Develop Angular JS MVC 5 MySQL multiple tier web application

Author: David Lizhong Huang

Date: 19, Apr, 2015

1, Introduction

If we want to manipulate DOM operation in HTML, JQuery is enough. However, A big data era needs front end application can integrate with data from backend. JQuery is not strong enough to support such a requirement. Nowadays the big change of modern IT industry is the invention of mobile strategy for businesses. A lots of front end JS frameworks are developed to meet such a tendency. Angular JS is one of excellent front end JS framework that is widely used due to its excellent MVW and performance for multiple devices. No matter what Angular JS inventors said that the Angular JS is reinventing and will have a new version at the end of 2015, Business cannot wait for this reinventing. Business needs Angular JS now.

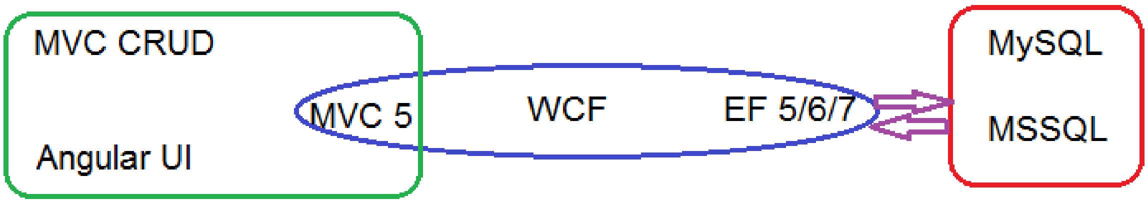
MVC 5 is an excellent .Net middle tier and front end development pattern we can rely on and develop it for Angular JS front end. Controllers in MVC 5 not only provides an extra layer of data source for Angular JS front end but also supply an extra security mechanism to protect data and business logics from backend. MVC 5 can integrate into WCF seamlessly via DI technology. We inject WCF SOA services into MVC 5 controllers. Angular Front end now can ignore the backend development. Developer needs to develop MVC controllers for angular HTTP services that is the main data source Angular front end can use.

WCF development is the citizen in the backend development. It is a business logic layer only that is a class library assembly with ABC endpoints configuration mechanism installed. It needs to open its mouth to angular JS front end to get food from Angular and to open its mouth twin to backend to feed the food from angular front end to backend data source. It also accepts data from backend and sends the data back to MVC5. WCF constructs SOA together with other web services such as asp.net Web API, Service Stack, and Java web services, etc. It is a straightforward to develop a WCF web services, however, it needs skills to glue WCF to MVC 5 and data access layer.

Data access layer in .Net now is widely developed by using Entity Framework 5/6/7. Entity Framework not only can build entity data model from MSSQL but also can generate entity data models from other databases such as MySQL database system. Repository assembly developed from those entity frameworks can then be injected into WCF web services as a data service source. VS 2013 and VS 2015 can allow us to develop such an entity framework based data access layer quickly and easily. Entity Data Models can be updated seamlessly in anytime without affect the front end development. We can use different Dis to inject those data sources into WCF. VS 2015 can make us to do this job done easier and more quickly.

MySQL and MSSQL are the excellent data stores for Entity Framework data access layer development. We can develop databases systems via code first. Then we can develop various plans to develop database systems. For example, based on the business requirement, we decide if we need to develop store procedures and functions in databases for database performance and data security. In some cases, we leave the data security to front end. So we can simply develop LINQ syntax for data manipulation such as CRUD operations from web pages. MVC 5/6 provides excellent CRUD operations for us to manipulate data from web site.

Therefore, after I completed the backend development, I simply develop MVC CRUD operations for data management by myself and angular JS front end for user experience. See image below.



2, Project Example – Online Advertisement

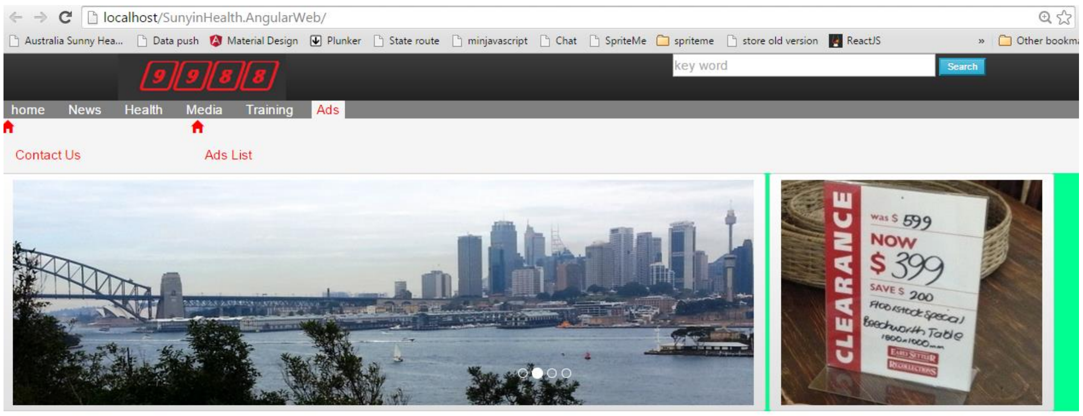
This section will mainly show you one example I developed in this angular JS front end web application. I want to develop an e-commerce web site combining with the daily Australia news abstract web contents. So people can browse news and go to shopping online at any time.

Free Online Advertisement

This example is the free advertisement online. This web function will allow users to create their own advertise content in web without any charges. Traditionally we add new ads to database and needs to refresh the page to get advertisement list back. Using Angular JS, We do not need to refresh the page. We can display the data list in real time (of course, we can use singular in .Net to do so). How do I achieve this? I will show you the example below

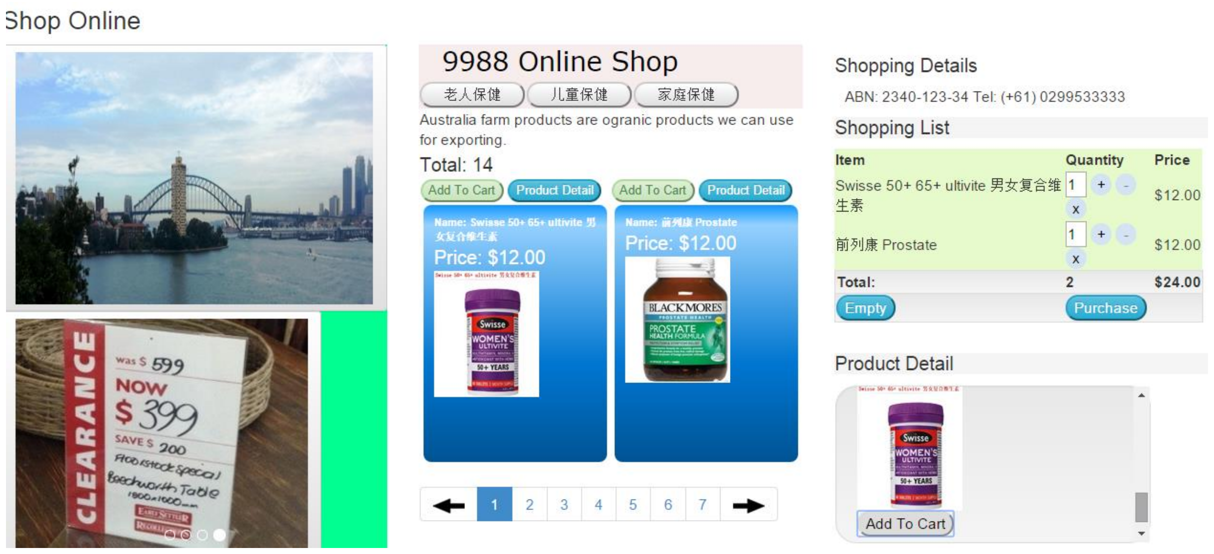
Interface:

1, front page


localhost/SunyinHealth.AngularWeb/
Australia Sunny Hea... Data push Material Design Plunker State route minjavascript Chat SpriteMe spriteme store old version ReactJS Other bookm
9988
key word
Search
home News Health Media Training Ads
Contact Us
Ads List
CLEARANCE
was $ 599
NOW $ 399
SAVE $ 200

Australia Daily News

2, Shop Online


Shop Online

9988 Online Shop

Australia farm products are ogranic products we can use for exporting.

Total: 14

Name: Swisse 50+ 65+ ultivite 男女复合维生素
Price: $12.00

Name: 前列康 Prostate
Price: $12.00

Shopping Details

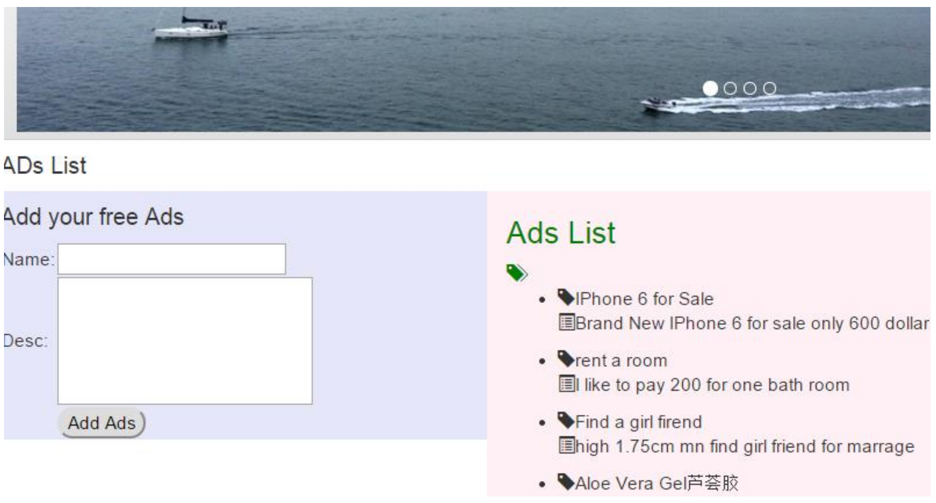
ABN: 2340-123-34 Tel: (+61) 0299533333

Shopping List

Item	Quantity	Price
Swisse 50+ 65+ ultivite 男女复合维生素	1 + - x	$12.00
前列康 Prostate	1 + - x	$12.00
Total:	2	$24.00

Product Detail

3, Free Online Advertisement


ADs List

Add your free Ads

Name:

Desc:

Ads List

- IPhone 6 for Sale
  Brand New IPhone 6 for sale only 600 dollar
- rent a room
  I like to pay 200 for one bath room
- Find a girl firend
  high 1.75cm mn find girl friend for marrage
- Aloe Vera Gel芦荟胶

CSS bootstrap:

```
<div class="row">

<div class="col-sm-4" style="background-color:lavender;" ng-
controller="PushAdsController">
                <input id="zId" ng-model="Id" type="hidden" />
                <input id="zsku" ng-model="sku" type="hidden" />
                <input id="zUnitPrice" ng-model="UnitPrice" type="hidden" />
                <input id="zStock" ng-model="Stock" type="hidden" />
                <input id="zCId" ng-model="CId" type="hidden" />
                <input id="zSndId" ng-model="sndId " type="hidden" />
                <input id="zQuantity" ng-model="Quantity" type="hidden" />
                <input id="zDiscount" ng-model="Discount" type="hidden" />
                <input id="zlast_updated" ng-model="last_updated" type="hidden" />
                <h4>Add your free Ads</h4>
                <table>
                     <tr>
```

```
                    <td>
                           Name:</td><td><input id="Name" data-ng-model="Name" />
                    </td>
                    </tr>
                    <tr>

                    <td>
                           Description:</td><td><input id="Desc" data-ng-model="Desc"
                    type="text" cols="40" rows="10" style="width:200px;
                    height:100px;" />
                    </td>
                    </tr>

                   <tr>
                       <td></td>
                       <td>
                           <button ng-click="insertads()">
                               Add Ads
                           </button>
                       </td>
                   </tr>
               </table>
</div>
<div class="col-sm-8" style="background-color:lavenderblush;" ng-
controller="DisplayAdsController" >
               <div>
                   <span style="color:{{statusColor}}">
                       <h3>Ads List</h3>
                       <span class="glyphicon glyphicon-tags"></span>
                   </span>
               </div>
              <ul ng-repeat="ad in Adslists" style="display: -webkit-box;">
                   <li>
                       <span class="glyphicon glyphicon-tag"></span>{{ad.Name}}<br />
                       <span class="glyphicon glyphicon-list-alt"></span>{{ad.Desc}}
                   </li>
               </ul>
          </div>
       </div>

</div>
```

We have two angular controllers here. They can talk each other. Therefore, this free Advertisement function I developed here can also be used as an online Chat.

JS:

CSS bootstrap builds the UI for Angular data.   JS now needs to be developed under the help of Angular JS framework. At first, Angular wants us to develop an application module for it. This module will be loaded when UI page is loaded. A global $rootScope is created. Secondly, Angular wants us to develop controllers for its module. After those controllers are loaded, local $scope is created for each controller. $scope from different controllers can communicate each other. &rootScope is used as a global data or message transport bus to transfer data among different controllers. Angular then use $broadcast and $emit to chat each other. Therefore, data exchange can happen in real time. See the JS code below

```
//Controller to add new Ads item
dirApp.controller('PushAdsController', function ($scope, $http, $rootScope) {
  var count = 0;

  $scope.insertads = function ()
    {
        count++;
        var ozpt = new OzProduct(1, $scope.Name, $scope.Name, $scope.Desc, 0, 0, 1, 2,
0, 0, null);

         $http.post('http://localhost/SunyinHealth.AngularWeb/UPOClass/AddAds',
{ ozpdt: ozpt })
            .success(function (data, status, headers, config) {

                     $rootScope.$broadcast('EVENT_NO_DATA', 'New data is displayed
below');
                    $rootScope.$on('EVENT_RECEIVED', function () {
                        console.log('Received from Ads display controller’s emitter
that tells us that Ads are displayed in real time');
                    });
             })
            .error(function (data, status, headers, config) {
                console.log('not success');
            });

    }

});

//Controller to display Ads list  after new Ads is added.
dirApp.controller('DisplayAdsController', function ($scope, $http, $rootScope) {

    $http.get('http://localhost/SunyinHealth.AngularWeb/UPOClass/GetAllAds')
           .success(function (data, status, headers, config) {
               $scope.Adslists = data; //need to put to grid
           })
           .error(function (data, status, headers, config) {
               console.log('not success');
           });

        $scope.name = ‘Display’;
        $scope.status = 'Connected';
        $scope.statusColor = 'green';

    //reload when new Ads is added.
        $rootScope.$on('EVENT_NO_DATA', function (event, data) {
            console.log('receive event from PushAdsController');
            $scope.status = data;
            $scope.statusColor = 'red'; //status is changed

            $http.get('http://localhost/SunyinHealth.AngularWeb/UPOClass/GetAllAds')
            .success(function (data, status, headers, config) {
                $scope.Adslists = data; //need to put to grid
                $scope.$emit('EVENT_RECEIVED');
            })
            .error(function (data, status, headers, config) {
                console.log('not success');
            });
        });
});
```

<u>HTTP service:</u>

From those controllers, we can discover that both Angular controllers use HTTP service to get data from a resource URL. Angular collects data input from UI and pass those data input as an object to HTTP service. HTTP post method passes this data input object to this resource URL. URL sends this request with data object to the data services from backend , backend responses the request and sends data back to Angular HTTP service via Success() function. After Successful response, controller broadcasts a message to other controllers. Other controller receives the broadcast message and decides to send new search data list request to backend via HTTP service, backend then finally returns all Ads data list back to Angular UI and emits an event back to other controllers to tell it the final status. Therefore, after we click Add Ads button, two controllers talks each other to make job done in one time. Therefore, a real time chat communication is happened. A new Ads is displayed immediately after button is clicked without refreshing this page. This is so called Angular SPA page.

<u>Resource URL</u>

Now we know Angular HTTP service can get data from resource URL. Those URLs actually are the actions in MVC 5 controllers. Based on MVC 5 Routing mechanism, we know we access to web site UPOClass controller to call action AddAds for data input and action GetAllAds to display all Ads data. Therefore, controller opens two mouths for Angualr UI. The codes are listed below

```
[HttpPost]
        public ActionResult AddAds(data.OzProduct ozpdt)
        {
            db.CreateOzproduct(ozpdt);
            return RedirectToAction("Adslist");
        }

        [HttpGet]
        public JsonResult GetAllAds()
        {
            return Json(db.clientAll_ozproduct().Where(x=>x.SndId==2),
JsonRequestBehavior.AllowGet);
        }
```

AddAds action is a behaviour to insert new record to database. However, GetAllAds action will returns all Ads list data from backend to UI, therefore, it is return format should be a JonResult. HTTP service from Angular needs Json data. The codes here are very simple. However, its background process is huge. This section combines MVC 5 with Angualr JS.

Db in both actions is the WCF web service transformation classes to bring WCF service in and to map the data between data model class and entity class. The example code is as below

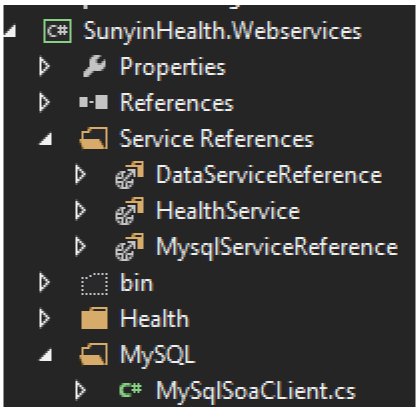
```
namespace SunyinHealth.AngularWeb.transmitter
{
   public class mysqltomodel
   {
      public List<data.OzProduct> clientOne_Ozproduct(int id)
      {
        List<OzProduct> ProductList= new List<OzProduct>();

            OzProduct clientdb = new data.OzProduct();

        var serverdb = db.GetOneOzproduct(id);

        if (serverdb != null)
        {
            clientdb.CId = serverdb.CId;
            clientdb.Desc = serverdb.Desc;
            clientdb.Discount = serverdb.Discount ;
            clientdb.Id = serverdb.Id;
            clientdb.last_updated = clientdb.last_updated ;
            clientdb.Name = clientdb.Name ;
            clientdb.Quantity = clientdb.Quantity ;
            clientdb.sku = clientdb.sku;
            clientdb.SndId = clientdb.SndId;
            clientdb.Stock = clientdb.Stock;
            clientdb.UnitPrice = clientdb.UnitPrice ;


            ProductList.Add(clientdb);
        }
        else
        {
            ProductList =  null;
        }

        return ProductList;
    }

    public List<data.OzProduct> clientAll_ozproduct()
    {

        List<OzProduct> ProductList = new List<OzProduct>();

        var serverdb = db.GetAllOzproducts();

       if (serverdb != null)
        {
            foreach (var sh in serverdb)
            {
                OzProduct clientdb = new data.OzProduct();
                clientdb.CId = sh.CId;
                clientdb.Desc = sh.Desc;
                clientdb.Discount = sh.Discount;
                clientdb.Id = sh.Id;
                clientdb.last_updated = sh.last_updated;
                clientdb.Name = sh.Name;
                clientdb.Quantity = sh.Quantity;
                clientdb.sku = sh.sku;
                clientdb.SndId = sh.SndId;
                clientdb.Stock = sh.Stock;
                clientdb.UnitPrice = sh.UnitPrice;

                ProductList.Add(clientdb);
            }

          }
          else
          {
             ProductList = null;
          }
          return ProductList ;
       }
       public void CreateOzproduct(OzProduct  t)
       {
          var tdb = OzproducttoDB.ToDBWebNew(t);
          db.CreateOzProduct(tdb);
       }
       public void DeleteOzproduct(int id)
       {
          db.DeleteOzproduct(id);
       }
       public void UpdateOzproduct(OzProduct t)
       {
          var tdb = OzproducttoDB.ToDBWebNew(t);
         db.EditOzProduct(tdb);
       }

   public static class OzproducttoDB
    {
       public static SunyinHealth.Webservices.MysqlServiceReference.oz_product
ToDBWebNew(SunyinHealth.AngularWeb.data.OzProduct t)
       {
          SunyinHealth.Webservices.MysqlServiceReference.oz_product  tdb = new
Webservices.MysqlServiceReference.oz_product();
          tdb.CId  = t.CId;
          tdb.Desc  = t.Desc;
          tdb.Discount  = t.Discount;
          tdb.Id  = t.Id ;
          tdb.last_updated = t.last_updated ;
          tdb.Name  = t.Name ;
          tdb.Quantity = t.Quantity;
          tdb.sku  = t.sku;
          tdb.SndId  = t.SndId;
          tdb.Stock = t.Stock;
          tdb.UnitPrice = t.UnitPrice;
          return tdb;
       }
}
}
```

WCF web service:

Db in resource URLS are the client of WCF web services SOA. We know the entity class from entity framework could be different from the data model class in MVC 5, Therefore, We need to map properties in both classes to make sure the data can be transferred among client and server properly. There are many techniques that can be used to achieve this. I just simply develop my own mapping class to make job done easily. Therefore, the example of WCF web service client I develop for this project is as below

```
namespace SunyinHealth.Webservices.MySQL
{
   public class MySqlSoaCLient
   {
         //ozproduct

       public List<MysqlServiceReference.oz_product> GetAllOzproducts() {
          return db.GetAllOzproducts().ToList();
       }

       public MysqlServiceReference.oz_product GetOneOzproduct(int id) {
          return db.GetoneOzproduct(id);
       }

       public void CreateOzProduct(MysqlServiceReference.oz_product t) {
          db.CreateOzProduct(t);
       }

       public void EditOzProduct(MysqlServiceReference.oz_product t)
       {
          db.EditOzProduct(t);
       }

       public void DeleteOzproduct(int id) {
          db.DeleteOzProduct(id);
       }
   }
}
```

Db In that Web service client actually is the WCF web service reference SOA see image below. The codes are very simple here. However, its background is supported by WCF web services.


```
SunyinHealth.Webservices
  Properties
  References
  Service References
    DataServiceReference
    HealthService
    MysqlServiceReference
  bin
  Health
  MySQL
    MySqlSoaCLient.cs
```

<u>WCF web service:</u>

Now we touch the core data component WCF web service. In VS 2015 people suggest to use Service Stack as a future Web Service component. However, people also believe that WCF can get survive. WCF actually is nothing but a class assembly work as an intermediate data transfer component to pass and receive data from or to client/server. We create ABC and then we configure and host them in system to use. Endpoints and connection strings need to be managed carefully or they will not work. One example web service I developed is listed below

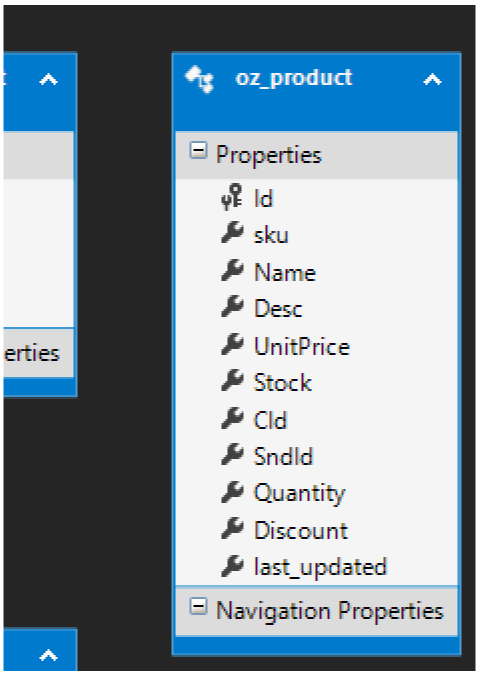
```
namespace WcfService_Mysql
{
    [ServiceContract]
   public interface IMySqlService
   {
       //ozproduct
       [OperationContract]
       List<MySQLRepository.oz_product> GetAllOzproducts();
       [OperationContract]
       MySQLRepository.oz_product GetoneOzproduct(int id);
       [OperationContract]
       void CreateOzProduct(MySQLRepository.oz_product t);
       [OperationContract]
       void EditOzProduct(MySQLRepository.oz_product t);
       [OperationContract]
       void DeleteOzProduct(int Id);
   }
}

namespace WcfService_Mysql
{
     public class MySqlService : IMySqlService
   {

       public List<MySQLRepository.oz_product> GetAllOzproducts()
       {
          return ozproduct.GetAll().ToList();
       }
       public MySQLRepository.oz_product GetoneOzproduct(int id)
       {
          return ozproduct.FindById(id);
       }
       public void CreateOzProduct(MySQLRepository.oz_product t)
       {
          ozproduct.Create(t);
       }
       public void EditOzProduct(MySQLRepository.oz_product t)
       {
          ozproduct.Edit(t);
       }
       public void DeleteOzProduct(int Id)
       {
          ozproduct.Delete(Id);
       }
   }
}
```

Ozproduct object here is the repository class for Ads List data CRUD operation. Of course, you can see the codes here are very simple. However, its background is entity data model and repository class both of them are communicate directly with SQL server databases via connection strings.

Entity Framework and Repository :

Now we know Entity Framework data model and Repository used by WCF web service can directly talks with databases in SQL server. Asp.Net really creates an excellent data model technique – entity framework to allow us to do CRUD transactions in a safe way. Famous SaveChanges() function in this model can manage the data unitofwork transaction perfectly. We can simply use data entity model to create such a model from different data sources such as SQL server quickly and maintain it separately in a happy way. We do not worry about the data transactions between data model and databases too much. What we need to do is to create a repository class that generates CRUD operations from data models. Those CRUD operations then can be injected into WCF web services.

One example code of such Entity Framework is as below

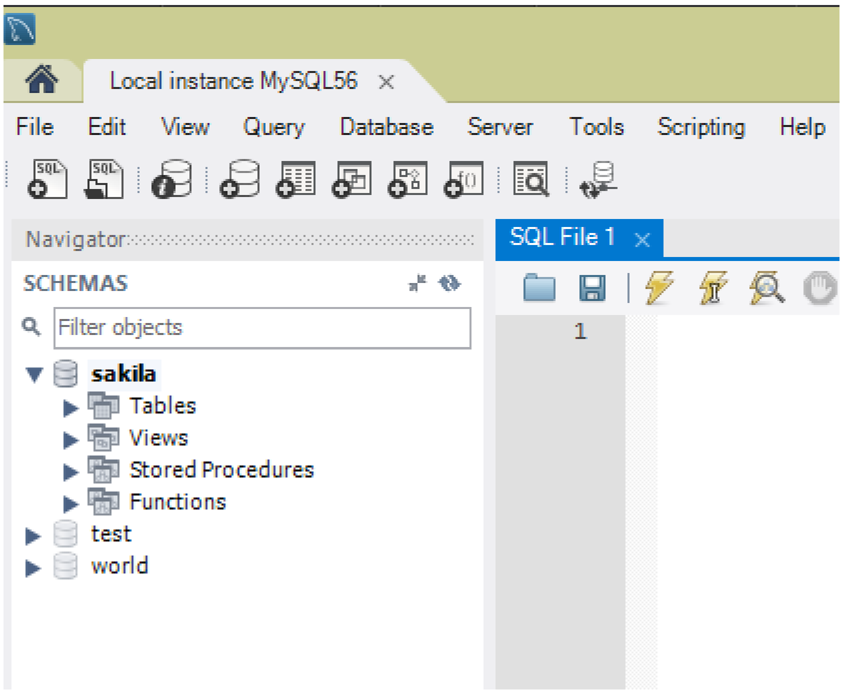


```
//------------------------------------------------------------------------------
// <auto-generated>
//    This code was generated from a template.
//
//    Manual changes to this file may cause unexpected behavior in your application.
//    Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------
namespace MySQLRepository
{
   using System;
   using System.Data.Entity;
   using System.Data.Entity.Infrastructure;

   public partial class sakilaEntities : DbContext
   {
      public sakilaEntities()
         : base("name=sakilaEntities")
      {
      }

      protected override void OnModelCreating(DbModelBuilder modelBuilder)
      {
         throw new UnintentionalCodeFirstException();
      }
       public DbSet<oz_product> oz_product { get; set; }
   }
}

namespace MySQLRepository
```

```
{
  public class OzProductsEntity : IDataEntity<oz_product>
  {

    sakilaEntities Db = new sakilaEntities();

    public List<oz_product> GetAll()
    {
      return Db.oz_product.ToList();
    }

    public void Create(oz_product t)
    {
      Db.oz_product.Add(t);
      Db.SaveChanges();
    }

    public void Edit(oz_product t)
    {
       oz_product wnlocal = Db.oz_product.First(x => x.Id == t.Id);

        wnlocal.Id =t.Id;
        wnlocal.CId =t.CId;
        wnlocal.Desc =t.Desc;
        wnlocal.Discount =t.Discount;
        wnlocal.last_updated =t.last_updated;
        wnlocal.Name =t.Name;
        wnlocal.Quantity =t.Quantity;
        wnlocal.sku =t.sku;
        wnlocal.SndId = t.SndId;
        wnlocal.Stock =t.Stock;
        wnlocal.UnitPrice =t.UnitPrice;

        Db.SaveChanges();
    }

    public void Delete(int id)
    {
      oz_product t = Db.oz_product.Find(id);
      Db.oz_product.Remove(t);
      Db.SaveChanges();
    }

    public oz_product FindById(int id)
    {
      return Db.oz_product.Find(id);
    }
  }

  public interface IDataEntity<T>
  {
        List<T> GetAll();
        void Create(T t);
        void Edit(T t);
        void Delete(int id);
        T FindById(int id);
    }
}
```

MySQL database:

The database used in Entity Framework is MySQL database and MSSQL database I love to use. One example of MySQL database I used is snapshot as below



3, Summary

This is the one example that uses Anguala r JS as front end. Another example I have not documented here is the shopping cart for health product online sale and other functions I developed such as online news evaluation from readers, etc. Web has proved that Angular JS is one of excellent front end techniques we can mix it with MVC 5 to use the benefits from .Net technology for enterprise web applications development. This brand new enterprise angular web application still is under the development based on the changes of web contents and logics. It will be hosted in the near future.