EntityFramework Eager Load all Navigation Properties

Asked 6 years, 7 months ago Active 3 years, 5 months ago Viewed 17k times

15

8

I'm using the Repository pattern with DI and IoC.

I have created a function in my Repository:

```
T EagerGetById<T>(Guid id, string include) where T : class
{
    return _dbContext.Set<T>().Include(include).Find(id);
}
```

This will eagerly load one navigation property in my entity right.

But if my entity looks like this:

```
public class Blog : PrimaryKey
{
    public Author Author {get;set;}
    public ICollection<Post> Posts {get;set;}
}
```

How would I get eager loading for `Author` and `Posts`? Would I literally have to do:

```
_dbContext.Set<T>().Include("Author").Include("Posts").Find(id);
```

inevitably producing a function like this:

```
T EagerGetById<T>(Guid id, string include, string include2, string include3) where T :
class
{
    return _dbContext.Set<T>
().Include(include).Include(include2).Include(include3).Find(id);
}
```

Because that would be really inefficient for a `Generic` Repository!

c# entity-framework

edited Nov 24 '16 at 10:45
wonea
4,859 ● 15 ● 67 ● 130

asked Sep 14 '13 at 18:07
Callum Linington
12.4k ● 9 ● 57 ● 118

2 Answers

Active Oldest Votes

25

If you don't want to use strings, you can also do the same for any N number of includes by using an expression which returns the navigation properties to be eager loaded. (original source here)

```
public IQueryable<TEntity> GetAllIncluding(params Expression<Func<TEntity, object>>[]
includeProperties)
{
   IQueryable<TEntity> queryable = GetAll();
   foreach (Expression<Func<TEntity, object>> includeProperty in includeProperties)
   {
      queryable = queryable.Include<TEntity, object>(includeProperty);
   }

   return queryable;
}
```

edited Sep 14 '13 at 23:37

answered Sep 14 '13 at 18:34
Daniel Auger
11.8k ● 5 ● 44 ● 71

I was actually thinking about doing that, but is it easy in Repository pattern with DI and IoC?? – Callum Linington Sep 14 '13 at 18:38

On 2nd glance... context.Configuration.ProxyCreationEnabled = false; doesn't force eager loading. It just turns of proxy creation. You'd still need to do the include. I'm editing it out of the answer. – Daniel Auger Sep 14 '13 at 18:44

does that make sense.... what is the point in enabling proxy creation, what benefits does it give?! – Callum Linington Sep 14 '13 at 21:26

Proxy creation enables lazy loading of virtual properties. Essentially EF will put "stand in classes" for the navigation properties that hit the database when accessed. If proxy generation isn't enabled, those the child objects that haven't been "included" in the query will appear null and EF won't hit the db behind the scenes to do lazy loading. – Daniel Auger Sep 14 '13 at 23:22

I'm having a difficult time understanding this implementation. Can you give a usage example? – Xipooo Jan 6 '16 at 22:10

0

If you need all navigation properties, you have no choice but to read all of them from the database. You either `Include` them in your query, or read them in advance to the DbSet's local data.

If you want to pass multiple includes to your method, just define it like this:

```
T EagerGetById<T>(Guid id, params string[] includes)
```

Your users will be able to call `EagerGetById(id, "inc1", "inc2", ...)`

Inside your method, just call `Include` for every element in the `includes` array.

You should ready about the `params` keyword

edited Sep 14 '13 at 18:25

answered Sep 14 '13 at 18:18
zmbq
33.8k ● 12 ● 68 ● 138

See edits, specifically the new kind of function that would have to be created for this... – Callum Linington Sep 14 '13 at 18:21

Oh, that's not inefficient, it's just ugly. I'll edited my response. – zmbq Sep 14 '13 at 18:23

i know all about params.. still didnt really answer the question! Sorry dude, this hasn't cut it! – Callum Linington Sep 14 '13 at 18:37

For the record, @zmbq 's answer was just fine. It seems you didn't get it. The accepted answer is almost literally the same thing. It's just, arguably, more 'clean' to use property references instead of property names. – Dinerdo Mar 21 '19 at 19:44