ANGULARJS by Google

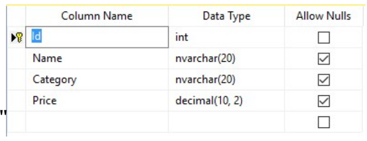
Darryl Ng
issnes@nus.edu.sg
Institute of System Science
National University of Singapore

Practitioner

Building a simple CRUD service using Angular

- DB & OR/M
- Repository/DAO implementation
- API controller
- HTML code (with Angular notations)
- Angular Code
  - Define Angular module
  - Define Angular service
  - Define Angular controller

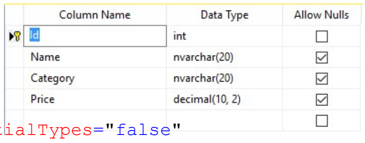
DB & OR/M – 1
Storage Schema



Column Name	Data Type	Allow Nulls
Id	int	☐
Name	nvarchar(20)	☑
Category	nvarchar(20)	☑
Price	decimal(10, 2)	☑
		☐

```
<!-- SSDL content -->
    <edmx:StorageModels>
    <Schema Namespace="SampleModel.Store" Provider="System.Data.SqlClient"
            ProviderManifestToken="2012" Alias="Self"
            xmlns:store="http://schemas.microsoft.com/ado/2007/12/edm/EntityStoreSchemaGenerator"
            xmlns:customannotation="http://schemas.microsoft.com/ado/2013/11/edm/customannotation"
            xmlns="http://schemas.microsoft.com/ado/2009/11/edm/ssdl">
        <EntityType Name="Products">
          <Key>
            <PropertyRef Name="Id" />
          </Key>
          <Property Name="Id" Type="int" StoreGeneratedPattern="Identity" Nullable="false" />
          <Property Name="Name" Type="nvarchar" MaxLength="20" />
          <Property Name="Category" Type="nvarchar" MaxLength="20" />
          <Property Name="Price" Type="decimal" Precision="10" Scale="2" />
        </EntityType>
        <EntityContainer Name="SampleModelStoreContainer">
          <EntitySet Name="Products" EntityType="Self.Products" Schema="dbo" store:Type="Tables" />
        </EntityContainer>
      </Schema></edmx:StorageModels>
```

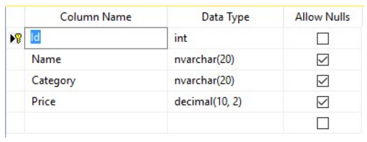
DB & OR/M – 2
Conceptual Schema



Column Name	Data Type	Allow Nulls
Id	int	☐
Name	nvarchar(20)	☑
Category	nvarchar(20)	☑
Price	decimal(10, 2)	☑
		☐

```
<!-- CSDL content -->
   <edmx:ConceptualModels>
     <Schema Namespace="SampleModel" Alias="Self" annotation:UseStrongSpatialTypes="false"
             xmlns:annotation="http://schemas.microsoft.com/ado/2009/02/edm/annotation"
             xmlns="http://schemas.microsoft.com/ado/2009/11/edm">
       <EntityContainer Name="ProductDBEntities" annotation:LazyLoadingEnabled="true">
         <EntitySet Name="Products" EntityType="SampleModel.Product" />
       </EntityContainer>
       <EntityType Name="Product">
         <Key>
           <PropertyRef Name="Id" />
         </Key>
         <Property Name="Id" Type="Int32" Nullable="false" annotation:StoreGeneratedPattern="Identity" />
         <Property Name="Name" Type="String" MaxLength="20" FixedLength="false" Unicode="true" />
         <Property Name="Category" Type="String" MaxLength="20" FixedLength="false" Unicode="true" />
         <Property Name="Price" Type="Decimal" Precision="10" Scale="2" />
       </EntityType>
     </Schema>
   </edmx:ConceptualModels>
```

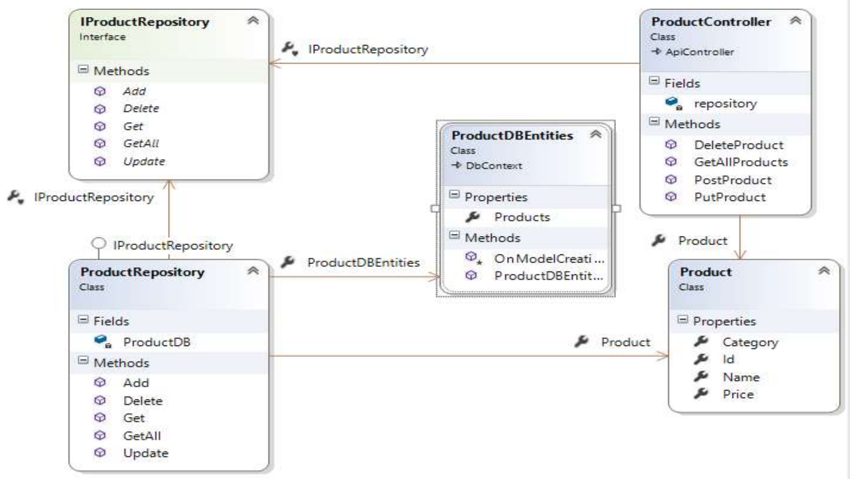
DB & OR/M – 3
Mapping Specification



	Column Name	Data Type	Allow Nulls
	Id	int	☐
	Name	nvarchar(20)	☑
	Category	nvarchar(20)	☑
	Price	decimal(10, 2)	☑
			☐

```
<!-- C-S mapping content -->
    <edmx:Mappings>
      <Mapping Space="C-S"
xmlns="http://schemas.microsoft.com/ado/2009/11/mapping/cs">
        <EntityContainerMapping
StorageEntityContainer="SampleModelStoreContainer"
CdmEntityContainer="ProductDBEntities">
          <EntitySetMapping Name="Products">
            <EntityTypeMapping TypeName="SampleModel.Product">
              <MappingFragment StoreEntitySet="Products">
                <ScalarProperty Name="Price" ColumnName="Price" />
                <ScalarProperty Name="Category" ColumnName="Category" />
                <ScalarProperty Name="Name" ColumnName="Name" />
                <ScalarProperty Name="Id" ColumnName="Id" />
              </MappingFragment>
            </EntityTypeMapping>
          </EntitySetMapping>
        </EntityContainerMapping>
      </Mapping>
    </edmx:Mappings>
```

Class Diagram Mapping



Repository/DAO implementation - 1

```
using AngularJS_Demo.Interface;
using AngularJS_Demo.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace AngularJS_Demo.Repositories
{

    public class ProductRepository : IProductRepository
    {
        ProductDBEntities ProductDB = new ProductDBEntities();
. . .
}
```

Repository/DAO implementation - 2

```
public IEnumerable<Product> GetAll()
{
    // TO DO : Code to get the list of all the records in database
    return ProductDB.Products;
}

public Product Get(int id)
{
    // TO DO : Code to find a record in database
    return ProductDB.Products.Find(id);
}
```

Repository/DAO implementation - 3

```
public Product Add(Product item)
{
    if (item == null)
    {
        throw new ArgumentNullException("item");
    }

    // TO DO : Code to save record into database
    ProductDB.Products.Add(item);
    ProductDB.SaveChanges();
    return item;
}
```

Repository/DAO implementation - 4

```
public bool Update(Product item)
{
    if (item == null)
    {
        throw new ArgumentNullException("item");
    }

    // TO DO : Code to update record into database
    var products = ProductDB.Products.Single(a => a.Id == item.Id);
    products.Name = item.Name;
    products.Category = item.Category;
    products.Price = item.Price;
    ProductDB.SaveChanges();

    return true;
}
```

Repository/DAO implementation - 5

```
public bool Delete(int id)
{
    // TO DO : Code to remove the records from database
    Product products = ProductDB.Products.Find(id);
    ProductDB.Products.Remove(products);
    ProductDB.SaveChanges();
    return true;
}
```

Repository/DAO implementation – 6 Repository Interface

```
using AngularJS_Demo.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace AngularJS_Demo.Interface
{
    interface IProductRepository
    {
        IEnumerable<Product> GetAll();
        Product Get(int id);
        Product Add(Product item);
        bool Update(Product item);
        bool Delete(int id);
    }
}
```

Repository Controller - 1

```
using AngularJS_Demo.Repositories;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using AngularJS_Demo.Models;
using System.Web.Http;

namespace AngularJS_Demo.Controllers
{
    public class ProductController : ApiController
    {
        static readonly IProductRepository repository = new ProductRepository();
. . .
    }
}
```

Repository Controller - 2

```
public IEnumerable GetAllProducts()
{
    return repository.GetAll();
}

public Product PostProduct(Product item)
{
    return repository.Add(item);
}
```

Repository Controller - 3

```
public IEnumerable PutProduct(int id, Product product)
{
    product.Id = id;
    if (repository.Update(product))
    {
        return repository.GetAll();
    }
    else
    {
        return null;
    }
}
```

Repository Controller - 4

```
public bool DeleteProduct(int id)
{
    if (repository.Delete(id))
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Route Configuration

RouteConfig

```
 routes.MapRoute(
                 name: "Default",
                 url: "{controller}/{action}/{id}",
                 defaults: new { controller = "Home", action = "Product", id = UrlParameter.Optional
}
```

WebApiConfig

```
 config.Routes.MapHttpRoute(
                 name: "DefaultApi",
                 routeTemplate: "api/{controller}/{action}/{id}",
                 defaults: new { id = RouteParameter.Optional }
             );
```

HTML Code - 1

```
@{
    ViewBag.Title = "Products List";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
@section scripts {

    <link href="~/Content/CustomStyle.css" rel="stylesheet" />
    <script src="~/Scripts/jquery-1.10.2.min.js"></script>
    <script src="~/Scripts/angular.js"></script>
    <script src="~/Scripts/AngularDemo.js"></script>
}

<div ng-app="demoModule" id="body">
    <div ng-controller="demoCtrl">
. . .
 </div>
</div>
```

HTML Code - 2

```
<h2>AngularJS CRUD Operations with MVC5 WebAPI</h2>
<h3>List of Products</h3>
<table ng-cloak>
    <thead>
        <tr>
            <th style="display: none;">ID</th>
            <th>Name</th><th>Category</th><th>Price</th><th>Actions</th>
        </tr>
    </thead>
    <tbody>
        <tr ng-repeat="items in productsData">
            <td hidden="hidden">{{items.Id}}</td>
            <td>{{items.Name}}</td>
            <td>{{items.Category}}</td>
            <td>{{items.Price | currency:'&#8377;':2}}</td>
            <td>
                <button ng-model="$scope.Product" ng-click="edit(productsData[$index])">Edit</button>
                <button ng-click="delete($index)">Delete</button>
            </td>
        </tr>
    </tbody>
    <tfoot> . . .
```

HTML Code - 3

```
 <div ng-show="Product.Id != '' ">
    <div><h2>Update Product</h2></div>
    <div hidden="hidden">
        <label for="id">Id</label>
        <input type="text" data-ng-model="Product.Id" />
    </div>
    <div>
        <label for="name">Name</label>
        <input type="text" data-ng-model="Product.Name" />
    </div>
      <div>
        <label for="category">Category</label>
        <input type="text" data-ng-model="Product.Category" />
    </div>
     <div>
        <label for="price">Price</label>
        <input type="text" data-ng-model="Product.Price" />
    </div>
    <br />
    <div>
        <button data-ng-click="update()">Update</button>
        <button data-ng-click="cancel()">Cancel</button>
    </div>
</div>
```

HTML Code - 4

```
 <div ng-hide="Product.Id != '' ">
    <div>
        <h2>Add New Product</h2>
    </div>
    <div>
        <label for="name">Name</label>
        <input type="text" data-ng-model="Product.Name" />
    </div>
    <div>
        <label for="category">Category</label>
        <input type="text" data-ng-model="Product.Category" />
    </div>
    <div>
        <label for="price">Price</label>
        <input type="text" data-ng-model="Product.Price" />
    </div>
    <br />
    <div>
        <button data-ng-click="save()">Save</button>
        <button data-ng-click="clear()">Clear</button>
    </div>
</div>
```

Angular Code – 1 Module

```
// Defining angularjs module
var app = angular.module('demoModule', []);
```

Angular Code – 2
Service/Factory

```
// Here I have created a factory which is a populer way to create and configure services.
// You may also create the factories in another script file which is best practice.
// You can also write above codes for POST,PUT,DELETE in this factory instead of controller,
// so that our controller will look clean and exhibits proper Separation of Concern.
app.factory('ProductsService', function ($http) {
    var fac = {};
    fac.GetAllRecords = function () {
        return $http.get('api/Product/GetAllProducts');
    }
    return fac;
});
```

Angular Code – 3
Controller - 1

```
// Defining angularjs Controller and injecting ProductsService
app.controller('demoCtrl', function ($scope, $http, ProductsService) {

    $scope.productsData = null;
    // Fetching records from the factory created at the bottom of the script file
    ProductsService.GetAllRecords().then(function (d) {
        $scope.productsData = d.data; // Success
    }, function () {
        alert('Error Occured !!!'); // Failed
    });

. . .
});
```

Angular Code – 3
Controller - 2

```
// Calculate Total of Price After Initialization
   $scope.total = function () {
       var total = 0;
       angular.forEach($scope.productsData, function (item) {
           total += item.Price;
       })
       return total;
   }

   $scope.Product = {
       Id: '',
       Name: '',
       Price: '',
       Category: ''
   };
```

Angular Code – 3
Controller - 3

```
// Reset product details
   $scope.clear = function () {
       $scope.Product.Id = '';
       $scope.Product.Name = '';
       $scope.Product.Price = '';
       $scope.Product.Category = '';
   }
```

Angular Code – 3 Controller - 4

```
//Add New Item
   $scope.save = function () {
       if ($scope.Product.Name != "" &&
   $scope.Product.Price != "" && $scope.Product.Category != "") {
           // Http request using $http
           $http({
               method: 'POST',
               url: 'api/Product/PostProduct/',
               data: $scope.Product
           }).then(function successCallback(response) {
               // this callback will be called asynchronously
               // when the response is available
               $scope.productsData.push(response.data);
               $scope.clear();
               alert("Product Added Successfully !!!");
           }, function errorCallback(response) {
               // called asynchronously if an error occurs
               // or server returns response with an error status.
               alert("Error : " + response.data.ExceptionMessage);
           });
       }
       else {
           alert('Please Enter All the Values !!');
       }
   };
```

Angular Code – 3 Controller - 5

```
// Edit product details
    $scope.edit = function (data) {
        $scope.Product = { Id: data.Id, Name: data.Name, Price:
data.Price, Category: data.Category };
    }

    // Cancel product details
    $scope.cancel = function () {
        $scope.clear();
    }
```

Angular Code – 3 Controller - 6

```
// Update product details
    $scope.update = function () {
        if ($scope.Product.Name != "" &&
       $scope.Product.Price != "" && $scope.Product.Category != "") {
            $http({
                method: 'PUT',
                url: 'api/Product/PutProduct/' + $scope.Product.Id,
                data: $scope.Product
            }).then(function successCallback(response) {
                $scope.productsData = response.data;
                $scope.clear();
                alert("Product Updated Successfully !!!");
            }, function errorCallback(response) {
                alert("Error : " + response.data.ExceptionMessage);
            });
        }
        else {
            alert('Please Enter All the Values !!');
        }
    };
```

Angular Code – 3 Controller - 7

```
// Delete product details
   $scope.delete = function (index) {
       $http({
           method: 'DELETE',
           url: 'api/Product/DeleteProduct/' + $scope.productsData[index].Id,
       }).then(function successCallback(response) {
           $scope.productsData.splice(index, 1);
           alert("Product Deleted Successfully !!!");
       }, function errorCallback(response) {
           alert("Error : " + response.data.ExceptionMessage);
       });
   };
```

캄샤 하미다