Saving Data in the Connected Scenario

Saving entity data in the connected scenario is a fairly easy task because the context automatically tracks the changes that happened on the entity during its lifetime.

Here, we will use the same EDM for CRUD operations which we created in the Create Entity Data Model chapter. An entity which contains data in its scalar property will be either inserted, updated or deleted, based on its `EntityState`.

In the Entity Framework, there are two persistence scenarios to save an entity data: connected and disconnected. In the connected scenario, the same instance of `DbContext` is used in retrieving and saving entities, whereas this is different in the disconnected scenario. In this chapter, you will learn about saving data in the connected scenario.

The following figure illustrates the CUD (Create, Update, Delete) operations in the connected scenario.
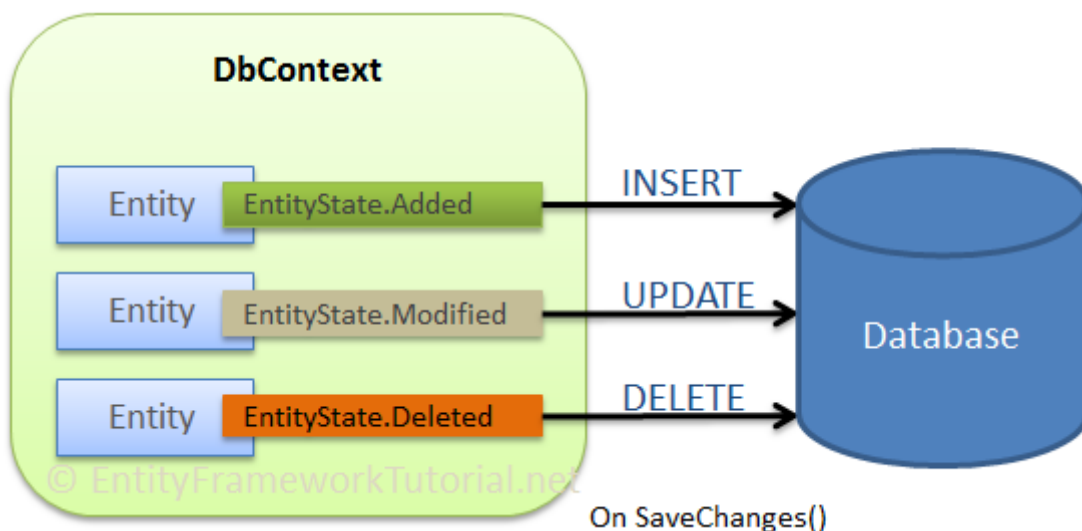
As per the above figure, Entity Framework builds and executes INSERT, UPDATE, and DELETE statements for the entities whose `EntityState` is Added, Modified, or Deleted when the `DbContext.SaveChanges()` method is called. In the connected scenario, an instance of `DbContext` keeps track of all the entities and so, it automatically sets an appropriate `EntityState` to each entity whenever an entity is created, modified, or deleted.

Insert Data

Use the `DbSet.Add` method to add a new entity to a context (instance of `DbContext`), which will insert a new record in the database when you call the `SaveChanges()` method.

```
using (var context = new SchoolDBEntities())
{
    var std = new Student()
    {
        FirstName = "Bill",
        LastName = "Gates"
    };
    context.Students.Add(std);

    context.SaveChanges();
}
```

In the above example, `context.Students.Add(std)` adds a newly created instance of the `Student` entity to a context with `Added` EntityState. The `context.SaveChanges()` method builds and executes the following INSERT statement to the database.

```
exec sp_executesql N'INSERT [dbo].[Students]([FirstName], [LastName])
VALUES (@0, @1)
SELECT [StudentId]
FROM [dbo].[Students]
WHERE @@ROWCOUNT > 0 AND [StudentId] = scope_identity()',N
''@0 nvarchar(max) ,@1 nvarchar(max) ',@0=N'Bill',@1=N'Gates'
go
```

Updating Data

In the connected scenario, EF API keeps track of all the entities retrieved using a context. Therefore, when you edit entity data, EF automatically marks `EntityState` to `Modified`, which results in an updated statement in the database when you call the `SaveChanges()` method.

```
using (var context = new SchoolDBEntities())
{
    var std = context.Students.First<Student>();
    std.FirstName = "Steve";
    context.SaveChanges();
}
```

In the above example, we retrieve the first student from the database using `context.Students.First<student>()`. As soon as we modify the `FirstName`, the context sets its `EntityState` to `Modified` because of the modification performed in the scope of the `DbContext` instance (context). So, when we call the `SaveChanges()` method, it builds and executes the following Update statement in the database.

```
exec sp_executesql N'UPDATE [dbo].[Students]
SET [FirstName] = @0
WHERE ([StudentId] = @1)',
N'@0 nvarchar(max) ,@1 int',@0=N'Steve',@1=2
Go
```

In an update statement, EF API includes the properties with modified values, other properties being ignored. In the above example, only the `FirstName` property was edited, so an update statement includes only the `FirstName` column.

Deleting Data

Use the `DbSet.Remove()` method to delete a record in the database table.

```
using (var context = new SchoolDBEntities())
{
    var std = context.Students.First<Student>();
    context.Students.Remove(std);

    context.SaveChanges();
}
```

In the above example, `context.Students.Remove(std)` marks the `std` entity object as `Deleted`. Therefore, EF will build and execute the following DELETE statement in the database.

```
exec sp_executesql N'DELETE [dbo].[Students]
WHERE ([StudentId] = @0)',N'@0 int',@0=1
Go
```

Thus, it is very easy to add, update, or delete data in Entity Framework 6.x in the connected scenario.

Further Reading

> [Saving data in the disconnected scenario in EF Core](#).

[Download EF 6 DB-First Demo Project from Github](#)

[‹ 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

> 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