Documentation of SW star ship code challenge

Author: SIMILIEN Michel Fils

Resume

Throughout this documentation we will explain how the development of this application using the SWAPI API to calculate the stop number for each "StarShip" is done. The .Net technology is used to create an MVC application that consumes the Api.

all the architecture, the structure, the technical aspects and the unit test part are explained in the different chapters of this document.

Index

1. APPLICATION SET UP

This is an ASP .Net MVC application developed on **visual studio 2013** to consume an API provided by SWAPI.

Install HttpClient library

After creating an empty MVC application the first thing i do is to install **HttpClient library.**

I use HttpClient to consume the Web API REST Service, so i need to install this library from NuGet Package Manager .

HttpClient is base class which is responsible to send HTTP request and receive HTTP response resources i.e from REST services.

Install WebAPI.Client

I also install WebAPI.Client library from NuGet . This package is used for formatting and content negotiation which provides support for System.Net.Http.

2. STRUCTURE, ARCHITECTURE AND DEVELOPMENT OF THE APPLICATION

Structure and Architecture

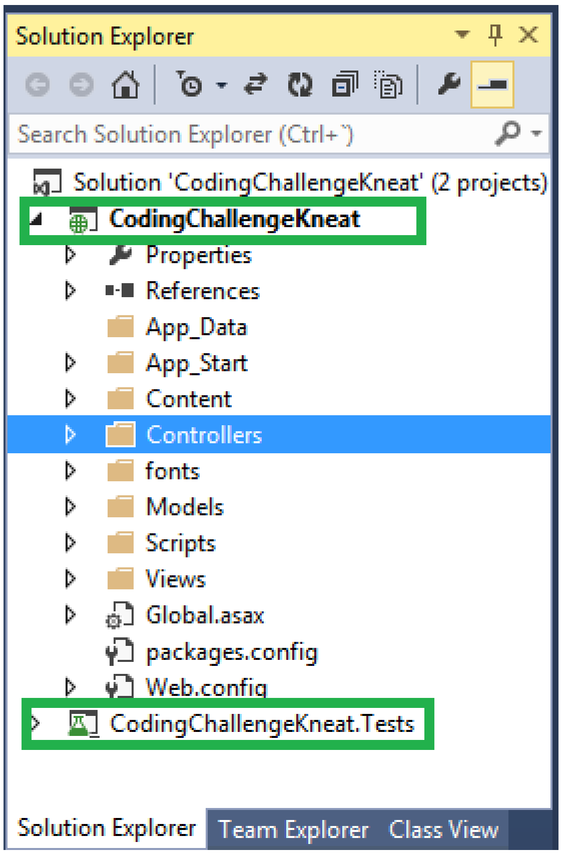
The application contains one solution with 2 projects:

- CodingChallengeKneat
- CodingChallengeKneat.Test

The CodingChallengeKneat is the main MVC project that contain all the calculation methods after calling the API by the controller.

The CodingChallengeKneat.Test is used to generate a unit test for the controller methods.

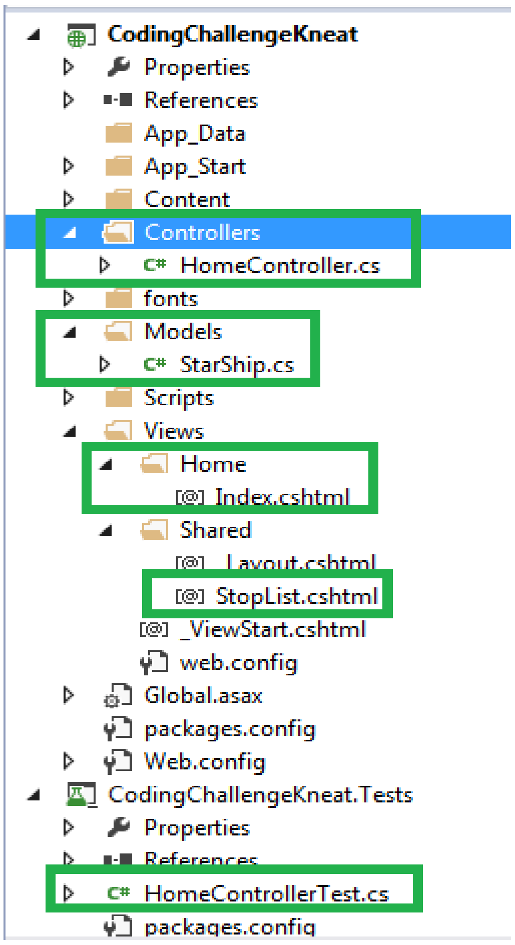
The architecture of the application is presented as follows:


Solution Explorer

Search Solution Explorer (Ctrl+`)

```
Solution 'CodingChallengeKneat' (2 projects)
  CodingChallengeKneat
    Properties
    References
    App_Data
    App_Start
    Content
    Controllers
    fonts
    Models
    Scripts
    Views
    Global.asax
    packages.config
    Web.config
  CodingChallengeKneat.Tests
```

Solution Explorer | Team Explorer | Class View



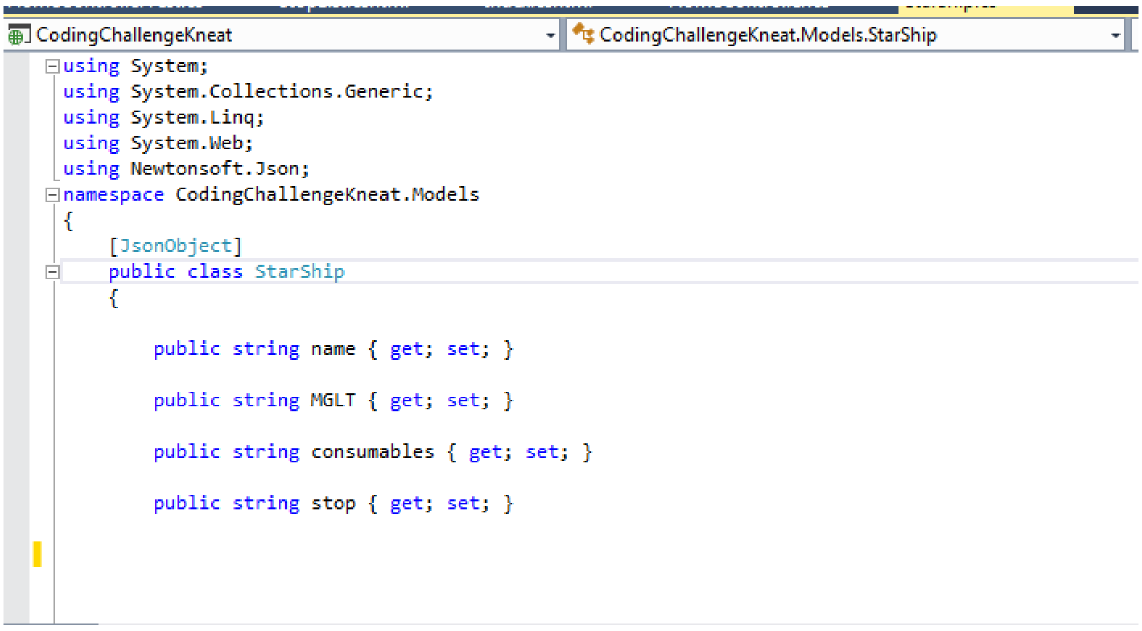
```
CodingChallengeKneat
  Properties
  References
  App_Data
  App_Start
  Content
  Controllers
    HomeController.cs
  fonts
  Models
    StarShip.cs
  Scripts
  Views
    Home
      Index.cshtml
    Shared
      _Layout.cshtml
      StopList.cshtml
    _ViewStart.cshtml
    web.config
  Global.asax
  packages.config
  Web.config
CodingChallengeKneat.Tests
  Properties
  References
  HomeControllerTest.cs
  packages.config
```

Development of the application

Models

The model class StarShip.cs is created with some property that we will get from the API to chow on the index view. So i don't create this class with all values that return the API.

The code snippet of created StarShip.cs class look like this.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Newtonsoft.Json;
namespace CodingChallengeKneat.Models
{
    [JsonObject]
    public class StarShip
    {

        public string name { get; set; }

        public string MGLT { get; set; }

        public string consumables { get; set; }

        public string stop { get; set; }
```

Controller

This project have one controller : Home Controller. In this they are 4 methods.

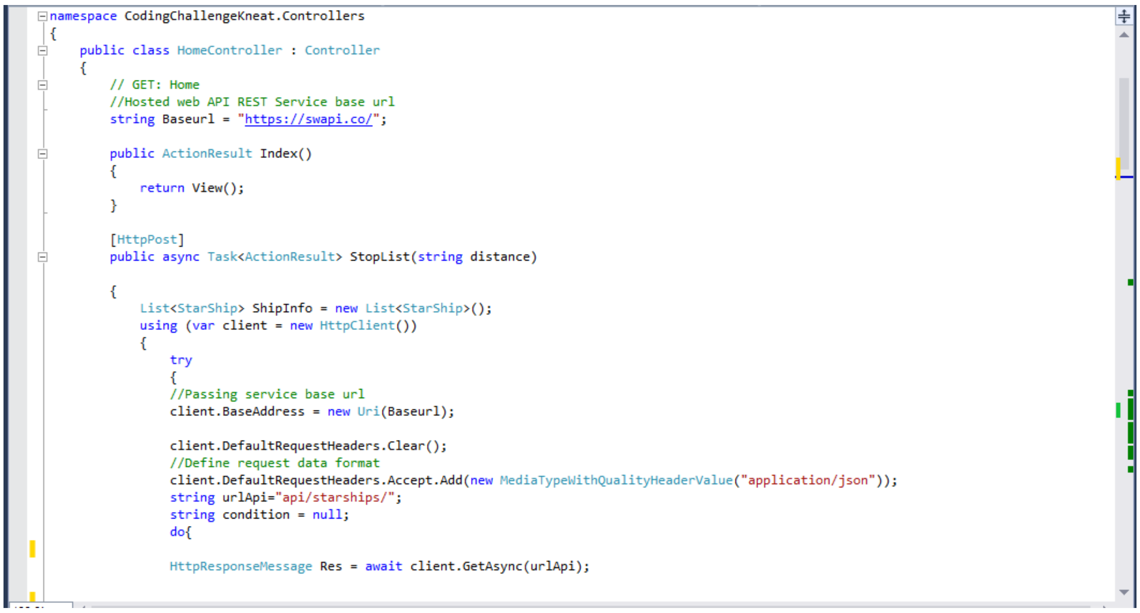
The first more important method is an async HttpPost method that receive a given distance string and return a partial View with the Model.

The call of the API is done in the method.

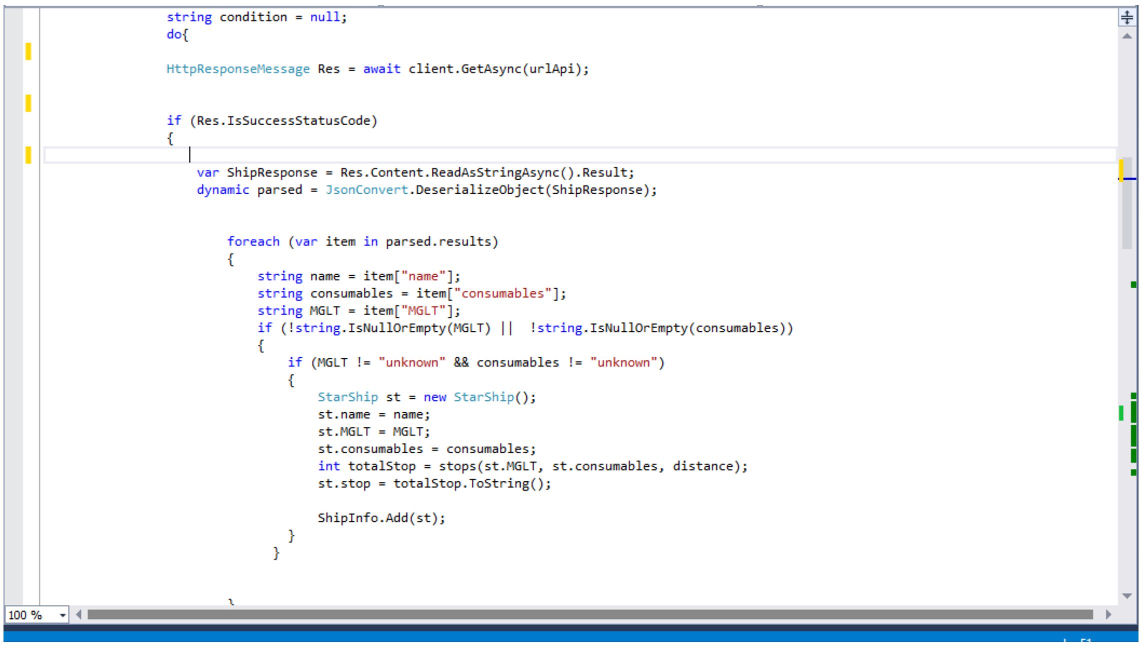
The base url to call the StarShips is: https://swapi.co/api/ starships/. So this call return only the first page json file. To get all json file page I had to make a call in loop while.

For each page I retrieved the name, the MGLT and the consumable. The MGLT, consumable and the given distance are passed as parameters to a method named “stop”.

The code snippet of all the controller method look like this:

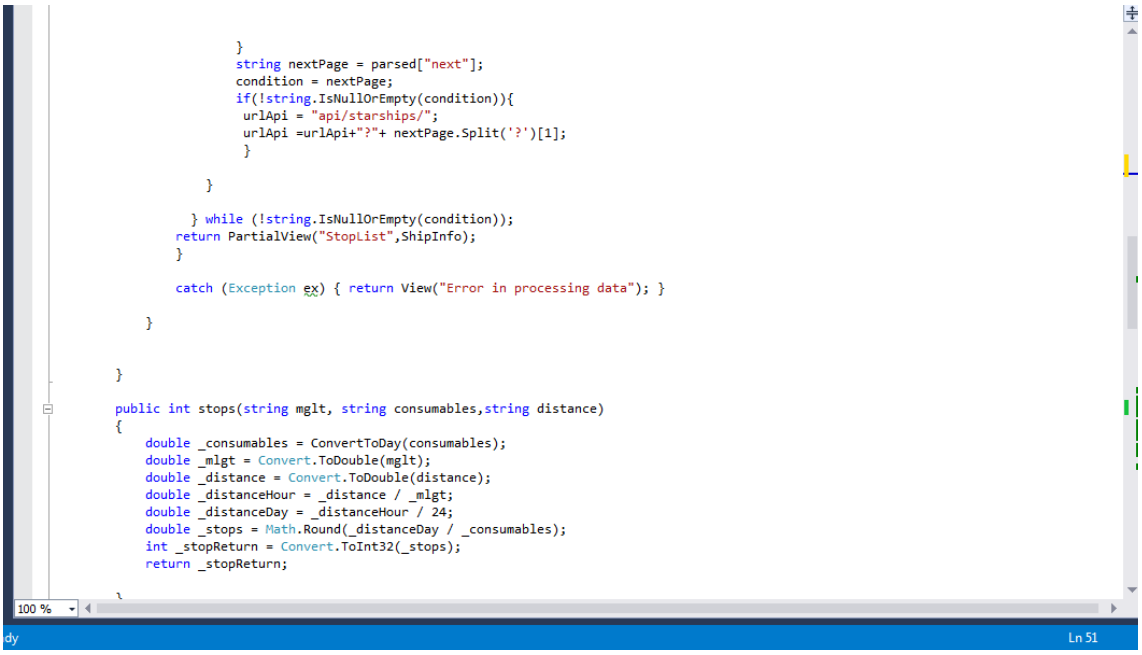


```
namespace CodingChallengeKneat.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        //Hosted web API REST Service base url
        string Baseurl = "https://swapi.co/";

        public ActionResult Index()
        {
            return View();
        }

        [HttpPost]
        public async Task<ActionResult> StopList(string distance)

        {
            List<StarShip> ShipInfo = new List<StarShip>();
            using (var client = new HttpClient())
            {
                try
                {
                //Passing service base url
                client.BaseAddress = new Uri(Baseurl);

                client.DefaultRequestHeaders.Clear();
                //Define request data format
                client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
                string urlApi="api/starships/";
                string condition = null;
                do{

                HttpResponseMessage Res = await client.GetAsync(urlApi);
```



```
                string condition = null;
                do{

                HttpResponseMessage Res = await client.GetAsync(urlApi);


                if (Res.IsSuccessStatusCode)
                {

                    var ShipResponse = Res.Content.ReadAsStringAsync().Result;
                    dynamic parsed = JsonConvert.DeserializeObject(ShipResponse);


                        foreach (var item in parsed.results)
                        {
                            string name = item["name"];
                            string consumables = item["consumables"];
                            string MGLT = item["MGLT"];
                            if (!string.IsNullOrEmpty(MGLT) ||  !string.IsNullOrEmpty(consumables))
                            {
                                if (MGLT != "unknown" && consumables != "unknown")
                                {
                                    StarShip st = new StarShip();
                                    st.name = name;
                                    st.MGLT = MGLT;
                                    st.consumables = consumables;
                                    int totalStop = stops(st.MGLT, st.consumables, distance);
                                    st.stop = totalStop.ToString();

                                    ShipInfo.Add(st);
                                }
                              }
```

100 %



```
                    }
                    string nextPage = parsed["next"];
                    condition = nextPage;
                    if(!string.IsNullOrEmpty(condition)){
                     urlApi = "api/starships/";
                     urlApi =urlApi+"?"+ nextPage.Split('?')[1];
                     }

                }

              } while (!string.IsNullOrEmpty(condition));
            return PartialView("StopList",ShipInfo);
            }

            catch (Exception ex) { return View("Error in processing data"); }

        }


    }

    public int stops(string mglt, string consumables,string distance)
    {
        double _consumables = ConvertToDay(consumables);
        double _mlgt = Convert.ToDouble(mglt);
        double _distance = Convert.ToDouble(distance);
        double _distanceHour = _distance / _mlgt;
        double _distanceDay = _distanceHour / 24;
        double _stops = Math.Round(_distanceDay / _consumables);
        int _stopReturn = Convert.ToInt32(_stops);
        return _stopReturn;
```

Views

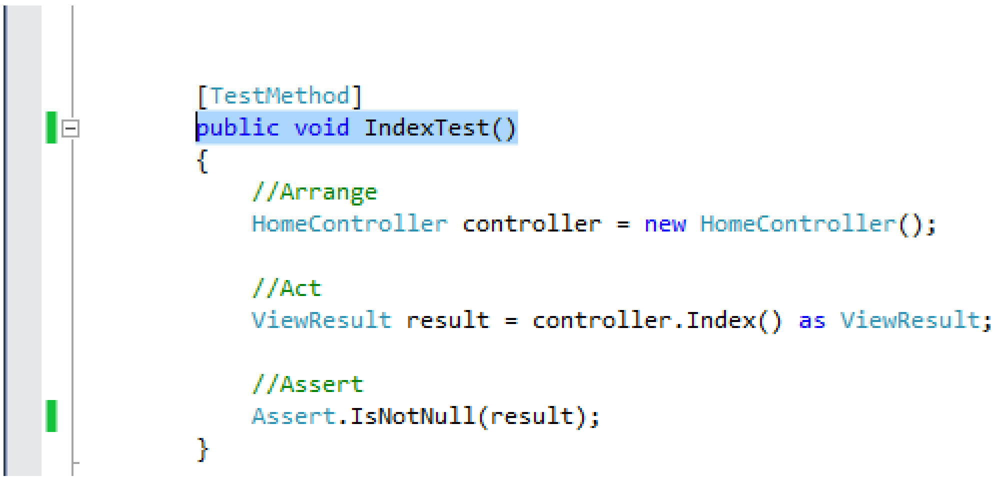
There are only 2 views: the first is an index view that the user can send the distance and the other is a partial view that show the result.

3. UNIT TEST

The main goal of the unit test is to assert that all the method in the controller work right.

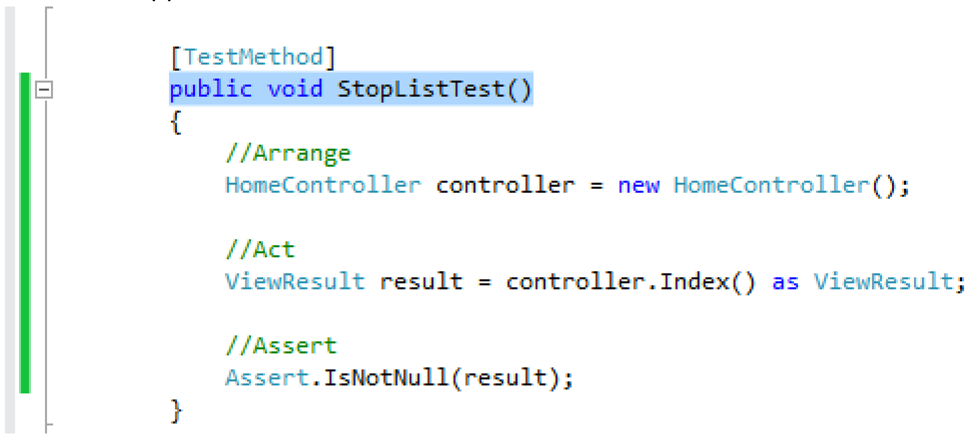
So the unit test will run for 4 methods that contain the controller.

1- `public void IndexTest()` test the view return by the controller and this code snippet look like this:



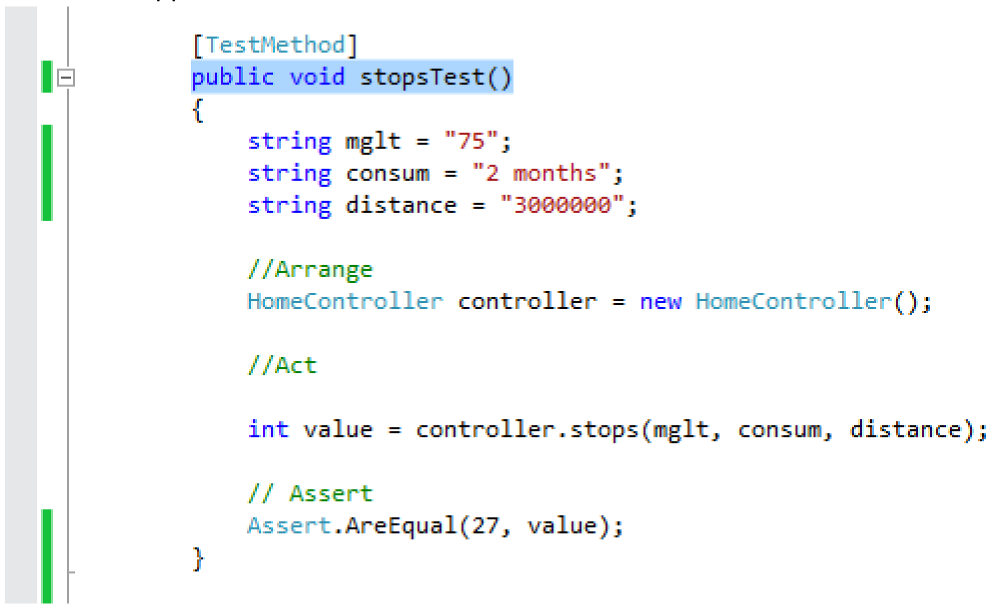
```
[TestMethod]
public void IndexTest()
{
    //Arrange
    HomeController controller = new HomeController();

    //Act
    ViewResult result = controller.Index() as ViewResult;

    //Assert
    Assert.IsNotNull(result);
}
```

2- `public void StopListTest()` `test the partial view` return by the controller and the code snippet look like this:



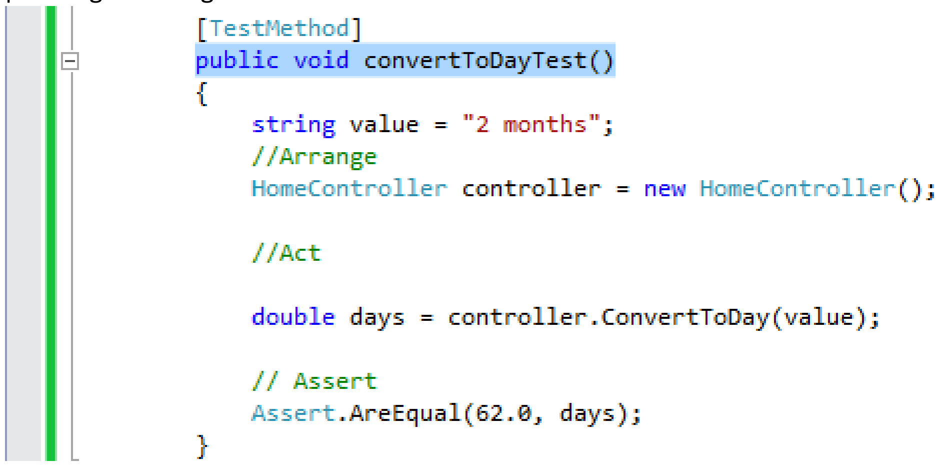
```
[TestMethod]
public void StopListTest()
{
    //Arrange
    HomeController controller = new HomeController();

    //Act
    ViewResult result = controller.Index() as ViewResult;

    //Assert
    Assert.IsNotNull(result);
}
```

3- `public void stopsTest() test value returned after giving a MGLT, consumable and a distance.`
The code snippet look like this:



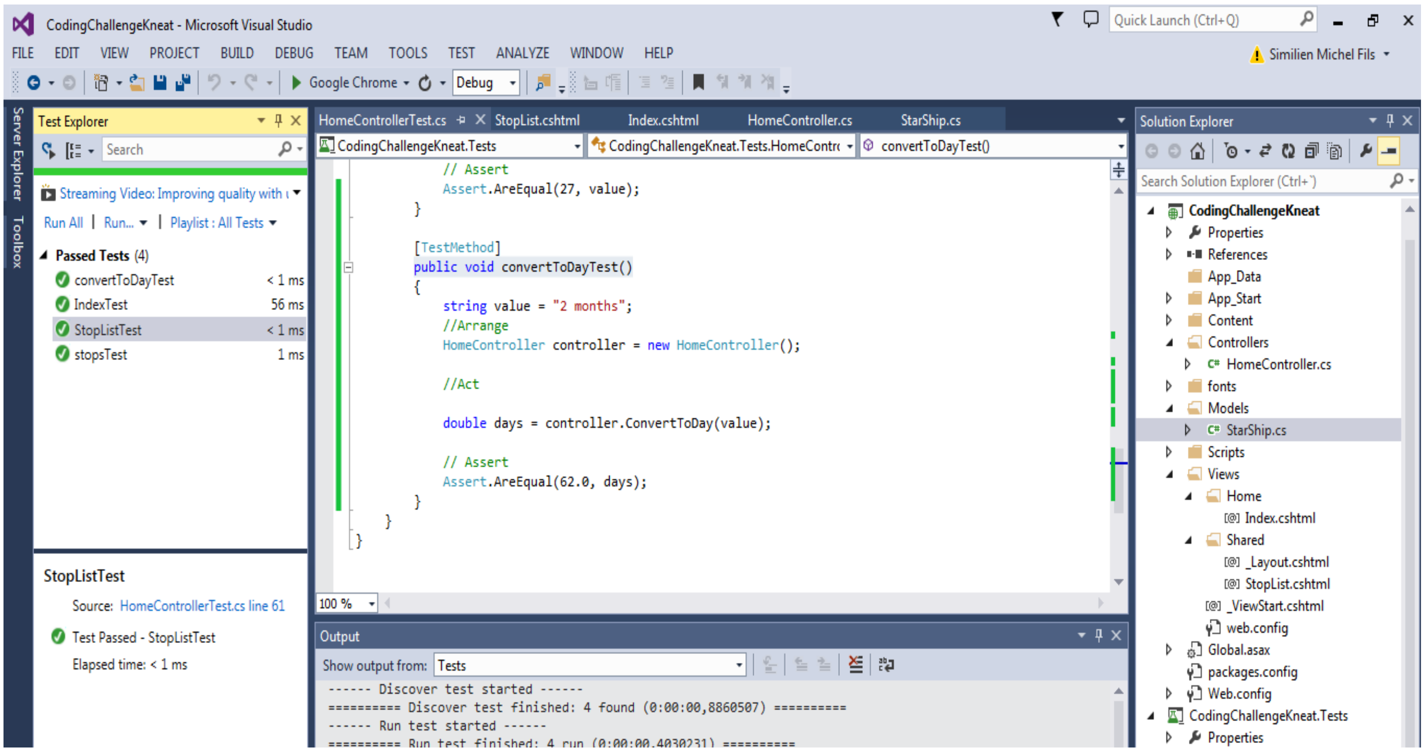
```
[TestMethod]
public void stopsTest()
{
    string mglt = "75";
    string consum = "2 months";
    string distance = "3000000";

    //Arrange
    HomeController controller = new HomeController();

    //Act

    int value = controller.stops(mglt, consum, distance);

    // Assert
    Assert.AreEqual(27, value);
}
```

4- `public void convertToDayTest() test the value in days returned after passing a tring of consumables.`



```
[TestMethod]
public void convertToDayTest()
{
    string value = "2 months";
    //Arrange
    HomeController controller = new HomeController();

    //Act

    double days = controller.ConvertToDay(value);

    // Assert
    Assert.AreEqual(62.0, days);
}
```

This the capture that show all the test run pass.



Passed Tests (4)

convertToDayTest	< 1 ms
IndexTest	56 ms
StopListTest	< 1 ms
stopsTest	1 ms

StopListTest

Source: HomeControllerTest.cs line 61

Test Passed - StopListTest

Elapsed time: < 1 ms

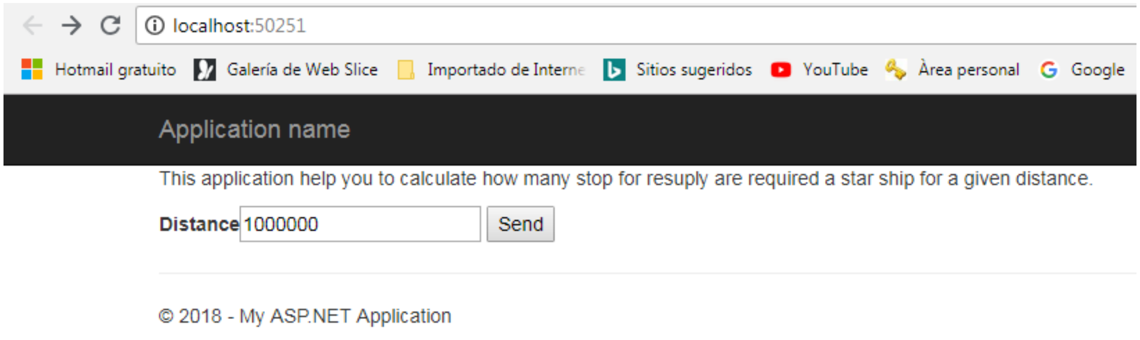
```
            // Assert
            Assert.AreEqual(27, value);
        }

        [TestMethod]
        public void convertToDayTest()
        {
            string value = "2 months";
            //Arrange
            HomeController controller = new HomeController();

            //Act

            double days = controller.ConvertToDay(value);

            // Assert
            Assert.AreEqual(62.0, days);
        }
    }
}
```

```
------ Discover test started ------
========== Discover test finished: 4 found (0:00:00,8860507) ==========
------ Run test started ------
========== Run test finished: 4 run (0:00:00,4030231) ==========
```

4. INSTRUCTION ON THE USAGE OF THE APPLICATION

The application is easy to use. It has a main screen that allows the user to enter a distance and then return in a table the names and the amount of "stop" of each "StarShip".

1- The user enter a given distance and then press send button.



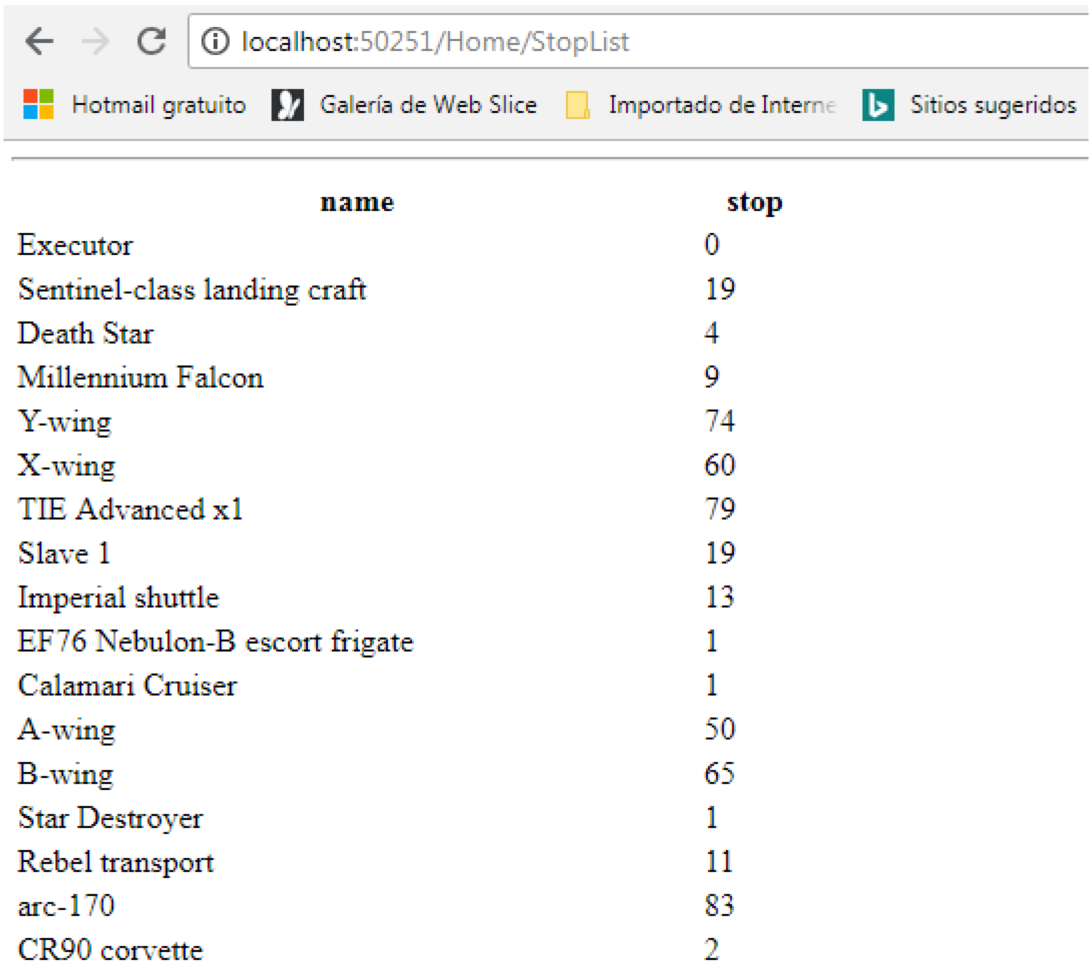
Application name

This application help you to calculate how many stop for resuply are required a star ship for a given distance.

Distance 1000000

© 2018 - My ASP.NET Application

2- The user get all the ships with its number of stop for the given distance.



name	stop
Executor	0
Sentinel-class landing craft	19
Death Star	4
Millennium Falcon	9
Y-wing	74
X-wing	60
TIE Advanced x1	79
Slave 1	19
Imperial shuttle	13
EF76 Nebulon-B escort frigate	1
Calamari Cruiser	1
A-wing	50
B-wing	65
Star Destroyer	1
Rebel transport	11
arc-170	83
CR90 corvette	2