How to Implement AutoMapper with Generic Repository Pattern

Asked 2 years, 2 months ago Active 2 years, 2 months ago Viewed 2k times

0

1

I am trying to implement AutoMapper with Generic Repository Pattern where it contains BaseController, BaseRepository of T type.

MapperConfig:

This is part of the code where AutoMapper is implemented in WebAPI Project and this is called on App_Start()

```
 public static class APIMapperConfig
{

    static MapperConfiguration adminConfig;

    public static IMapper adminMapper;

    public static void Configure()
    {
        ConfigureAdminConfiguration();
    }


    public static void ConfigureAdminConfiguration()
    {
        adminConfig = new MapperConfiguration(cfg => {
            cfg.CreateMap<ArticleType, DAL.ArticleType>();
            });

        adminMapper = adminConfig.CreateMapper();

        adminMapper.Map<ArticleType, DAL.ArticleType>(new ArticleType());
    }

}
```

APIModel(API.ArticleType)

This is a model which is used by API to receive data from client.

```
public class ArticleType
    {

        public int id { get; set; }

        public String name { get; set; }

        public String displayName { get; set; }

    }
```

EntityModel(DAL.ArticleType)

This is EntityFramework auto generated model

```
public class ArticleType
    {

        public int id { get; set; }

        public String name { get; set; }

        public String displayName { get; set; }

    }
```

ArticleTypeController: (Project: API)

This is the controller which is called to add ArticleType model and it is derived from BaseController of T type.

```
 public class ArticleTypeController : BaseController<ArticleType>
    {

        private IArticleTypeServices article;

        ArticleTypeController()
        {
            this.article = UnityConfig.ResolveObject<IArticleTypeServices>();
        }
    }
```

BaseController: : Project(API)

This is BaseController of T type, this is used to perform common tasks like Add, Update etc for each model.

```
    public class BaseController<T> : ApiController
        {
        IBaseRepository<T> rep;

        public BaseController()
        {
            rep= UnityConfig.ResolveObject<IBaseRepository<T>>();
        }
            [HttpPost]
            public void Add(T item)
            {
                rep.Add(item);
            }
        }
```

BaseRepository: (project: DALrepository)

This repository is called by BaseController to add model, and this repository calls Entity framework to add in database.

```
 public class BaseRepository<T> : IBaseRepository<T> where T : class
    {
        BlogDBContext db;

        public BaseRepository()
        {
            db = UnityConfig.ResolveObject<BlogDBContext>();

        }

        public void Add(T item)
        {

         // Here is the problem

    /*item which is passed from Controller is of type API.ArticleType, and what is
expected to pass to entity frmawework is of type DAL.ArticleType.
       Here how do I map between API type and DAL type? It is to be noted that
ArticleType of API and DAL is already mapped inside AutoMapper code at top.*/
```

```
            this.Entities.Add(item);
            this.db.SaveChanges();
        }
```

When I run this in Fiddler (http://localhost:xxxxx/api/ArticleType), It throws error:

the entity type ArticleType is not part of the model for the current context

This is obvious since there is no mapping between ArticleType of API and DAL.

This error occurs in BaseRepository Add().

I believe this could be issue with the way I have called Map<> in App_Start.

The code might look lengthy, but if I have missed anything to put please let me know.

All your help and time are much appreciated.

c# entity-framework generics asp.net-web-api automapper

edited Jan 15 '18 at 12:02

asked Jan 14 '18 at 9:07
Dheeraj Kumar
2,788 ● 3 ● 23 ● 45

Just use a *canonical data model/schema* and be done with it. Defining essentially the same type per layer adds nothing but increased maintenance. soapatterns.org/design_patterns/canonical_schema – MickyD Jan 14 '18 at 9:23

downvote??? for what? – Dheeraj Kumar Jan 14 '18 at 9:23

@MickyD. It is not recommended to use Entity Mode as BusinessModel. is it..? – Dheeraj Kumar Jan 14 '18 at 9:24

1 Answer

Active Oldest Votes

2

After configuring auto mapper you have to use it within your code to do the transformation. Within your Add method you only have an information about the source type but you are missing the desired destination type. So you need some kind of type mapping. This has to be done manually, through some kind of `Dictionary<Type, Type>` (the job of AutoMapper is to automatically map properties from one type to another, but not to figure out what destination type would best fit). By having this kind of information you could create your method like this:

```
public void Add(T item)
{
    var destinationType = _mappings[typeof(T)];
    var newEntity = _mapper.Map(item, typeof(T), destinationType);

    this.Entities.Add(newEntity);
    this.db.SaveChanges();
}

// Maybe injected through UnityConfig...
private static _mappings = new Dicionary<Type, Type> {{ typeof(API.ArticleType),
typeof(DAL.ArticleType) }};
```

answered Jan 15 '18 at 12:26
Oliver
36.2k ● 7 ● 77 ● 126

Thank you for your time. Considering there would be no way except providing required type through external object as you suggested a dictionary... Would this be an ideal architecture as whole. My purpose here is to make code as generic as possible. Please suggest. – Dheeraj Kumar Jan 15 '18 at 13:26

1 Sometimes it is not ideal to have anything as generic as possible. This may lead to problems when you have *special cases*. So I'm a fan of this approach (also maybe watch the video, linked in his blog, linked in the readme). What you really do in your case is up to you and whatever I would suggest to you, you could have a dozen arguments for or against it, just to drive it to the direction you like. All I can say is, you need this kind of information, how to get it into the controller is your job. ;-) – Oliver Jan 15 '18 at 14:00

Thank you for your valuable advice ;) – Dheeraj Kumar Jan 16 '18 at 6:12