Development of enterprise base MVC 5 WCF web application

Author: David Lizhong Huang

Date: 3, May, 2015

Introduction

This is a big data era. .Net web technology will play a key role in this great time. Enterprise business will find out that .Net web technology is the best option for their e-businesses. No matter when they will implement mobile strategy, I have experience in IBM BlueMix Cloud, AWS web service, and Microsoft Azure. I found out that Microsoft .Net Web technology is the best option.

Now it is clear to build up enterprise base web application as below

1, Extract classes from business requirement and build up the OO class models for application

2, Use Dependency Injection and SOLID design principle and other design patterns such as OO design pattern, MVC design pattern, MVVM design patterns, WPF Prism design patterns, Angular JS design pattern, factory and Repository design patterns, Entity Framework and LINQ design patterns, Dependency injection design pattern, etc.

3, Bring in Scrum development method to create user stories for developers

4, Work under TFS environment to use the MS build CI and deployment benefits

5, Use TDD development methodology such as .Net Unit test, Jasmine JavaScript unit test, and Bower/Grunt/Gulp automation test to get started the project

6, Use multiple tier architecture to develop loose coupled web application.

- We normally can develop the class library project of asp.net entity data model and repository class as a data access layer
- We can develop a WCF web service and/or asp.net web API or other REST web services as a data access proxy layer
- We then can develop another class library project as a web service client that can build up the end points of web services for client caller, this is called SOA project
- Finally we can build up front end web applications to access to this web service client backend
- Front end application can be based on Mobile, HTML+AngularJS (node js etc), HTML+mobile Jquery, MVC, Asp.Net Web Form, WPF, Window Store, and mixed front end such as MVC+Angular JS, etc.

7, Use Octopus Deploy, TeamCity, and other deployment technology to implement web application

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.

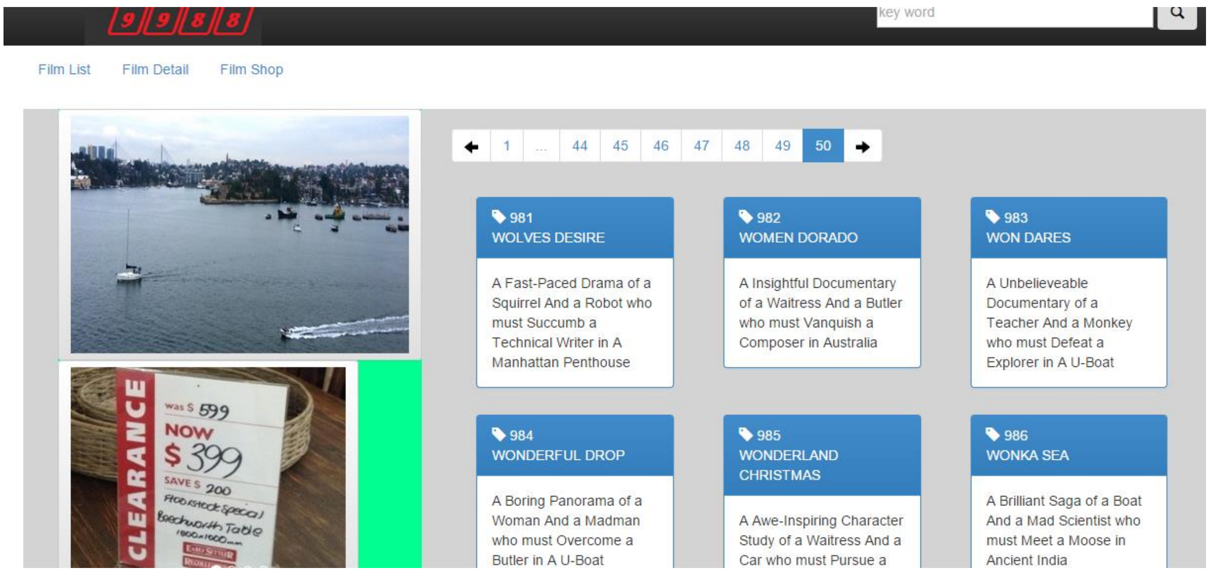
5) MVC controller

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`



key word

Film List   Film Detail   Film Shop

1 ... 44 45 46 47 48 49 50

981
WOLVES DESIRE
A Fast-Paced Drama of a Squirrel And a Robot who must Succumb a Technical Writer in A Manhattan Penthouse

982
WOMEN DORADO
A Insightful Documentary of a Waitress And a Butler who must Vanquish a Composer in Australia

983
WON DARES
A Unbelieveable Documentary of a Teacher And a Monkey who must Defeat a Explorer in A U-Boat

984
WONDERFUL DROP
A Boring Panorama of a Woman And a Madman who must Overcome a Butler in A U-Boat

985
WONDERLAND CHRISTMAS
A Awe-Inspiring Character Study of a Waitress And a Car who must Pursue a

986
WONKA SEA
A Brilliant Saga of a Boat And a Mad Scientist who must Meet a Moose in Ancient India

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.