[TeacherId] AS [TeacherId],
    [Project1].[TeacherName] AS [TeacherName],
    [Project1].[StandardId] AS [StandardId1],
    CASE WHEN ([Project1].[TeacherId] IS NULL) THEN CAST(NULL AS int) ELSE 1 END AS [C1]
      FROM    (SELECT TOP (1) [Extent1].[StudentID] AS [StudentID], [Extent1].[StudentName] AS [StudentName],
[Extent1].[StandardId] AS [StandardId2], [Extent2].[StandardId] AS [StandardId1], [Extent2].[StandardName] AS
[StandardName], [Extent2].[Description] AS [Description]
        FROM  [dbo].[Student] AS [Extent1]
        LEFT OUTER JOIN [dbo].[Standard] AS [Extent2] ON [Extent1].[StandardId] = [Extent2].[StandardId]
        WHERE 'updated student' = [Extent1].[StudentName] ) AS [Limit1]
    LEFT OUTER JOIN  (SELECT
        [Extent3].[TeacherId] AS [TeacherId],
        [Extent3].[TeacherName] AS [TeacherName],
        [Extent3].[StandardId] AS [StandardId]
        FROM [dbo].[Teacher] AS [Extent3]
            WHERE [Extent3].[StandardId] IS NOT NULL ) AS [Project1] ON [Limit1].[StandardId2] = [Project1].
[StandardId]
)  AS [Project2]
ORDER BY [Project2].[StudentID] ASC, [Project2].[StandardId] ASC, [Project2].[C1] ASC
```

EF Core supports additional method `IncludeThen` for eager loading. Learn about it here.

Learn how Entity Framework 6.x supports lazy loading in the next chapter.

Download EF 6 DB-First Demo Project from Github

< Previous Next >

feedback@entityframeworktutorial.net

TUTORIALS

- EF Basics
- EF Core
- EF 6 DB-First
- EF 6 Code-First

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.

HOME PRIVACY POLICY ADVERTISE WITH US