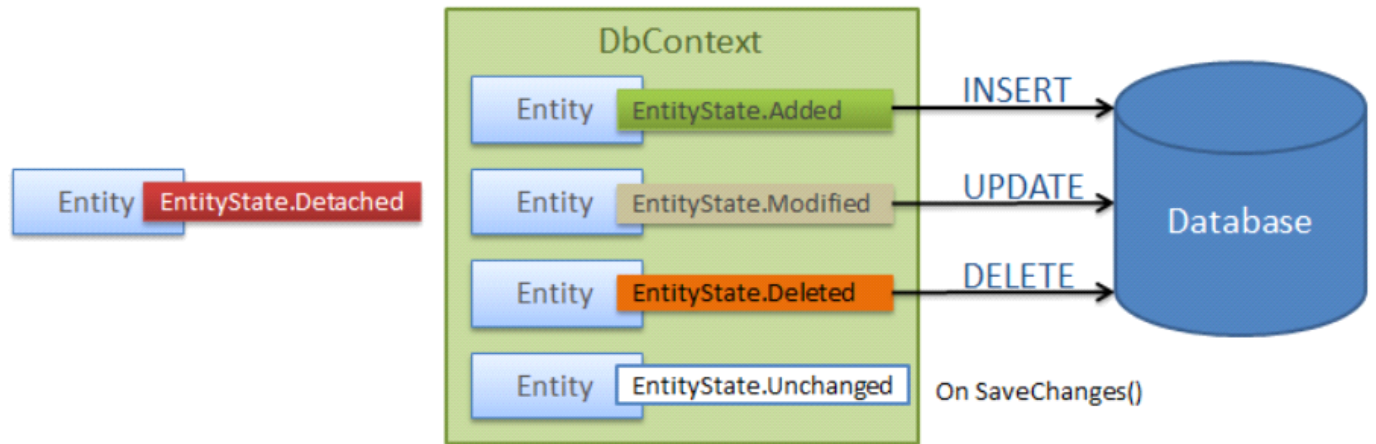


第39期-添加数据的多种方案

2019年6月17日 21:53

每个上下文实例都有一个 `ChangeTracker`，它负责跟踪需要写入数据库的更改。更改实体类的实例时，这些更改会记录在 `ChangeTracker` 中，然后在调用 `SaveChanges` 时被写入数据库。此数据库提供程序负责将更改转换为特定于数据库的操作（例如，关系数据库的 `INSERT`、`UPDATE` 和 `DELETE` 命令）。



ChangeTracker

提供对上下文所有实体状态的跟踪信息。

EntityEntry

表示给定实体的跟踪信息，可通过 `ChangeTracker.Entries`、`DbContext.Entry` 和 `DbSet.Entry` 获取。

EntityState

被跟踪实体的状态

状态演练

查询一个对象，观察跟踪和未跟踪情况下对象的状态。

```
_context.ChangeTracker.QueryTrackingBehavior = QueryTrackingBehavior.NoTracking;  
Blog blog = _context.Find<Blog>(1) ;  
EntityEntry entityEntry = _context.Entry(blog);
```

添加数据

添加数据是指：数据库无记录，添加新的记录到数据库。

DbContext.Add() 或 DbSet.Add()

开始跟踪给定实体，将尚未被跟踪的任何实体更改为已添加状态。Detached=>Added

更多重载：AddAsync、AddRange 和 AddRangeAsync

<https://www.cnblogs.com/CreateMyself/p/9043744.html>

DbContext.Attach() 或 DbSet.Attach()

开始在上下文中跟踪给定的实体，将执行导航属性的递归搜索以查找尚未被上下文跟踪的可到达实体。这些实体也将开始被上下文跟踪。通俗点来讲就是查找所有 Detached 游离状态的对象图，将其设置为被上下文跟踪的状态，跟踪起来，跟踪成何种状态，有以下两种情况。

如果可访问实体的主键值已设置，则将以“未更改”状态跟踪它。Detached=>Unchanged

如果未设置主键值，则将在“已添加”状态下跟踪它。Detached=>Added

批处理此操作：AttachRange

```
Blog blog = new Blog() { Name = "zeroblog", Url = "www.xcode.me" };
```

```
EntityEntry entityEntry = _context.Entry(blog);  
Console.WriteLine(entityEntry.State);
```

```
_context.Attach(blog);  
Console.WriteLine(entityEntry.State);
```

```
_context.SaveChanges();  
Console.WriteLine(entityEntry.State);
```

添加新实体的关系图

```
var blog = new Blog  
{  
    Name = "zerodo",  
    Url = "www.xcode.me",  
    Posts = new List<Post>  
    {  
        new Post { Title = "Intro to C#" },  
        new Post { Title = "Intro to VB.NET" },  
        new Post { Title = "Intro to F#" }  
    }  
};
```

```
_context.Blogs.Add(blog);  
//_context.Blogs.Attach(blog);  
_context.SaveChanges();
```

注意无法插入数据的情况

```
var blog = new Blog
{
    Name = "zerodo",
    Url = "www.xcode.me",
};

var posts = new List<Post>
{
    new Post { Title = "Intro to C#", Blog=blog },
    new Post { Title = "Intro to F#", Blog=blog }
};

await _context.AddAsync(blog);

await _context.SaveChangesAsync();
```

添加相关实体

```
var blog = _context.Blogs.Include(b => b.Posts).First();

var post = new Post { Title = "Intro to EF Core" };
blog.Posts.Add(post);

_context.SaveChanges();
```

关于自增主键数据的强行插入

```
var blog1 = new Blog { BlogId = 33, Name = "zerodo1", Url = "www.xcode1.me", };
var blog2 = new Blog { BlogId = 44, Name = "zerodo2", Url = "www.xcode2.me", };
var blog3 = new Blog { BlogId = 55, Name = "zerodo3", Url = "www.xcode3.me", };

_context.Add(blog1);
_context.Add(blog2);
_context.Add(blog3);

_context.Database.OpenConnection();

try
{
    _context.Database.ExecuteSqlCommand("SET IDENTITY_INSERT [Blogs] ON");
    await _context.SaveChangesAsync();
    _context.Database.ExecuteSqlCommand("SET IDENTITY_INSERT [Blogs] OFF");
}
finally
{
    _context.Database.CloseConnection();
}
```

```
}
```

允许将显式值插入到表的标识列中。

SET IDENTITY INSERT

当多次调用插入数据时，最后执行 **SaveChanges** 保存时，将生成批处理语句，而不像之前版本的EF那样，生成多条 SQL 语句，相比之下，EF Core 性能更高。

```
var blog1 = new Blog { Name = "zerodo1", Url = "www.xcode1.me", };  
var blog2 = new Blog { Name = "zerodo2", Url = "www.xcode2.me", };  
var blog3 = new Blog { Name = "zerodo3", Url = "www.xcode3.me", };
```

```
_context.Add(blog1);  
_context.Add(blog2);  
_context.Add(blog3);
```

```
_context.SaveChanges();
```

What is Batching of Statement in Entity Framework Core?

Entity Framework Core 批处理语句

使用存储过程或者原生SQL保存数据

```
_context.Database.ExecuteSqlCommand("SQL");
```