

第40期-更新数据的多种方案

2019年6月17日 21:53

更新数据

数据库中有主键所对应的记录，修改实体到 Modified 状态，调用 SaveChanges 时，生成 Update 语句。

更新已跟踪实体的数据

当实体由 DbContext 获取，且默认为已为跟踪状态，当我们改变实体的属性值时，跟踪器将自动将实体的状态修改为 Modified 状态。

```
var blog = context.Blogs.First();
blog.Url = "www.xcode.me";
context.SaveChanges();
```

更新未跟踪实体的数据

对于未被跟踪的断开实体，可通过以下三种方案更新数据。

方案一：显式设置 EntityState 状态

```
context.Entry(blog).State = EntityState.Modified;
context.SaveChanges();
```

方案二：在 DbContext 或 DbSet 上使用 Update 方法

```
context.Update(blog);
context.SaveChanges();
```

Update 方法与设置 EntityState 方案一样，会将实体状态设置为 Modified 状态。由于跟踪器没有任何方法来识别哪些属性值已经更改，所以生成的 UPDATE 语句会更新所有字段属性。

Update 方法与显示设置设置 EntityState 不同的是，update 方法会修改相关实体（如 Blog 的 Posts 导航属性）的状态为已修改，从而会为每个实体生成 UPDATE 语句。如果相关实体没有对应的键值，就会标记为 Added 状态，生成一条 Insert 语句。

方案三：在 DbContext 或 DbSet 上使用 Attach 方法，然后遍历对象图，设置各个属性的状态。

当在**已设置键值**的实体上使用 Attach 方法时，它的状态将被设置为 Unchanged 未修改状态，这将导致根本不会生成任何 SQL 语句。

当在**没有键值**的实体上使用 Attach 方法时，一个 Detached 游离状态的实体将被标记为 Added 已添加状态。

```
context.Entry(entity).IsKeySet
```

不管怎么说，调用 `Attach` 方法后，该实体将被跟踪器跟踪，我们可以通知跟踪器哪些属性被修改，生成正确的 `UPDATE` 语句，而不是更新所有字段，这也许会获得更好的性能。

```
var blog = new Blog
{
    BlogId=1,
    Name="ZEROBLOG",
    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.Attach(blog);
_context.Entry(blog).Property("Name").IsModified = true;

_context.SaveChanges();
```

以上代码，生成的 `UPDATE` 语句只会更新 `Name` 属性。

如果实体不使用自动生成的键，则应用程序必须确定是应插入实体还是应更新实体：

```
public static void InsertOrUpdate(BloggingContext context, Blog blog)
{
    var existingBlog = context.Blogs.Find(blog.BlogId);
    if (existingBlog == null)
    {
        context.Add(blog);
    }
    else
    {
        context.Entry(existingBlog).CurrentValues.SetValues(blog);
    }

    context.SaveChanges();
}
```

TrackGraph 跟踪对象图

在内部，`Add`、`Attach` 和 `Update` 使用图形遍历，为每个实体就是否应将其标记为 `Added`（若要插入）、`Modified`（若要更新）、`Unchanged`（不执行任何操作）或 `Deleted`（若要删除）作出决定。此机制是通过 `TrackGraph` API 公开的。

`TrackGraph` API 提供了对对象图中各个实体的访问，并允许您对每个实体分别执行定制代码。这在处理这种由不同对象的相关实体的复杂对象图的场景中非常有用。下面的示例复制了一个场景，其中对象图是在 `Context` 之外构

造的。

```
var blog = new Blog
{
    BlogId = 1,
    Name = "zerodo",
    Url = "www.xcode.me",
    Posts = new List<Post>
    {
        new Post { PostId=1, Title = "Intro to C#",Content="AAA" },
        new Post { PostId=2, Title = "Intro to VB.NET",Content="BBB" },
        new Post { PostId=3, Title = "Intro to F#",Content="CCC"}
    }
};

context.ChangeTracker.TrackGraph(blog, e =>
{
    if (e.Entry.Entity is Blog)
    {
        e.Entry.State = EntityState.Unchanged;
    }

    if (e.Entry.Entity is Post)
    {
        e.Entry.State = EntityState.Unchanged;
        context.Entry(e.Entry.Entity as Post).Property("Title").IsModified = true;
    }
});
context.SaveChanges();
```

以上示例，无论如何修改 Blog 或者 Post 的属性值，只有 Post 的 Title 字段会被更新。

更改关系

```
var blog = new Blog { Name = "zeroblog", Url = "www.xcode.me" };
var post = context.Posts.First();
post.Blog = blog;
context.SaveChanges();
```

更改关系：可设置导航属性（不存在则创建），也可设置外键的值。