Entity Framework - Include Multiple Levels of Properties

Asked 7 years, 3 months ago Active 10 months ago Viewed 194k times

321

68

The Include() method works quite well for Lists on objects. But what if I need to go two levels deep? For example, the method below will return ApplicationServers with the included properties shown here. However, ApplicationsWithOverrideGroup is another container that holds other complex objects. Can I do an Include() on that property as well? Or how can I get that property to fully load?

As it stands now, this method:

```
public IEnumerable<ApplicationServer> GetAll()
{
    return this.Database.ApplicationServers
        .Include(x => x.ApplicationsWithOverrideGroup)
        .Include(x => x.ApplicationWithGroupToForceInstallList)
        .Include(x => x.CustomVariableGroups)
        .ToList();
}
```

Will populate only the Enabled property (below) and not the Application or CustomVariableGroup properties (below). How do I make this happen?

```
public class ApplicationWithOverrideVariableGroup : EntityBase
{
    public bool Enabled { get; set; }
    public Application Application { get; set; }
    public CustomVariableGroup CustomVariableGroup { get; set; }
}
```

c# entity-framework

asked May 30 '12 at 19:07
Bob Horn
20.4k 23 88 179

1 @BobHorn, I have the same issue.. In my case, the nesting goes deep down multiple layers, i managed to do a include you pointed out. In the SQL which got generated, i could see all columns are returning with different alias name as c1,c2 something like that. My question is , how i can form a nested DTO collection out of all my includes:(.. May be you can take the above example itself, in that we are returning all the columns without any custom DTO (which itself is collection of DTO's) – TechQuery Nov 20 '15 at 21:32

7 Answers

625

For EF 6

```
using System.Data.Entity;

query.Include(x => x.Collection.Select(y => y.Property))
```

See Remarks for more examples.

Make sure to add `using System.Data.Entity;` to get the version of `Include` that takes in a lambda.

For EF Core

Use the new method `ThenInclude`

```
query.Include(x => x.Collection)
     .ThenInclude(x => x.Property);
```

edited Nov 9 '18 at 8:13
Richard Garside 42.3k 9 66 78

answered May 30 '12 at 19:38
Diego Torres 14.4k 4 30 45

1 I can't do Include() on ApplicationsWithOverrideGroup. It doesn't show up in intellisense. – Bob Horn May 30 '12 at 19:42

I can't use your edit because ApplicationsWithOverrideGroup is a List. Application is a property on each item in the list, not on the list itself. – Bob Horn May 30 '12 at 19:45

1 Ahhhh, but that link you provided seems to provide the answer. Let me try this: To include a collection and then a collection one level down:

OJ Raqueño Jul 28 '13 at 3:15

4 @Adeem you need to call `Include` for each property: `Db.States.Include(state => state.Cities.Select(city => city.Customers).Include(state => state.Cities.Select(city => city.Vendors)` – Diego Torres Feb 24 '16 at 15:13

59

If I understand you correctly you are asking about including nested properties. If so :

```
.Include(x => x.ApplicationsWithOverrideGroup.NestedProp)
```

or

```
.Include("ApplicationsWithOverrideGroup.NestedProp")
```

or

```
.Include($"{nameof(ApplicationsWithOverrideGroup)}.{nameof(NestedProp)}")
```

edited May 6 '17 at 14:47
tchelidze
5,905 1 20 36

answered May 30 '12 at 19:37
Judo
4,389 3 20 33

4 Thanks, I can try that. I was hoping to be able to keep things strongly typed and avoid string literals. But if that's how it has to be done... – Bob Horn May 30 '12 at 19:42

You were close. I may not have been clear that ApplicationsWithOverrideGroup was a list. Thanks for helping! – Bob Horn May 30 '12 at 19:51

@Judo, I have the same issue.. In my case, the nesting goes deep down multiple layers, i managed to do a include you pointed out. In the SQL which got generated, i could see all columns are returning with different alias name as c1,c2 something like that. My question is , how i can form a nested DTO collection out of all my includes:(.. May be you can take the above example itself, in that we are returning all the columns without any custom DTO (which itself is collection of DTO's) – TechQuery Nov 20 '15 at 21:32

1 Remember to include **System.Data.Entity** in the usings. Otherwise Intellisense will only give you the `Include(string path)` version of the method. – AndreyWD May 29 '18 at 10:32

45

```
var blogs = context.Blogs
    .Include(blog => blog.Posts)
        .ThenInclude(post => post.Author)
        .ThenInclude(author => author.Photo)
    .ToList();
```

edited Sep 4 '18 at 13:48
Michael Freidgeim
15k 6 97 119

answered Jul 10 '16 at 3:13
thangcao
1,226 1 10 12

51 Looks like this is EF Core only – Chris Marisic Jul 25 '16 at 20:59

18 FYI: VS2017 the intellisense wasn't working for .ThenInclude. Just type it in how you think it should be and the error highlighting should go away. – JohnWrensby Apr 22 '17 at 20:29

2 I want to emphasize @JohnWrensby 's comment, the Intellisense can sometimes take especially long to handle these ThenInclude , this can be quite confusing for new users. I also had cases where the simple Include lambda expression was not handled properly, until you just type it and compile it, ignoring the "errors" shown in VS. – Pac0 Jun 28 '17 at 8:46

26

I made a little helper for Entity Framework 6 (.Net Core style), to include sub-entities in a nice way.

It is on NuGet now : Install-Package ThenInclude.EF6

```
using System.Data.Entity;

var thenInclude = context.One.Include(x => x.Twoes)
    .ThenInclude(x=> x.Threes)
    .ThenInclude(x=> x.Fours)
    .ThenInclude(x=> x.Fives)
    .ThenInclude(x => x.Sixes)
    .Include(x=> x.Other)
    .ToList();
```

The package is available on GitHub.

edited Sep 4 '18 at 14:09
Michael Freidgeim

answered Mar 17 '17 at 10:51
Lenny32

hi, i have an exception at runtime, cannot cast IncludableQueryable<observablecollection> to IncludableQueryable<genericcollection> – user2475096 Mar 29 '18 at 18:00

i am using db first and i have modified the tt file to get ObservableCollections for all my entities, any help is welcome . – user2475096 Mar 29 '18 at 18:01

2 @lenny32 anything to be aware of with this extension? – Aaron Hudon Jun 12 '18 at 4:10

18

I also had to use multiple includes and at 3rd level I needed multiple properties

```
(from e in context.JobCategorySet
                      where e.Id == id &&
                            e.AgencyId == agencyId
                      select e)
                      .Include(x => x.JobCategorySkillDetails)
                      .Include(x => x.Shifts.Select(r => r.Rate).Select(rt =>
rt.DurationType))
                      .Include(x => x.Shifts.Select(r => r.Rate).Select(rt =>
rt.RuleType))
                      .Include(x => x.Shifts.Select(r => r.Rate).Select(rt =>
rt.RateType))
                      .FirstOrDefaultAsync();
```

This may help someone :)

answered Jun 17 '17 at 12:56
dnxit
5,142 2 19 27

1 can this be done without repeating `.Include(x => x.Shifts.Select(r => r.Rate).Select(rt => rt......` – Multinerd Jan 10 '18 at 20:58

well it depends, how deep you wanna go – dnxit Jan 11 '18 at 7:34

More EFCore examples on MSDN show that you can do some quite complex things with `Include` and `ThenInclude`.

This is a good example of how complex you can get (this is all one statement!):

```
viewModel.Instructors = await _context.Instructors

      .Include(i => i.OfficeAssignment)

      .Include(i => i.CourseAssignments)
        .ThenInclude(i => i.Course)
            .ThenInclude(i => i.Enrollments)
                .ThenInclude(i => i.Student)

      .Include(i => i.CourseAssignments)
        .ThenInclude(i => i.Course)
            .ThenInclude(i => i.Department)

      .AsNoTracking()
      .OrderBy(i => i.LastName)
      .ToListAsync();
```

See how you can chain `Include` even after `ThenInclude` and it kind of 'resets' you back to the level of the top level entity (Instructors).

You can even repeat the same 'first level' collection (CourseAssignments) multiple times followed by separate `ThenIncludes` commands to get to different child entities.

Note your actual query must be tagged onto the end of the `Include` or `ThenIncludes` chain. The following does NOT work:

```
var query = _context.Instructors.AsQueryable();
query.Include(i => i.OfficeAssignment);

var first10Instructors = query.Take(10).ToArray();
```

Would strongly recommend you set up logging and make sure your queries aren't out of control if you're including more than one or two things. It's important to see how it actually works - and you'll notice each separate 'include' is typically a new query to avoid massive joins returning redundant data.

`AsNoTracking` can greatly speed things up if you're not intending on actually editing the entities and resaving.

answered Jan 16 '18 at 8:48
Simon_Weaver
79.5k 65 491 551

If you want lazy-loading stay tuned for EF Core 2.1 blogs.msdn.microsoft.com/dotnet/2018/02/02/… but if you just want to load more at the same level I think this is by design. I'm not sure what you're thinking - it doesn't require much extra to do this and it greatly reduces what comes back from the database. An entity may just have one or two 'same-level' things but it may also have 50 for a large project, being explicit makes your app much faster. – Simon_Weaver Feb 8 '18 at 23:06

3

Let me state it clearly that you can use the string overload to include nested levels regardless of the multiplicities of the corresponding relationships, if you don't mind using string literals:

```
query.Include("Collection.Property")
```

answered Jul 29 '18 at 5:32

mrmashal
688 8 16

This method was helpful for me to figure out how this can be coded in VB , as i cant find anywhere after hours of googling. – Coder May 5 at 20:59