Ignore mapping one property with Automapper

Asked 8 years, 7 months ago Active 3 months ago Viewed 178k times

258

39

I'm using Automapper and I have the following scenario: Class OrderModel has a property called 'ProductName' that isn't in the database. So when I try to do the mapping with:

```
Mapper.CreateMap<OrderModel, Orders>();
```

It generates an exception :

"The following 1 properties on Project.ViewModels.OrderModel are not mapped: 'ProductName'

I've read at AutoMapper's Wiki for Projections the opposite case (the extra attribute is on the destination, not in the source which is actually my case)

How can I avoid automapper to make the mapping of this property?

c# model viewmodel automapper object-object-mapping

edited Aug 24 '17 at 17:13
Andries Koorzen
21 7

asked Feb 14 '11 at 0:22
Msam85
1,479 2 13 17

7 Automapper doesn't work that way. Its only concerned about properties on the destination object. The src can contain 100 extra properties -- Automapper only maps the dest properties. There must be something else causing the mapping exception. Can you post some code of what is not working? – PatrickSteele Oct 30 '10 at 2:57

It does what you ask automatically. Post some code to clarify – BeRecursive Nov 10 '10 at 10:18

Have a look at the following posts, these might help you stackoverflow.com/questions/4456519/… stackoverflow.com/questions/4052579/… – Divi Feb 14 '11 at 0:54

3 @Patrick AutoMapper does some tricks with analyzing method/property names. It is possible that there is a property on the source that is being unintentionally mapped even if there isn't an exact match on the destination. This is why there is a ForSourceMember(...Ignore()) to prevent this when it

7 Answers

418

From Jimmy Bogard: `CreateMap<Foo, Bar>().ForMember(x => x.Blarg, opt => opt.Ignore());`

It's in one of the comments at his blog.

edited Nov 27 '11 at 12:28 — Pure.Krome — 46.6k 94 328 537

answered Feb 14 '11 at 1:39 — smartcaveman — 27.3k 23 106 199

9 Also, `CreateMap<Foo, Bar>().ForSourceMember(x => x.Blarg, opt => opt.Ignore());` might be useful – stackoverfloweth Apr 26 '18 at 18:35

2 @stackoverfloweth Don't you mean: `CreateMap<SourceType, DestType> (MemberList.Source).ForSourceMember(x => x.MySourceProperty, opt => opt.DoNotValidate())` ? – monty Dec 3 '18 at 2:40

4 Ignore has been replaced with DoNotValidate in ForSourceMember: github.com/AutoMapper/AutoMapper/blob/master/docs/… – Jamie Jan 4 at 11:11

@Jamie @monty - I started to update this re: your comment, but it looks like the syntax change only affects the projection case (where the source property needs to be ignored). The OP's request is to ignore a destination property, so, `Ignore()` remains the correct syntax. This is because the syntax change for `Ignore` was made on the `ISourceMemberConfigurationExpression` interface but not on the disjoint `IMemberConfigurationExpression`3` interface. – smartcaveman Jan 11 at 7:48

1 @Franva ForMember() is actually "ForDestinationMember()" – rvnlord Aug 2 at 11:45

226

I'm perhaps a bit of a perfectionist; I don't really like the ForMember(..., x => x.Ignore()) syntax. It's a little thing, but it it matters to me. I wrote this extension method to make it a bit nicer:

```
public static IMappingExpression<TSource, TDestination> Ignore<TSource, TDestination>(
    this IMappingExpression<TSource, TDestination> map,
    Expression<Func<TDestination, object>> selector)
{
    map.ForMember(selector, config => config.Ignore());
    return map;
}
```

```
Mapper.CreateMap<JsonRecord, DatabaseRecord>()
        .Ignore(record => record.Field)
        .Ignore(record => record.AnotherField)
        .Ignore(record => record.Etc);
```

You could also rewrite it to work with `params`, but I don't like the look of a method with loads of lambdas.

answered May 29 '13 at 8:23

Steve Rukuts
6,979 3 38 63

6 I know this goes beyond the initial question but I really like this answer, its clean, very easy to read and instantly understand plus easy to reuse – Lski Dec 3 '14 at 16:15

Regarding `params`: You could return an array of selectors from inside a single lambda, then map over each selector with `foreach` or `Select()` Perhaps not less messy-looking, though. – jpaugh May 17 '17 at 23:51

75

You can do this:

```
conf.CreateMap<SourceType, DestinationType>()
   .ForSourceMember(x => x.SourceProperty, y => y.Ignore());
```

answered Apr 16 '12 at 19:12

Richard
928 9 13

Does automapper have a ForSourceMember extension? – Redeemed1 May 21 '12 at 14:46

3 @Redeemed1 Yes it does. – AaronLS Jun 15 '12 at 16:37

I do this currently, but it would be ideal to NOT have to create all these Ignore... :/ – Tom Stickel Mar 14 '13 at 4:49

do you know if there's a way to ignore when actually doing the mapping and not when creating the map? – Sam I am Nov 7 '14 at 17:10

2 For the scenario given in the question, this should be the accepted answer. The current accepted answer ignores mapping of properties in the destination object. This question is asking about ignoring mappings in the source object. – Rob S. Jan 31 '17 at 15:07

28

Just for anyone trying to do this automatically, you can use that extension method to ignore non existing properties on the destination type :

```
public static IMappingExpression<TSource, TDestination> IgnoreAllNonExisting<TSource,
TDestination>(this IMappingExpression<TSource, TDestination> expression)
{
    var sourceType = typeof(TSource);
    var destinationType = typeof(TDestination);
    var existingMaps = Mapper.GetAllTypeMaps().First(x =>
x.SourceType.Equals(sourceType)
        && x.DestinationType.Equals(destinationType));
    foreach (var property in existingMaps.GetUnmappedPropertyNames())
    {
        expression.ForMember(property, opt => opt.Ignore());
    }
    return expression;
}
```

to be used as follow :

```
Mapper.CreateMap<SourceType, DestinationType>().IgnoreAllNonExisting();
```

thanks to Can Gencer for the tip :)

source : http://cangencer.wordpress.com/2011/06/08/auto-ignore-non-existing-properties-with-automapper/

answered Feb 13 '13 at 15:30

Stéphane
8,529 7 40 58

3 FYI: merged from stackoverflow.com/questions/4052579/… – Shog9 ♦ Nov 13 '14 at 21:48

1 This doesn't work when injecting IMapper. Mapper.GetAllTypeMaps doesn't exist in the latest version of AutoMapper. Additionally, when I setup my maps in an AutoMapper.Profile and then subsequently injected IMapper, I got this exception "Mapper not initialized. Call Initialize with appropriate configuration. If you are trying to use mapper instances through a container or otherwise, make sure you do not have any calls to the static Mapper.Map methods, and if you're using ProjectTo or UseAsDataSource extension methods, make sure you pass in the appropriate IConfigurationProvider instance." – Ristogod Dec 27 '16 at 15:36

I just get `'Mapper' does not contain a definition for 'GetAllTypeMaps' [DSSTools]` .. – Bassie Apr 27 '18 at 1:07

28

There is now (AutoMapper 2.0) an `IgnoreMap` attribute, which I'm going to use rather than the fluent syntax which is a bit heavy IMHO.

edited Jun 11 at 8:22

Vahid Farahmandian
3,707 4 27 46

answered Sep 23 '11 at 8:38

Guillaume
799 8 17

32 The ignore attribute leaks auto-mapper through your application though. – Phill Nov 27 '11 at 12:38

11 AutoMapper is one thing which I don't mind leaking all over the place. ;) – Pawel Krakowiak Dec 7 '12 at 16:46

4 You can always consider deriving `IgnoreMapAttribute` . – Alapago Mar 21 '14 at 12:08

1 This is a good way to ignore a base property that is inherited across many objects. Saves from having to ignore it in every mapping config. – Chase Florell Feb 20 '18 at 14:34

21

When mapping a view model back to a domain model, it can be much cleaner to simply validate the source member list rather than the destination member list

```
Mapper.CreateMap<OrderModel, Orders>(MemberList.Source);
```

Now my mapping validation doesn't fail, requiring another `Ignore()` , every time I add a property to my domain class.

answered Jun 15 '15 at 3:49

Loren Paulsen
6,740 1 23 34

6 THIS is what I came looking for, so useful when only modifying a subset of domain object properties from a much simpler DTO. – Adam Tolley Sep 3 '15 at 20:24

4 This is the answer kids, make that official so newbies won't be confused – Piotr M Jul 24 '18 at 23:41

Hello All Please Use this it's working fine ... for auto mapper use multiple **ForMember** in C#

-5

```
        if (promotionCode.Any())
        {
            Mapper.Reset();
            Mapper.CreateMap<PromotionCode, PromotionCodeEntity>().ForMember(d =>
d.serverTime, o => o.MapFrom(s => s.promotionCodeId == null ? "date" : String.Format("
{0:dd/MM/yyyy h:mm:ss tt}", DateTime.UtcNow.AddHours(7.0))))
                .ForMember(d => d.day, p => p.MapFrom(s => s.code != "" ?
LeftTime(Convert.ToInt32(s.quantity), Convert.ToString(s.expiryDate),
Convert.ToString(DateTime.UtcNow.AddHours(7.0))) : "Day"))
                .ForMember(d => d.subCategoryname, o => o.MapFrom(s => s.subCategoryId
== 0 ? "" : Convert.ToString(subCategory.Where(z =>
z.subCategoryId.Equals(s.subCategoryId)).FirstOrDefault().subCategoryName)))
                .ForMember(d => d.optionalCategoryName, o => o.MapFrom(s =>
s.optCategoryId == 0 ? "" : Convert.ToString(optionalCategory.Where(z =>
z.optCategoryId.Equals(s.optCategoryId)).FirstOrDefault().optCategoryName)))
                .ForMember(d => d.logoImg, o => o.MapFrom(s => s.vendorId == 0 ? "" :
Convert.ToString(vendorImg.Where(z =>
z.vendorId.Equals(s.vendorId)).FirstOrDefault().logoImg)))
                .ForMember(d => d.expiryDate, o => o.MapFrom(s => s.expiryDate == null ?
"" : String.Format("{0:dd/MM/yyyy h:mm:ss tt}", s.expiryDate)));
            var userPromotionModel = Mapper.Map<List<PromotionCode>,
List<PromotionCodeEntity>>(promotionCode);
            return userPromotionModel;
        }
        return null;
```

edited Dec 22 '15 at 14:04

answered Dec 15 '15 at 14:30

ravikant sonare

1 1