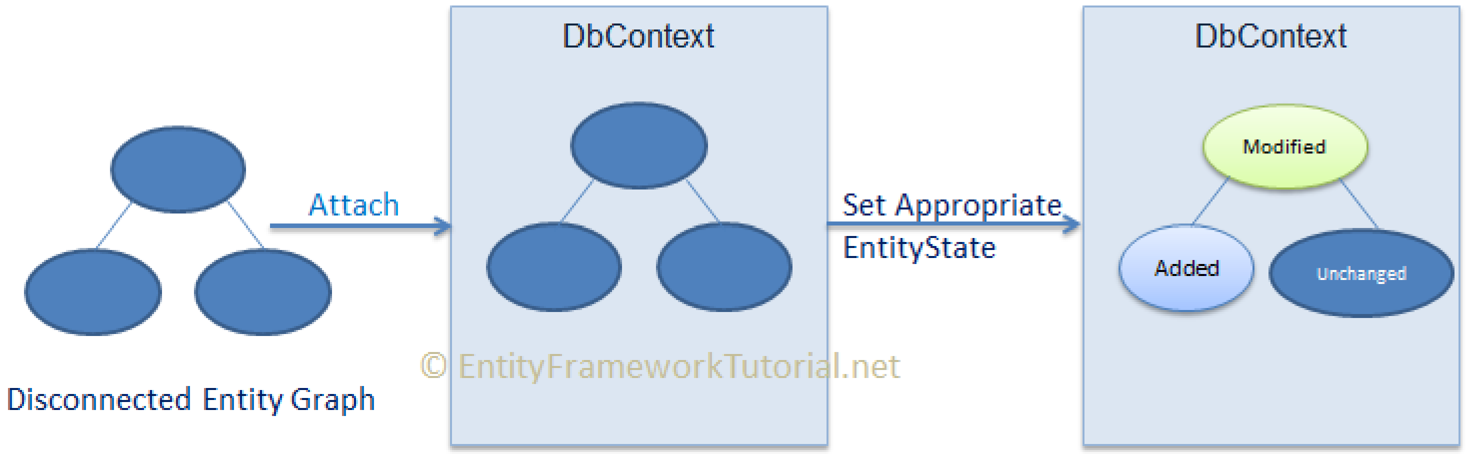
< Previous Next >

Methods to Attach Disconnected Entities in EF 6

Here you will learn about different methods in Entity Framework 6.x that attach disconnected entity graphs to a context.

Saving an entity in the disconnected scenario is different than in the connected scenario. There are two things we need to do when we get a disconnected entity graph or even a single disconnected entity. First, we need to attach entities with the new context instance and make the context aware about these entities. Second, set an appropriate `EntityState` to each entity manually because the instance doesn't know anything about the operations performed on the disconnected entities, so it cannot apply the appropriate `EntityState` automatically.

The following figure illustrates this process.


Disconnected Entity Graph
Attach
DbContext
Set Appropriate EntityState
DbContext
Modified
Added
Unchanged
© EntityFrameworkTutorial.net

Entity Framework provides the following methods that attach disconnected entities to a context and also set the `EntityState` to each entity in an entity graph.

- DbContext.Entry()
- DbSet.Add()
- DbSet.Attach()

DbContext.Entry()

The `Entry()` method of `DbContext` class returns an instance of `DbEntityEntry` for the specified entity. The `DbEntityEntry` object provides various information about the specified entity and also the ability to perform an action on the entity. Most importantly, we can change the `EntityState` of the specified entity using `State` property, as shown below.

```
context.Entry(entity).state = EntityState.Added/Modified/Deleted
```

The `Entry` method attaches an entire entity graph to a context with the specified state to a parent entity and also sets a different `EntityState` to other entities. Consider the following example.

```
var student = new Student() { //Root entity (empty key)
        StudentName = "Bill",
        StandardId = 1,
        Standard = new Standard()   //Child entity (with key value)
                    {
                        StandardId = 1,
                        StandardName = "Grade 1"
                    },
        Courses = new List<Course>() {
            new Course(){  CourseName = "Machine Language" }, //Child entity (empty key)
            new Course(){  CourseId = 2 } //Child entity (with key value)
        }
    };

using (var context = new SchoolDBEntities())
{
    context.Entry(student).State = EntityState.Added;

    foreach (var entity in context.ChangeTracker.Entries()){
        Console.WriteLine("{0}: {1}", entity.Entity.GetType().Name, entity.State);
    }
}
```

Output:

```
Student: Added
Standard: Added
Course: Added
Course: Added
```

In the above example, the `Studnet` entity graph includes the `Standard` and `Course` entities. `context.Entry(student).State = EntityState.Added;` sets the Added state to the parent entity as well as to all other child entities irrespective of whether a child entity contains an empty key value or not. Thus, it is recommended to use the `Entry()` method carefully.

The following table lists the behaviour of the `Entry()` method.

Parent Entity State	**Entity State of child entities**
Added	Added
Modified	Unchanged
Deleted	All child entities will be null

DbSet.Add()

The `DbSet.Add()` method attaches the entire entity graph to a context and automatically applies the Added state to all entities.

```
//disconnected entity graph
Student disconnectedStudent = new Student() { StudentName = "New Student" };
disconnectedStudent.StudentAddress = new StudentAddress() { Address1 = "Address", City = "City1" };

using (var context = new SchoolDBEntities())
{
    context.Students.Add(disconnectedStudent);

    // get DbEntityEntry instance to check the EntityState of specified entity
    var studentEntry = context.Entry(disconnectedStudent);
    var addressEntry = context.Entry(disconnectedStudent.StudentAddress);

    Console.WriteLine("Student: {0}", studentEntry.State);
    Console.WriteLine("StudentAddress: {0}", addressEntry.State);
}
```

Output:

```
Student: Added
StudentAddress: Added
```

The `DbSet.Add()` method attaches the entire entity graph to a context with the Added state to each entity. Calling `context.Savechanges()` will execute the INSERT command for all the entities, which will insert new records in the appropriate database table.

DbSet.Attach()

The `DbSet.Attach()` method attaches an entire entity graph to the new context with the Unchanged entity state.

```
//disconnected entity graph
Student disconnectedStudent = new Student() { StudentName = "New Student" };
disconnectedStudent.StudentAddress = new StudentAddress() { Address1 = "Address", City = "City1" };

using (var context = new SchoolDBEntities())
{
    context.Students.Attach(disconnectedStudent);

    // get DbEntityEntry instance to check the EntityState of specified entity
    var studentEntry = context.Entry(disconnectedStudent);
    var addressEntry = context.Entry(disconnectedStudent.StudentAddress);

    Console.WriteLine("Student: {0}",studentEntry.State);
    Console.WriteLine("StudentAddress: {0}",addressEntry.State);
}
```

Output:

```
Student: Unchanged
StudentAddress: Unchanged
```

Download EF 6 DB-First Demo Project from Github

< Previous

Next >