EF4: Difference between POCO , Self Tracking Entities , POCO Proxies

Asked 9 years, 5 months ago Active 8 years ago Viewed 4k times

5

Can someone point me the difference between POCO , Self Tracking Entities , POCO Proxies?

Actually, I am working Entity Framework 4.0 and POCO(Repository Pattern) and whenever I do some changes in the POCO and call ObjectContext.Savechanges then it reflects to the DB. My question is,

1. How does the Context persist the change to the DB since it is not tracked?
2. Does the Context generates the tracking info on the fly for POCO?

Sample Code I am using,

```
        IEFRepository<Category> catRepository = new EFRepository<Category>();
        Category c = catRepository.FindOne<Category>(x => x.Name == "Paper");

        c.Name = "Paper";
        catRepository.SaveChanges(System.Data.Objects.SaveOptions.None);
```

entity-framework-4 poco self-tracking-entities

asked Oct 19 '10 at 9:07

paragy
6,625 ● 3 ● 23 ● 38

1 Answer

Active Oldest Votes

6

Self tracking entities are not POCOs. On the contrary, they are very much persistence-aware. More so than `EntityObject` entities, even. What makes them unique is the changes can be tracked even when they are not attached to an `ObjectContext`.

"Pure" POCOs, as you say, make change tracking difficult. Really, the only thing you can do is compare snapshots of the object. The object context has a `DetectChanges` method for this.

With a pseudo-POCO proxy, what you really have is a type which looks (almost) like a POCO at compile time and like a non-POCO at runtime. I say "almost" because at runtime you will get an instance which is a subtype of the compile-time type. Because of this, any properties for which you want to track changes must be non-private and virtual. Similar restrictions apply to lazy loading. You can read more about this in this series of articles on the ADO.NET team blog.

answered Oct 19 '10 at 14:28

Craig Stuntz

122k ● 12 ● 241 ● 266

1 Thanks Craig, and answer for second part of my question is ProxyCreationEnabled should be false. As working with POCO Proxies, EF generates proxy types on the fly using the AssemblyBuilder and TypeBuilder in the .NET framework. – paragy Oct 20 '10 at 11:43