Mapper , Imapper<TEntity, TEntity>

Mapper slouzi k prevodu mezi vstupem a vystupem napr IMapper < Galaxie, GalaxyDataContract> ma jako vstup entitu a vystup dataContract. Mapper vezme vsechny property na entite a vyplni stejnomenne property na datacontractu ktery vrati.

Mapper je mozne pouzivat jako AutoMapper anebo si vytvorit vlastni implementaci rozhrani IMapper.

Aby bylo mozne IMapper pouzivat

* Vytvorime slozku Core.Mappers a v ni bude trida Scanning2CoreMappersConfiguration : MappingConfiguratorBase. Pomoci metody CreateMap definujeme vsechny potrebne mappery.

V nasl. prikladu uprostred Mapper nam pomuze namapovat vice tabulek a vlozit je do jedne mapovane tabulky . Potrebujeme pomoci mapperu vytvorit datacontract z entit v tabulkach PublicationPeriodicity a MediumScanningPreset a vlozit je do mapovane tridy Medium. Takze vlastne Medium bude mit na sobe vlastnosti Peridicity a ScanningPreset ktere v db vubec nema, navic na techto vlastnostech jsou dalsi podvlastnosti podle toho jake sloupce maji . pridane tabulky.

public class Scanning2CoreMappersConfiguration : MappingConfiguratorBase

{

protected override void ConfigureInternal()

{

CreateMap<Medium, MediumDataContract>();

CreateMap<MediumPublicationPeriodicity, PublicationPeriodicityDataContract>()

.ForMember(d => d.PeriodicityTypeId, x => x.MapFrom(v => v.PeriodicityType))

.ForMember(d => d.DayOfMonth, x => x.MapFrom(v => v.BasicFrequency));

CreateMap<MediumScanningPreset, MediumScanningPresetDataContract>();

}

}

* Vytvorime slozku Core.Installers kde bude trida MappingInstaller. Zaregistrujeme zde ConfigurationBase a kazdy IMapper. IMapper je implementovany by SimpleMapper, bez teto implementace Castle rve, ze nemuze vytvorit instanci rozhrani. Tridu SimpleMapper ja nevytvarim ta je pravdepodobne soucasti balicku

namespace MIR.Media.Changing2.Core.Installers

{

public class MappingInstaller : IWindsorInstaller

{

public void Install(IWindsorContainer container, IConfigurationStore store)

{

container.Register(Component.For<MappingConfiguratorBase>().ImplementedBy<Changing2CoreMappersConfiguration>().LifestyleSingleton());

container.Register(Component.For<IMapper<GetDataRequest, MotivletFilterParams>>().ImplementedBy<SimpleMapper<GetDataRequest,MotivletFilterParams>>());

}

}

}

* Metoda Install se nachazi na trech mistech a reference ukazuji navzajem na sebe. Umisteni jsou : ChangingInstaller, MappingInstaller a MediaDataDbAccessInstaller. V ChangingInstalleru je

container.Register(Component.For<DependencyMappingConfigurator>().ImplementedBy<DependencyMappingConfigurator>().LifestyleSingleton());

Vlastni implementace Mapperu

namespace WpfUniverse.Core.Mappers

{

public class GalaxyDataContractMapper : IMapper<Galaxie, GalaxyDataContract>

{

public GalaxyDataContract Map(Galaxie obj)

{

return new GalaxyDataContract(obj.Id, obj.Jmeno, obj.PolohaX, obj.PolohaY, obj.PolohaZ);

}

}

}

Se musi zaregistrovat do conteineru

public void Install(IWindsorContainer container, IConfigurationStore store)

{

container.Register(Component.For<IMapper<Galaxie, GalaxyDataContract>>()

.ImplementedBy<GalaxyDataContractMapper>());

}

Dalsi priklad: Ve zmenovadle uz byly funkcni mappery. Dostal jsem za ukol vytvorit explicitni mapper (ne auto) .

* Ve slozce Core.Mappers jsem vytvoril tridu ComposedContractMapper

To co se puvodne tahalo z ciselniku v servisni akci se ted bude dit v mapperu. Je to vlastne hodne podobna trida jako servisni akce akorat se zmenil nazev metody.

V tomto pripade dokonce mapujeme objekt sam na sebe . (to znamena ze mu jen zmenime nektere property podle hodnot z ciselniku a vratime stejny objekt)

public class ComposedContractMapper : IMapper<List<ComposedDataContract>, List<ComposedDataContract>>

{

private readonly IEnumTableValues<Category, short> m\_categories;

private readonly IEnumTableValues<Gender, byte> m\_genders;

private readonly IEnumTableValues<Market, byte> m\_markets;

private readonly IEnumTableValues<Platform, byte> m\_platforms;

private readonly IEnumTableValues<Role, byte> m\_roles;

private readonly IEnumTableValues<Telco, byte> m\_telcos;

public ComposedContractMapper(IEnumTableValues<Gender, byte> genders, IEnumTableValues<Category, short> categories, IEnumTableValues<Role, byte> roles, IEnumTableValues<Market, byte> markets,

IEnumTableValues<Telco, byte> telcos, IEnumTableValues<Platform, byte> platforms)

{

m\_genders = genders;

m\_categories = categories;

m\_roles = roles;

m\_markets = markets;

m\_telcos = telcos;

m\_platforms = platforms;

}

public List<ComposedDataContract> Map(List<ComposedDataContract> obj)

{

foreach (var item in obj)

{

item.GenderName = item.GenderId.HasValue ? m\_genders.GetValueById(item.GenderId.Value).NameValue : string.Empty;

item.CategoryName = item.CategoryId.HasValue ? m\_categories.GetValueById((short) item.CategoryId).Name : string.Empty;

item.CategoryCode = item.CategoryId.HasValue ? m\_categories.GetValueById((short) item.CategoryId).Code : string.Empty;

item.RoleName = item.RoleId.HasValue ? m\_roles.GetValueById(item.RoleId.Value).Name : string.Empty;

item.MarketName = item.MarketId.HasValue ? m\_markets.GetValueById(item.MarketId.Value).Name : string.Empty;

item.TelcoName = item.TelcoId.HasValue ? m\_telcos.GetValueById(item.TelcoId.Value).Name : string.Empty;

item.PlatformName = item.PlatformId.HasValue ? m\_platforms.GetValueById(item.PlatformId.Value).Name : string.Empty;

}

return obj;

}

}

* Mapper si zaregistruju v Core.MappingInstaller

namespace MIR.Media.Changing2.Core.Installers

{

public class MappingInstaller : IWindsorInstaller

{

public void Install(IWindsorContainer container, IConfigurationStore store)

{

container.Register(Component.For<MappingConfiguratorBase>().ImplementedBy<Changing2CoreMappersConfiguration>().LifestyleSingleton());

container.Register(Component.For<IMapper<GetDataRequest, MotivletFilterParamLists>>().ImplementedBy<SimpleMapper<GetDataRequest, MotivletFilterParamLists>>());

container.Register(Component.For<IMapper<List<ComposedDataContract>, List<ComposedDataContract>>>().ImplementedBy<ComposedContractMapper>());

}

}

}

**Zajimave je ze registruju rozhrani a implementuju ho svoji tridou**

* Pouziti je v servisni akci v miste kde se predtim vsechno mapovalo rucne.
* Do konstruktoru si pridame zavislost na Imapper

public class GetDataAction : ServiceAction<GetDataRequest, ListResponse<ComposedDataContract>>

{

private readonly IMapper<GetDataRequest, MotivletFilterParamLists> m\_filterMapper;

private readonly MotivletDao m\_motivletDao;

private readonly IMapper<List<ComposedDataContract>, List<ComposedDataContract>> m\_composedMapper;

public GetDataAction(MotivletDao motivletDao, IMapper<GetDataRequest, MotivletFilterParamLists> filterMapper,

IMapper<List<ComposedDataContract>, List<ComposedDataContract>> composedMapper)

{

m\_motivletDao = motivletDao;

m\_filterMapper = filterMapper;

m\_composedMapper = composedMapper;

}

protected override ListResponse<ComposedDataContract> ExecuteInternal(GetDataRequest request)

{

var parameters = m\_filterMapper.Map(request);

var list = m\_motivletDao.GetComposedMotive(parameters);

list = m\_composedMapper.Map(list);

return new ListResponse<ComposedDataContract>(list);

}

}

**Vse**

Dalsi priklad:  **Automapper**

Uz mappery funguji potrebuju pridat jeden automapper :

1. V MappingInstalleru pridam registraci

namespace MIR.Media.Changing2.Core.Installers

{

public class MappingInstaller : IWindsorInstaller

{

public void Install(IWindsorContainer container, IConfigurationStore store)

{

container.Register(Component.For<IMapper<GetValuesByParametersRequest,MotivletFilterParamLists>>().ImplementedBy<SimpleMapper<GetValuesByParametersRequest, MotivletFilterParamLists>>());

}

}

}

1. V Changing2CoreMappersConfiguration pridam CreateMap:

namespace MIR.Media.Changing2.Core.Mappers

{

public class Changing2CoreMappersConfiguration : MappingConfiguratorBase

{

protected override void ConfigureInternal()

{

CreateMap<GetDataRequest, MotivletFilterParamLists>();

CreateMap<GetValuesByParametersRequest, MotivletFilterParamLists>();

}

}

}

**Vse**