SYED USMAN

0413.680.650

[Syedusman\_1@gmail.com](mailto:Syedusman_1@gmail.com)

**ASSIGNMENT: TO DEVELP A SAMPLE FRAMEWORK**

As advised I created a sample web based framework to support Client – Server development. I used Visual studio 2015 and LOCAL DB to develop this. I have created a total of 8 projects in the solution and feel it could be a great starting point to support the small to medium based application architecture. The development can do a rapid development on this Seed Project.

# Program Execution Instructions

1. Please download the code from the GitHub
2. There is no nugget or database in the source code
3. Upon build nuggets are restored
4. This project is made with code first approach so running the first time project will create a new database. At the moment database is being created it in App\_Data, however the connection string can be changed from web.config

# Framework Methodology:

1. I have created a separate Client Project. In this project there are only HTML pages, **Angular framework**, bootstrap and other helping JS files. We can deploy the client project separately from Server project. We provide the server url in angular constant service.
2. Server side is being developed using WEBAPI project empty template. We have to enable “CORS” as the client could be deployed on a different url.

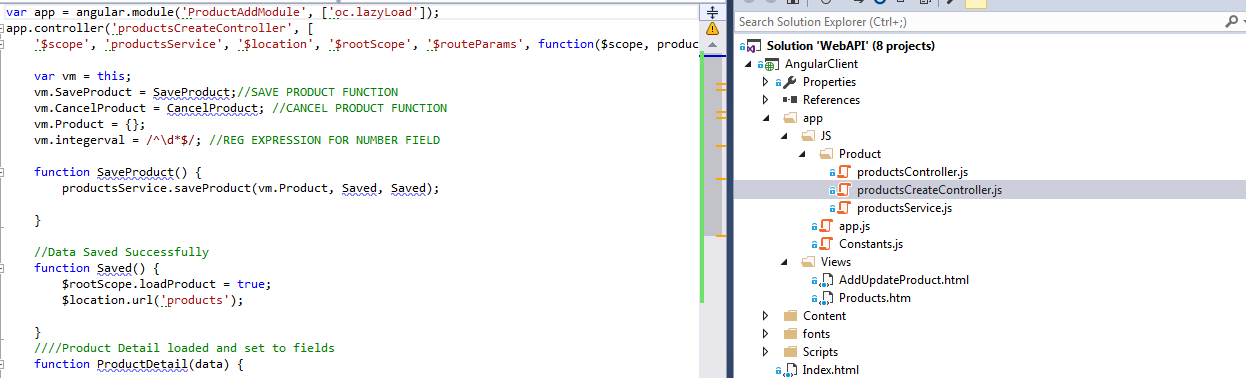
# Design Patterns

1. IOC: using Dependency injection (Microsoft.unity.WebAPI)
2. Generic Repository Pattern
3. MV\* at client side (angular)
4. MVC at server side

# Implementation Details:

## Client Side:

1. Used AngularJS:
   1. Routing
   2. Interceptors: to show busy indicator in a generic way, between http request is being sent and a response arrive
   3. Constants Service
   4. HttpProvider
   5. Lazyloading using ocLazyLoader
2. Bootstrap: use it and glyphicons to design a simple responsive layout. WE can extend it or incorporate new theme.
3. Client Side Structure:



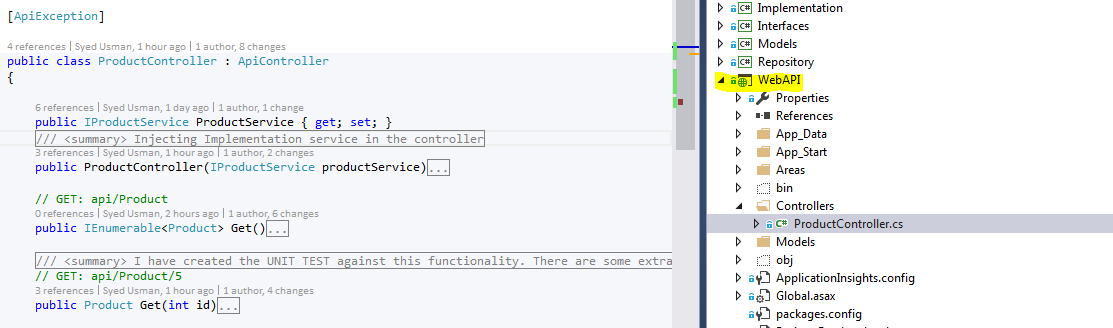
Developers will know that they have to create separate client side folders for the entities JS and HTML pages (as shown above). They will create \*\_Controller.js, \*\_Service.js and \*\_html page to work on an entity.

## Server Side:

Following projects are created at Server side:

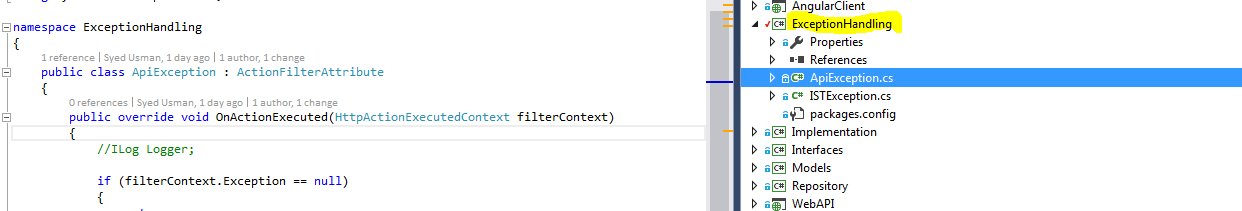
### WebAPI:

Handles web request, define RESTFUL API Controllers.



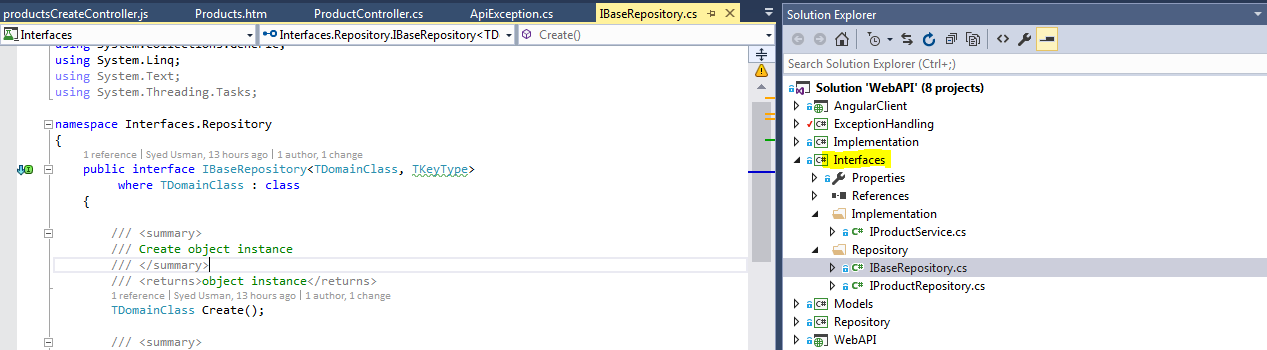
### ExceptionHandling

Use ActionFilters to write exception handling. We can use logging as well inside. Due to time constraint I only wrote the foundation code. We will use attributes on the top of controllers where we want to do Exception handling.



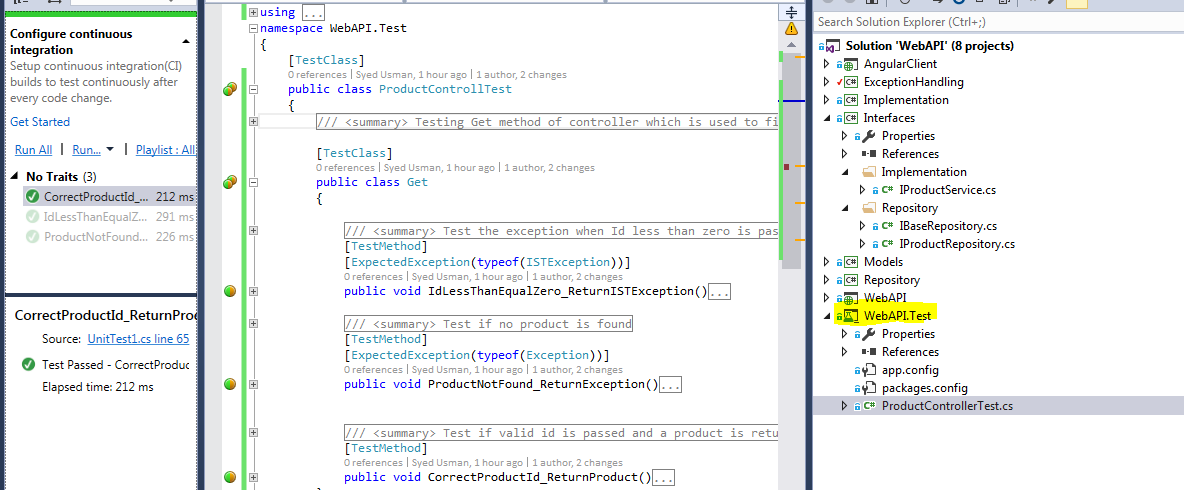
### Interfaces

To support IOC. I have created interfaces for Implementation and Repository services



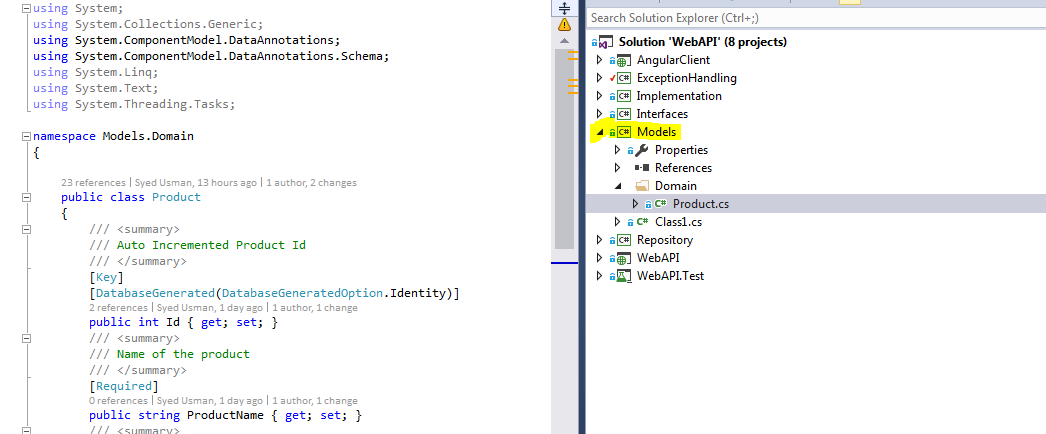
### Unit Test (MSTEST)

Unit test the project. Currently I have made a POC for Product->WEBAPIController->Detail



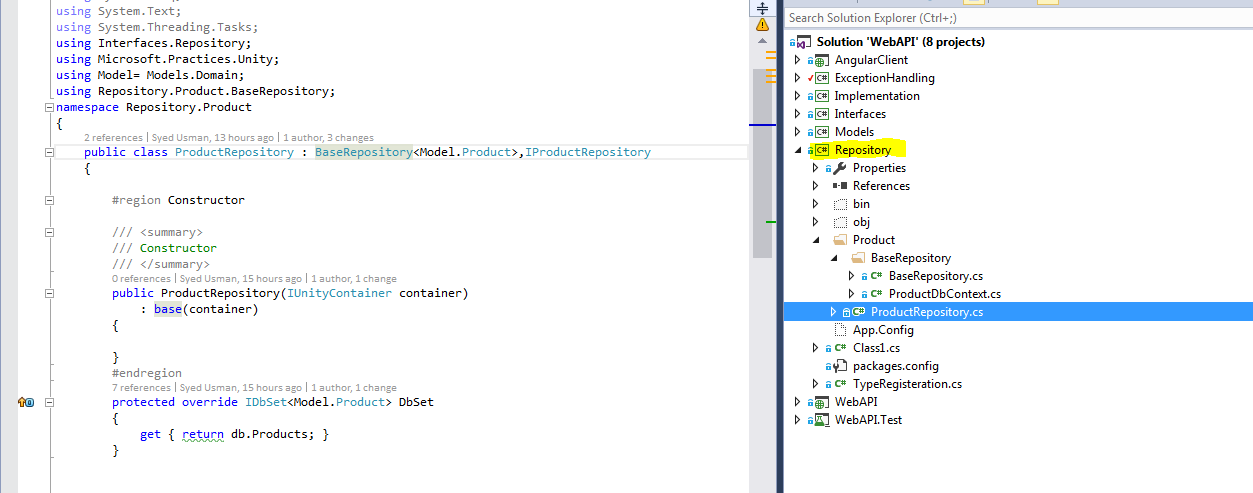
### Models

Defines the model/db entities of our system. We have used Code First.



### Repository

Defines the generic repository pattern and Repository classes for each entity. System repositories are inherited from BaseRepository.



### Implementation

The developers will be writing any business checks in this layer. These could be like unique product name etc.

