Tracking vs No-Tracking Queries

In Entity Framework Core, tracking behavior controls whether to keep track or not of information about an entity instance when it is changed. If an entity is tracked, any changes detected in the entity will be persisted to the database when `SaveChanges` is called.

- EF Core will also fix up navigation properties between the entities in a tracking query result and the entities that are in the change tracker.
- Keyless entity types are never tracked, so wherever entity types are mentioned, it refers to entity types that have a key defined.

Tracking Queries

By default, queries that return entity types are tracking. This means you can make changes to those entity instances and have those changes persisted by `SaveChanges()`. In the following example, the change to the author address will be detected and persisted in the database during `SaveChanges()`.

```
using (var context = new EntityContext())
{
    var author = context.Authors
        .Where(c => c.AuthorId == 2)
        .FirstOrDefault();

     author.Address = "43 rue St. Laurent";
     context.SaveChanges();
}
```

No-Tracking Queries

No tracking query executes quickly because there is no need to set up change tracking information. It is useful when the results are used in a read-only scenario.

You can convert a query to a no-tracking query by using the `AsNoTracking()` method.

AsNoTracking

The `AsNoTracking()` method returns a new query where the change tracker will not track any of the entities that are returned. If the entity instances are modified, this will not be detected by the change tracker, and `SaveChanges()` will not persist those changes to the database.

```
using (var context = new EntityContext())
{
     var authors = context.Authors
         .AsNoTracking().ToList();
}
```

You can also change the default tracking behavior at the context instance level.

```
using (var context = new EntityContext())
{
    context.ChangeTracker.QueryTrackingBehavior = QueryTrackingBehavior.NoTracking;
    var authors = context.Authors.ToList();
}
```

Tracking and Custom Projections

By default, EF Core will track entity types contained in the result, even if the result type of the query isn't an entity type. In the following query, which returns an anonymous type, the instances of `Author` in the result set will be tracked.

```
var author = context.Authors
    .Select(a =>
        new
        {
            Author = a,
            BookCount = a.Books.Count()
        });
```

EF Core will also track entity types contained in the result if the result set contains entity types coming out from LINQ composition.

```
var author = context.Authors
    .Select(a =>
        new
        {
            Author = a,
            Book = a.Books.OrderBy(b => b.Title).FirstOrDefault()
        });
```

If the result set doesn't contain any entity types, then EF Core will not track them. In the following query, we return an anonymous type with some of the values from the entity, so there are no tracked entities coming out of the query.

```
var author = context.Authors
    .Select(a =>
        new
        {
            Id = a.AuthorId,
            Name = a.Name
        });
```

EF Core supports doing client evaluation in the top-level projection. If EF Core materializes an entity instance for client evaluation, it will be tracked. In this example, we're passing the `FirstName` and `LastName` of author entities to the client method `CombineNames`, EF Core will track the blog instances too.

```
var author = context.Authors
    .Where(a => a.AuthorId == 1)
    .Select(au => new
    {
        FullName = CombineNames(au.FirstName, au.LastName)
    }).FirstOrDefault();
```

In the following example, a helper method is used to combine the first name and last name from a SQL Server database.

```
private static string CombineNames(string firstName, string lastName)
{
    return firstName + " " + lastName;
}
```

Got any EF Core Advanced Topics Question?

Ask any EF Core Advanced Topics Questions and Get Instant Answers from ChatGPT AI:

Ask any EF Core Advanced Topics question...

ChatGPT answer me!