

# 第47期-变更追踪策略与原理

2019年7月18日 20:17

当利用上下文对实体进行增删改查时，EF是如何生成正确 SQL 命令的？又是如何知道哪些属性值发生了改变呢？答案是利用变更追踪机制。

## Snapshot 快照式变更追踪（默认）

### DetectChanges 方法的用途

下列的代码中我们查询了一个 blog 实体，同时修改 Url 属性，最终对 Url 的改变被保存到了数据库。

```
Blog blog = context.Blogs.Find(1);
blog.Url = "www.xcode.me";
context.SaveChanges();
```

当从数据库进行查询数据时，上下文便捕获了每个实体属性的快照（数据库值，当前值，原始值），当调用 SaveChanges 时，在内部会自动调用 DetectChanges 方法，此方法将扫描上下文中所有实体，并比较当前属性值和存储在快照中的原始属性值，如果被找到的属性值发生了改变，此时EF将会与数据库进行交互，进行数据更新。

```
EntityEntry<Blog> entry = context.Entry(blog);
```

```
查询当前数据库中的值: string database = entry.GetDatabaseValues().GetValue<string>(propName);
首次从数据库中查询的值: string original = entry.OriginalValues.GetValue<string>(propName);
获取此实体的当前属性值: string current = entry.CurrentValues.GetValue<string>(propName);
```

默认情况下 DetectChanges 方法会在在很多场景下自动调用，并不需要手动去调用，比如以下场景就会自动调用。

会导致自动调用 DetectChanges 方法：Find、Local、Remove、Add、Update、Attach、SaveChanges 和 Entry 等。

### 关闭自动调用 DetectChanges 方法

```
context.ChangeTracker.AutoDetectChangesEnabled = false;
```

```
Blog blog = context.Blogs.Find(1);
Console.WriteLine(context.Entry(blog).State);
```

```
blog.Url = "www.xcode.me";
Console.WriteLine(context.Entry(blog).State);
```

```
context.ChangeTracker.DetectChanges();
Console.WriteLine(context.Entry(blog).State);
```

```
context.SaveChanges();
Console.WriteLine(context.Entry(blog).State);
```

## 使用属性变更通知接口

实现 INotifyPropertyChanged、INotifyPropertyChanging 和 INotifyCollectionChanged 接口。

谈谈 INotifyPropertyChanged 的实现

ObservableCollection 和 ObservableCollection 实现了 INotifyCollectionChanged 接口。

可使用 AOP 扩展拦截方法或属性的调用：Unity.Interception 动态拦截。