

第48期-属性变更事件追踪策略

2019年7月20日 23:48

如果上下文中加载了大量的实体准备修改，使用 DetectChanges 会造成性能开销，这是因为当任何一个实体变更时，都要将属性值与快照的原始值进行比较，从而生成快照中实体的状态，为了随时保持同步状态，EF提供的很多方法默认都会自动调用 DetectChanges 将扫描变更追踪器中的所有实体快照进行状态同步。

```
context.Entry(blog).CurrentValues.SetValues(new { Url = "v.xcode.me" });
```

```
var og = (int) context.Entry(me).Property(nameof(EmployeeNumber)).OriginalValue;
```

如果一个实体类实现了属性变更通知接口，则当该类中属性的值发生变化时，通过事件通知任何订阅者，这样就不会全部扫描实体快照了，而是针对变更的实体属性进行主动通知，减少额外的快照扫描开支，EF CORE 的变更跟踪器会订阅实现了属性变更通知接口的所有实体，只针对的变化的属性进行状态同步。

使用 ChangeTrackingStrategy 枚举指定变更跟踪策略

```
public enum ChangeTrackingStrategy
{
    Snapshot = 0,
    ChangedNotifications = 1,
    ChangingAndChangedNotifications = 2,
    ChangingAndChangedNotificationsWithOriginalValues = 3
}
```

```
modelBuilder.Entity<Blog>
().HasChangeTrackingStrategy(ChangeTrackingStrategy.Snapshot);
modelBuilder.HasChangeTrackingStrategy(ChangeTrackingStrategy.Snapshot);
```

关于变更跟踪器提供的方法

```
public event EventHandler<EntityTrackedEventArgs> Tracked;
public event EventHandler<EntityStateChangedEventArgs> StateChanged;

public virtual bool LazyLoadingEnabled { get; set; }
public virtual bool AutoDetectChangesEnabled { get; set; }
public virtual void AcceptAllChanges();
public virtual void DetectChanges();
public virtual bool HasChanges();
```

调用 AcceptAllChanges 方法将所有变更状态的实体更改为 Unchanged 状态。

```
context.ChangeTracker.AcceptAllChanges();
```

将更改成功发送到数据库后是否调用 `AcceptAllChanges` 方法。

```
context.SaveChanges(acceptAllChangesOnSuccess: false);
```

将查询结果加载到关联的上下文中

```
context.Blogs.Where(b => b.Url.Length > 10).Load()
```

这相当于调用 `ToList` 方法后将结果置于上下文内存，但不需要使用结果，没有实际创建 `List` 的开销。

从数据库重新装载选定实体

从数据库重新加载实体，使用数据库中的值覆盖任何属性值，调用此方法后，如果数据库有该实体主键锁对应的记录，实体将被重新装载，并置于 `Unchanged` 状态，如果不存在该实体，实体将是 `Detached` 状态。

```
context.Entry(blog).Reload();  
context.Entry(blog).ReloadAsync();
```

可用查询当前上下文内存中已经加载的实体对象。

```
LocalView<Blog> blogs= context.Blogs.Local;
```

获取一个 `LocalView <TEntity>`，它表示此集合中所有已添加、未更改和已修改实体的本地视图。当在上下文中添加或删除实体时，此本地视图将保持同步。同样，添加到本地视图或从本地视图中删除的实体将自动添加到上下文中或从上下文中删除。

使用 `ToObservableCollection` 方法可将 `LocalView` 集合转换成观察者集合，以便于进行事件跟踪。

```
var observableBlogs = context.Blogs.Local.ToObservableCollection();  
observableBlogs.CollectionChanged += (s, e) => { };  
observableBlogs.Add(new Blog { Url = "video.xcode.me" });
```