# Overview

This document provides the information to understand the unit testing approach and its implementation for testing CodingTest for GreenFinch project. It covers data flow in details, the advantages and workaround in unlikely scenarios.

# Design Architecture

1. Data flow for testing service classes :



1. Data flow for testing web api controllers :

****

# About data flow steps

1. Creating controller instance

A controller instance is created in **ClassInit**() method of a <ControllerName>ControllerTest class so that the same instance can be used for all test case methods also the dependency mock objects also get created only once. For each test method, **only test data gets reset** in the **TestInit**() method.

1. Generating Mock Manager1
   1. If the repository1’s **readonly collection** is empty, load the test data from the repository1 **test data file** and populate repository1 **data collection**.
2. Load Repository1 data from file
   1. Read test data file
   2. store the data in the repository1 **readonly** collections
   3. Populate local data collection from repository1 **readonly** collection.
3. Generating Mock Manager2 ( same as step 2)
   1. If the repository2’s **readonly collection** is empty, load the test data from the repository2 **test data file** and populate repository2 **data collection**.
4. Load Repository2 data from file (same as step 3)
   1. Read test data file
   2. store the data in the repository2 **readonly** collections
   3. Populate local data collection from repository2 **readonly** collection.
5. Call the controller instance in test cases
   1. The controller calls a manager class’s mock method.
   2. The mock method uses test data from repository data collection to generated expected output.

# Mock object generator classes

All **ManagerGenerator** and **RepositoryGenerator** classes has mock all/required methods so that the same mock object can be used throughout the test cases. Here is the overview for the mock generator classes:

** **

EmptyDataCollection & Empty<Entity>DataCollection – removes all the items of the **<entity>dataCollection** List.

GetDataCollection & Get<Entity>DataCollection – return the **<entity>dataCollection** list.

GetMockService & GetMockRepository – creates mock methods for **Service** class and **Repository** Class respectively. These methods use the **<entity>dataCollection** list for returning expected result.

GetReadOnlyData & GetRead<Entity>OnlyDaya – reads the data from a data file (in this case **<Entity>.js** file e.g. NewsLetter.js) and also populate the **<entity>dataCollection** List from the **readonly** Collection (**MockData.Collections.<Enity>s)**.

ResetDataCollection & Reset<Entity>DataCollection – removes old all items and repopulates the **dataCollection** list from **Readonly** Collection (in this case it is **MockData.Collections. SampleDatas.**

# Advantages

* Provides standard and consistency approach for writing test cases.
* Mock Generator classes can be reused since it is not coupled to any test method and test classes.
* It saves lots of repeated code so lesser code for each test case.
* Test cases execution times decreases since no need to create mock object for each test case.

# Workarounds

If because of some reasons if you cannot use an existing mock method, then you can override that method in a test case method;

The following method is part of mock generator class which returns list of newsletters,

mockRepository.Setup(a => a.GetNewsLettersByIds(It.IsAny<List<long>>())).Returns<List<long>>(newsLetterIds =>

{

return dataCollection.Where(o => newsLetterIds.Contains(o.Id)).ToList();

});

but if you could not able to use this method for testing a particular test case and you want to generate a different output, then you can redefine the output of the mock method in the particular test case .

e.g.

mockRepository.Setup(a => a.GetNewsLettersByIds(It.IsAny<List<long>>())).Returns<List<long>>(newsLetterIds =>

{

return new NewLetterDetailsModel{};

});

This override will not affect output of other test case methods.