Auto Mapper in multiple –tier MVC 5 WCF web application

Author: David Lizhong Huang

Date: 3, May, 2015

Introduction

We have data entity classes called product {id, name, price, quantity…} and productdetails such as {id, productid, date,…}. We can use code first to create a database for both classes that can be created as a product and a productdetals table in database. Now we need to build an MVC application to do CRUD operations on those tables and publish those tables' data in UI. UI may only need a couple of columns such as name and price data from those tables. Therefore, we build an asp.net entity data model to create a context repository class that can pull and push data from/into database for those tables' CRUD operations based on those tables' schema. Therefore, this data access layer uses entity class for data transfer.

Now assume we host this data access layer in cloud and we want to use it in our local network. We can inject this data access layer to local WCF web service we develop for accessing to this data access layer. When we develop CRUD in WCF web service, we quickly find out that we need to pass the data from client front end to them. We also quickly discover that we cannot pass the same data structure as the one in database table normally due to the relationships among tables. We then have to create a model class that can be used in WCF service to do the data transfer from client to this data access layer. This model class normally is different from entity class so it cannot be used directly. We need a mechanism to map both class properties so that they can know each other. AutoMapper component is invented for this mechanism see below example. In this scenario, we can say we are doing the backend mapping. Of course, we have a frontend mapping if necessary. For example, we also can host WCF web service remotely such as on Moon (yes, we can host our web site on Moon). Now, how can we pass data to this web service from front end? It is easy. Front end layer takes WCF web service as a data source. It also needs to map the data from UI to WCF. Business maybe does not need to use the model class from WCF to collect data; it creates its own view model class. So we also need a mechanism to Map front end view model class and middle-tier model class. This is so called front end mapping. AutoMapper component can also be used in front end mapping as example shown below.

Get Started

1, Update the entity data model

A new table such as Film table that contains over 1000 records will be injected into Entity Framework 6 for front end data consumer. Using the architecture in the web application, we can easily go through all necessary codes to wire up this table to front end. See the procedure below

1), Data Context films

```
        public DbSet<film> films { get; set; }
```

2), Repository

```
 public List<film> GetAll()
        {
            return Db.films.ToList();
        }
```

Repository used film entity class from database.

3), WCF web service

```
[OperationContract]
List<BusinessRules.Film> GetAllFilms();
```

WCF web service uses model class from business rules.

From 2 and 3 steps, we can find out that GetAll() method returns list of films from database, however, GetAllFilms() in WCF web service will return a list of films for Business Rules. Therefore, after WCF received data from GetAll() resource, it needs to convert data via AutoMapper for WCF, See example below.

a) Without using AutoMapper

```
 public List<BusinessRules.Film> GetAllFilms()
        {
            var al = fmDb.GetAll();
            var flm = new BusinessRules.Film ();
            var filmlist = new List<BusinessRules.Film>();
            filmlist.Clear();
            foreach (var lst in al)
            {
                flm.description = lst.description;
                flm.film_id = lst.film_id;
                flm.language_id = lst.language_id;
                flm.last_update = lst.last_update;
                flm.length = lst.length;
                flm.original_language_id = lst.original_language_id;
                flm.rating = lst.rating;
                flm.release_year = lst.release_year;
                flm.rental_duration = lst.rental_duration;
                flm.rental_rate = lst.rental_rate;
                flm.replacement_cost = lst.replacement_cost;
                flm.special_features = lst.special_features;
                flm.title = lst.title;

                filmlist.Add(flm);
            }

            return filmlist;

        }
```

b) Via using AutoMapper

```
public List<BusinessRules.Film> GetAllFilms()
        {
            var ServerDbs = fmDb.GetAll();
            var FilmList = new List<BusinessRules.Film>();
            foreach (var ServerDb in ServerDbs)
            {
               var ClientDb= new BusinessRules.Film();
        AutoMapper .Mapper .CreateMap <MySQLRepository .film , BusinessRules .Film >();
```

```
    var ToClient=AutoMapper .Mapper .Map <BusinessRules.Film>(lst);
    FilmList.Add(ToClient);
    }
      return FilmList;
     }
```
We can easily to find out that AutoMapping easily simplify our coding job. The backend mapping is then done.

4) WCF web service client

WCF web service normally is used in web service proxy application. Here we can build up the end points for WCF web services that can be used to build up the connection to client and server. See code below

```
        public List<BusinessRules.Film> GetAllFilms()
        {
            return db.GetAllFilms().ToList();
        }
```

Client simple returns list of data from WCF web service via using Film model class from business rules. It may be different from the view model class in front end.

5), MVC Data Transmitter

We now build up MVC front end to deal with this web service client see the code below

```
      SunyinHealth.Webservices.MySQL.IMySqlSoaCLient  db;
      public mysqltomodel(SunyinHealth.Webservices.MySQL.IMySqlSoaCLient _db)
      {
          db = _db;
      }
```

We inject this web service client dependency into constructor in MVC data transmitter class via Unity DI container. However, MVC views or Angular JS view do not need all properties from the model class in WCF proxy client; it needs view model class to transfer data. AutoMapper therefore is used here to map model class and view model class. See example below

```
public List<Film> clientGetAllFilms()

        {
            List<Film> webnewList = new List<Film>();
            var serverDb = db.GetAllFilms();
            AutoMapper.Mapper.CreateMap<BusinessRules.Film, data.Film>();
            if (serverDb != null)
            {
                foreach (var sd in serverDb)
                {
                    Film map = AutoMapper.Mapper.Map<data.Film>(sd);
                    webnewList.Add(map);
                }
            }
            else
            {
                webnewList = null;
            }
            return webnewList;         }
```

Here data.Film is the view model class that Is mapped to BusinessRules.Film model class in WCF web service client. We do the front end mapping here. Now it is done.

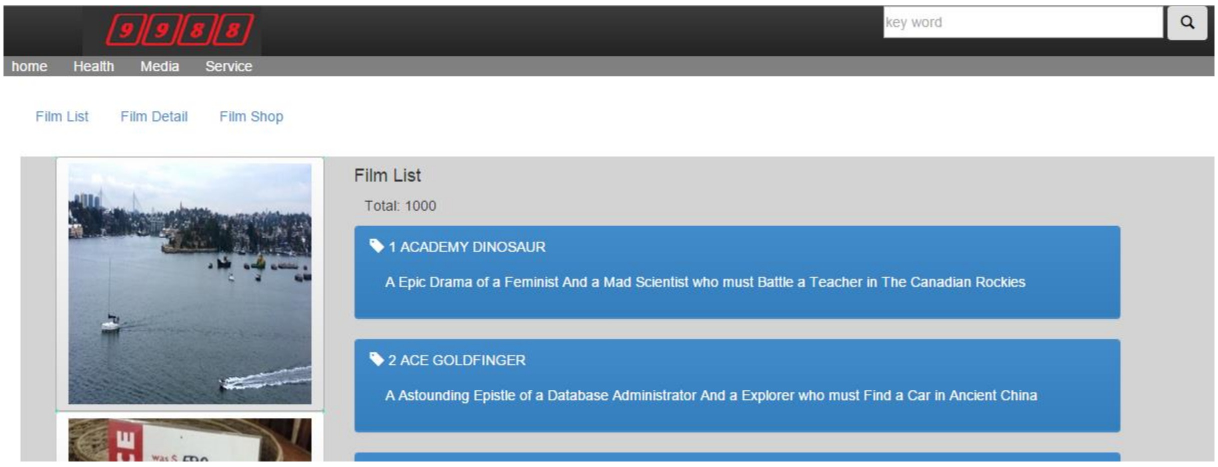
5) MVC controller

Finally we can inject this data transmitter class into MVC controller via Unity DI container see code below

```
Imysqltomodel db;
public FilmsController(Imysqltomodel _db)
{
       db = _db;
}

public ActionResult List()
{
       var all = db.clientGetAllFilms();
       return View(all);
}
```

```
The final result can be shown as below
```



Film List

Total: 1000

1 ACADEMY DINOSAUR

A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies

2 ACE GOLDFINGER

A Astounding Epistle of a Database Administrator And a Explorer who must Find a Car in Ancient China

6) Big data process

When a page wants to display over 1000 records, normally we need to customize some settings in the client and server end points of the WCF web services from web.config, so the web server can expand its timeout for big data processing.

Summary

AutoMapper component can be used to map view model class and model class in front end and model class and entity class in backend. Use install-package automapper we can simply pull this component from internet. AutoMapper also allows us to set the mapping manually.