**Alinta Code Test**

# Design

This solution follows one of basic micro service architectural pattern (Fig 1). Once service is consumes by another service directly. This solution can be easily extendible as per Fig 2.

Web Service 1 (CodeTest.API) – The newly created service to consume the Web service 2 (<http://alintacodingtest.azurewebsites.net>) and return the list of Actors with their movie characters. The characters are sorted by movie name

Web service 1

Web Service 2

CodeTest.API

http://alintacodingtest.azurewebsites.net

MovieDTO

ActorDTO

Fig 1 : Current solution architecture

Desktop Client

Web Client

Mobile Client

API

Web service 1

Web Service 2

CodeTest.API

http://alintacodingtest.azurewebsites.net

Web Service 3

Fig 2: Extendible solution architecture

Actor

Movie Character

Movie

1

1..\*

1..\*

1

Fig 3: Class diagram

# Implementation

Project Specification:

* IDE: Visual Studio 2015
* .Net Framework 4.5+
* Technologies: C#, Web API
* Unit test framework: MS Test
* Tools:
  + SpecFlow for BDD scripts (http://specflow.org/getting-started/)
  + Swashbuckle for Swagger integration with the solution
  + AutoMapper for object to object mapping
  + Newtonsoft.json for JSON/Object serialization and deserialization

The implementation of this solution follows,

* Object oriented programming
* Single responsibility principle (from SOLID design pattern)
* Test driven development
* Behavior driven development
* Restful API programming

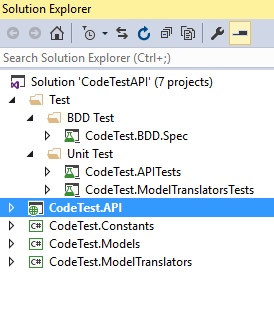


Fig 4: Packages structure

# Tests

The solution is implemented using BDD and TDD combined approach.

## 3.1 BDD Tests

BDD tests were captured using SpecFlow specs.

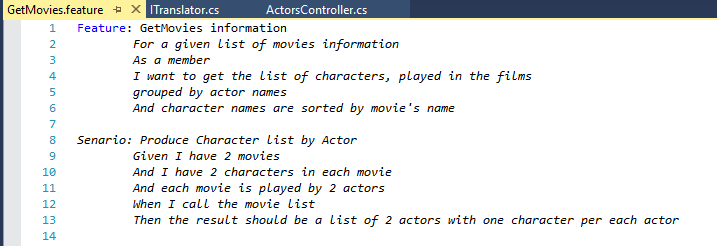


Fig: A SpecFlow feature file with one scenario. One feature can have many senarios.

## 3.2 Unit Tests

Unit tests have been created to test several layers of this application. These unit tests follow the best practices: AAA(Arrange/Act/Assert) standards.

This project is implemented using Test driven development practice. The approach is followed,

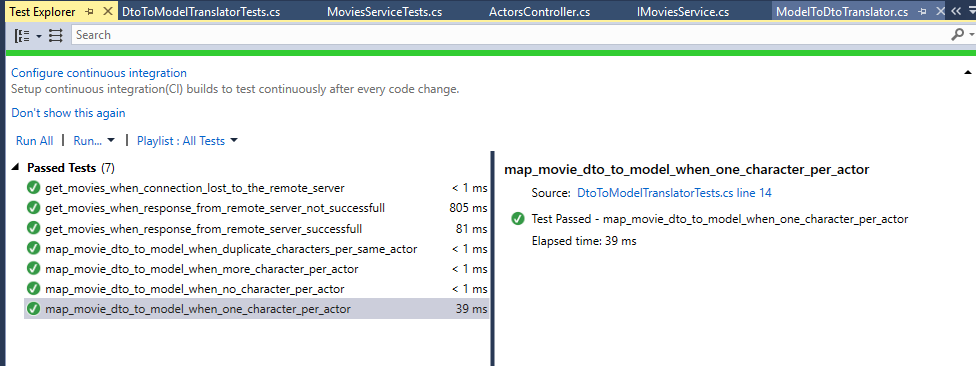
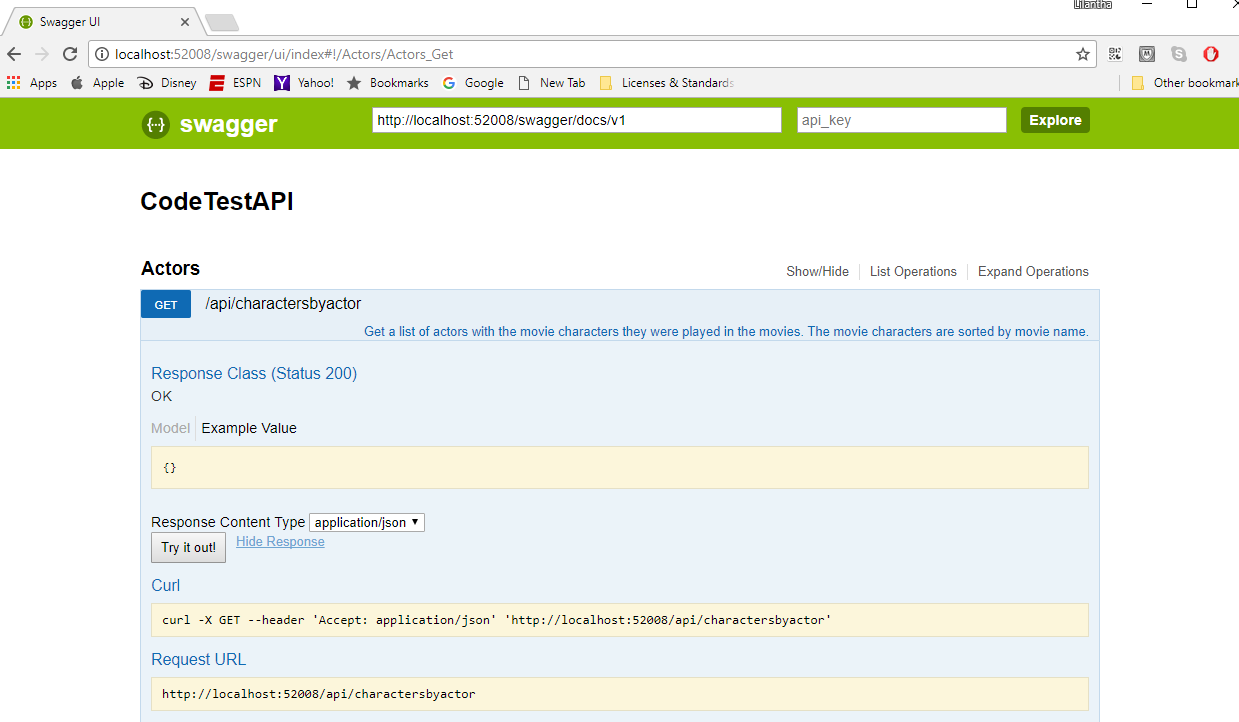
* Create unit tests
* Implementing code for each test
* Refining the code implementation
* Do any adjustment to the test

Fig: Total number of unit tests in the solution

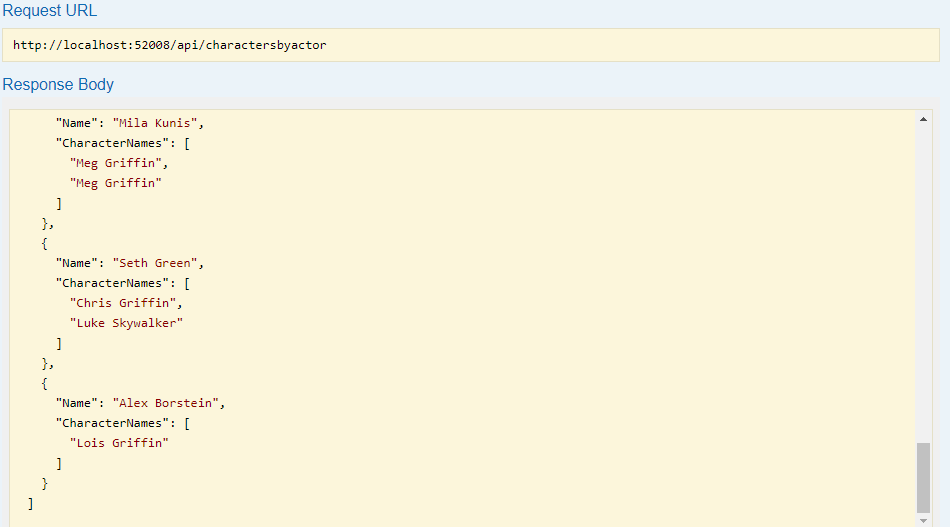
## 3.3 Test Results

CodeTest.API can be tested via swagger api which is bind to this solution.

Some of test results are captured here.







# Extendibility

