<https://intellipaat.com/community/16430/identityinsert-is-set-to-off-how-to-turn-it-on>

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.

# Lazy Loading in Entity Framework

Lazy loading is delaying the loading of related data, until you specifically request for it. It is the opposite of [eager loading](https://www.entityframeworktutorial.net/eager-loading-in-entity-framework.aspx). For example, the Student entity contains the StudentAddress entity. In the lazy loading, the context first loads the Student entity data from the database, then it will load the StudentAddress entity when we access the StudentAddress property as shown below.

using (var ctx = new SchoolDBEntities())

{

//Loading students only

IList<Student> studList = ctx.Students.ToList<Student>();

Student std = studList[0];

//Loads Student address for particular Student only (seperate SQL query)

StudentAddress add = std.StudentAddress;

}

The code shown above will result in two SQL queries. First, it will fetch all students:

SELECT

[Extent1].[StudentID] AS [StudentID],

[Extent1].[StudentName] AS [StudentName],

[Extent1].[StandardId] AS [StandardId]

FROM [dbo].[Student] AS [Extent1]

Then, it will send the following query when we get the reference of StudentAddress:

exec sp\_executesql N'SELECT

[Extent1].[StudentID] AS [StudentID],

[Extent1].[Address1] AS [Address1],

[Extent1].[Address2] AS [Address2],

[Extent1].[City] AS [City],

[Extent1].[State] AS [State]

FROM [dbo].[StudentAddress] AS [Extent1]

WHERE [Extent1].[StudentID] = @EntityKeyValue1',N'@EntityKeyValue1 int',@EntityKeyValue1=1

## **Disable Lazy Loading**

We can disable lazy loading for a particular entity or a context. To turn off lazy loading for a particular property, do not make it virtual. To turn off lazy loading for all entities in the context, set its configuration property to false.

using System;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Data.Entity.Core.Objects;

using System.Linq;

public partial class SchoolDBEntities : DbContext

{

public SchoolDBEntities(): base("name=SchoolDBEntities")

{

this.Configuration.LazyLoadingEnabled = false;

}

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

}

}

**Rules for lazy loading:**

1. *context.Configuration.ProxyCreationEnabled* should be true.
2. *context.Configuration.LazyLoadingEnabled* should be true.
3. Navigation property should be defined as public, virtual. Context will **NOT** do lazy loading if the property is not defined as virtual.

# Explicit Loading in Entity Framework

Here you will learn how to load related entities in an entity graph explicitly. Explicit loading is valid in EF 6 and EF Core both.

Even with lazy loading disabled (in EF 6), it is still possible to lazily load related entities, but it must be done with an explicit call. Use the Load() method to load related entities explicitly. Consider the following example.

using (var context = new SchoolContext())

{

var student = context.Students

.Where(s => s.FirstName == "Bill")

.FirstOrDefault<Student>();

context.Entry(student).Reference(s => s.StudentAddress).Load(); // loads StudentAddress

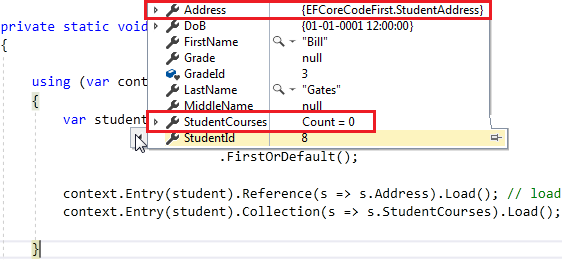
context.Entry(student).Collection(s => s.StudentCourses).Load(); // loads Courses collection

}

In the above example, context.Entry(student).Reference(s => s.StudentAddress).Load() loads the StudentAddress entity. The Reference() method is used to get an object of the specified reference navigation property and the Load() method loads it explicitly.

In the same way, context.Entry(student).Collection(s => s.Courses).Load() loads the collection navigation property Courses of the Student entity. The Collection()method gets an object that represents the collection navigation property.

The Load() method executes the SQL query in the database to get the data and fill up the specified reference or collection property in the memory, as shown below.

[](https://www.entityframeworktutorial.net/images/EF5/explicit-loading1.PNG)

## **Query()**

You can also write LINQ-to-Entities queries to filter the related data before loading. The Query() method enables us to write further LINQ queries for the related entities to filter out related data.

using (var context = new SchoolContext())

{

var student = context.Students

.Where(s => s.FirstName == "Bill")

.FirstOrDefault<Student>();

context.Entry(student)

.Collection(s => s.StudentCourses)

.Query()

.Where(sc => sc.CourseName == "Maths")

.FirstOrDefault();

}

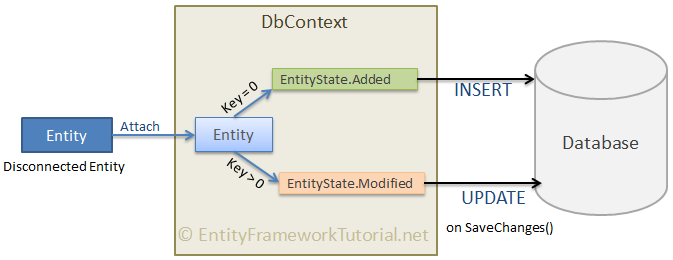
In the above example, .Collection(s => s.StudentCourses).Query() allows us to write further queries for the StudentCourses entity.

Saving a Disconnected Entity in EF 6

In this chapter, you will learn how to save a disconnected entity which is not being tracked by a context in EF 6.

Saving data in the disconnected scenario is a little bit different than in the connected scenario. In the disconnected scenario, an instance of DbContext is not aware of the disconnected entities because the entities were created or modified out of the scope of the current DbContext instance. So, you need to attach the disconnected entities to a context with an appropriate EntityState in order to perform INSERT or UPDATE operations in the database.

In the disconnected scenario, you need to find out whether an entity is new or existing and based on that you can set the EntityState. Here, if the key property value is zero then we will consider it a new entity and so we will set the Added state. If the key property value is greater than zero, then it means it is an existing entity and so we will set the Modified state.

[](https://www.entityframeworktutorial.net/images/ef6/save-disconnected-entity.png)

The following example demonstrates saving a disconnected entity.

// disconnected new entity

var student = new Student(){ StudentName = "Bill" };

using (var context = new SchoolDBEntities())

{

context.Entry(student).State = student.StudentId == 0? EntityState.Added : EntityState.Modified;

context.SaveChanges();

}

In the above example, studnet is a disconnected entity object and context is not aware of its state.context.Entry(student).State = student.StudentId == 0? EntityState.Added : EntityState.Modified;sets the Added state if the value of the key property StudentId is zero, otherwise it sets the Modified state. The SaveChanges() method will build and execute the following INSERT command to the database.

exec sp\_executesql N'INSERT [dbo].[Student]([StudentName], [StandardId])

VALUES (@0, NULL)

SELECT [StudentID] FROM [dbo].[Student]

WHERE @@ROWCOUNT > 0 AND [StudentID] = scope\_identity(),@0='Bill'

The same way, if the value of StudentId is non-zero, then it will assign the Modified state and so, the SaveChanges() method will execute the UPDATE command.

// disconnected existing entity

var student = new Student(){ StudentId = 1, StudentName = "Steve" };

using (var context = new SchoolDBEntities())

{

context.Entry(student).State = student.StudentId == 0? EntityState.Added : EntityState.Modified;

context.SaveChanges();

}

In the above example, an object of Student entity includes the key property StudentId greater than zero, so it will be marked as Modified. This will execute the following UPDATE command in the database.

exec sp\_executesql N'UPDATE [dbo].[Student]

SET [StudentName] = @0

WHERE @@ROWCOUNT > 0 AND [StudentID] = @1'N'@0 varchar(50),@1 int',@0='Steve',@1=1

[Download EF 6 DB-First Demo Project from Github](https://github.com/entityframeworktutorial/EF6-DBFirst-Demo)

[Previous](https://www.entityframeworktutorial.net/EntityFramework5/attach-disconnected-entity-graph.aspx)

[Next](https://www.entityframeworktutorial.net/entityframework6/save-entity-graph.aspx)

* Useful Resources
* [Fastest Way to Insert using EF Extensions](https://entityframework-extensions.net/fastest-way-to-insert?z=eft-tl)
* [Learn C#, MVC, ASP.NET Core, LINQ, etc.](https://www.tutorialsteacher.com/)
* [Entity Framework Courses on Pluralsight](https://pluralsight.pxf.io/c/407808/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fsearch%3Fq%3Dentity%2520framework)