**Vehicle Web API and Client Application:**

This application has two main parts, 1) Vehicle Web API; 2) Presentation layer using AngularJS and Razor html page.

Files to download:

1. Download and extract the “VehicleWebApp.Zip” file.
2. This document contains for steps to run the application.

Steps to run the Application:

1. Open “VehicleWebApp.Sln” file in Visual Studio tool.
2. Change the database connection string in “VehicleWebApp project -> Web.config” file (<add name="VehiclesDbContext" connectionString="data source=(LocalDb)\v11.0;initial catalog=VehiclesDB;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework" providerName="System.Data.SqlClient" />)
3. This application has developed using ASP.NET MVC, Web API, Entityframework and AngularJS tools.

Vehicles Web Application Solution Structure:

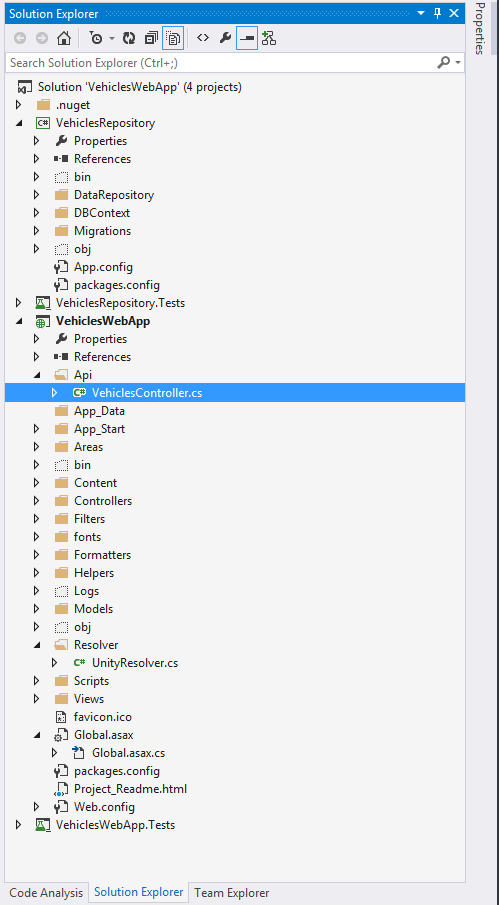
The VehiclesWebApp solution contains four projects. They are,

1. VehiclesWebApp - web project contains Web API and client
2. VehiclesRepository – Contains Code first DB entities and Repository class
3. VehiclesWebApp.Tests – It contains Unit test related to web project
4. VehiclesRepository.Tests – It contains Repository class unit test

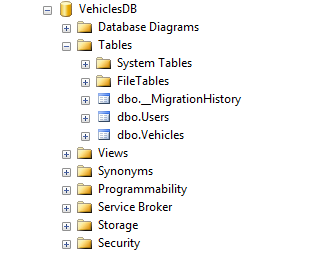
Important Points about Application:

1. The Web application contains VehiclesController web API used to do CRUD API operations
2. The Unity used to inject Repository and Logging instance.
3. Web API Help Nuget package used to explore API help page
4. MVC filters like Validationfilter, Exception filter, logging filters are used.
5. Helper class like Logging, covert ViewModel to Entity and vice versa conversion
6. The front-end Vehicle test application developed using AngularJS and ASP.NET MVC razor
7. VehiclesRepository library have Code-first Entity model created with default seed operations
8. Repository pattern used here. Like VehiclesDataRepository and UserDataRepository
9. Task based asyn calls are used. Default cache in Entity framework.
10. Unit test are done using Microsoft unit test framework and Moq framework
11. User entity added for future Authentication module. It is not complete.
12. The Vehicle APIs are tested using “Advanced Rest Client (Chrome Extension)

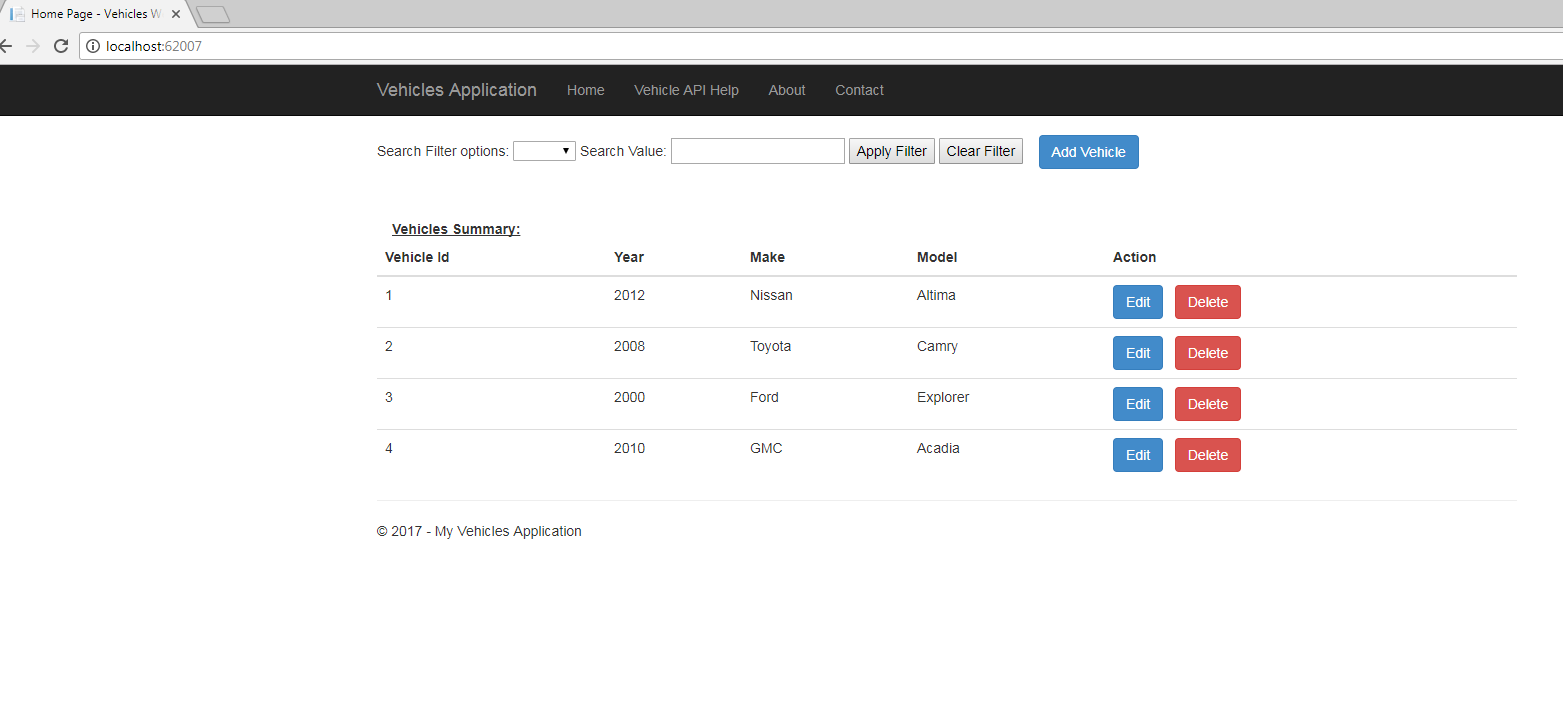
The Solution structure:

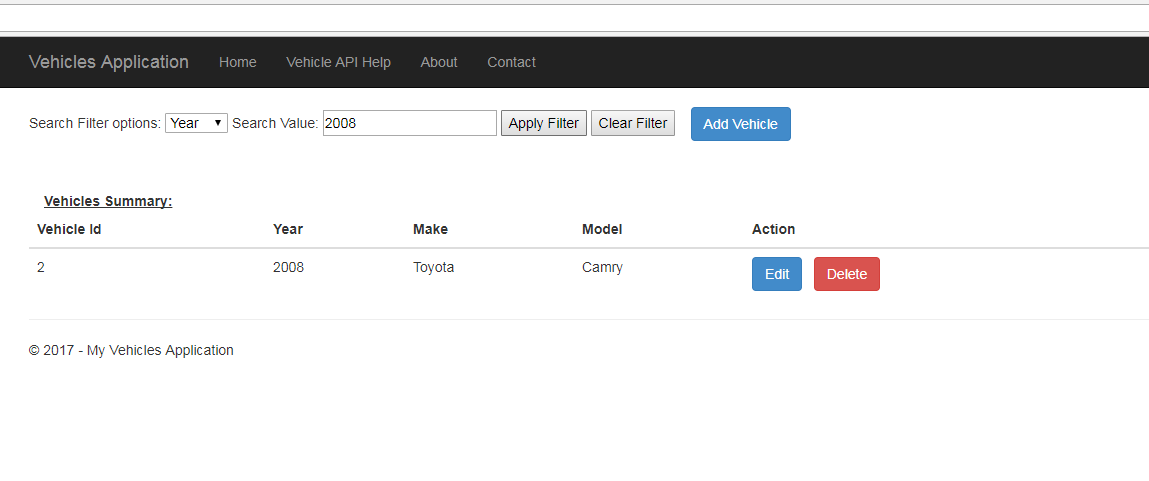


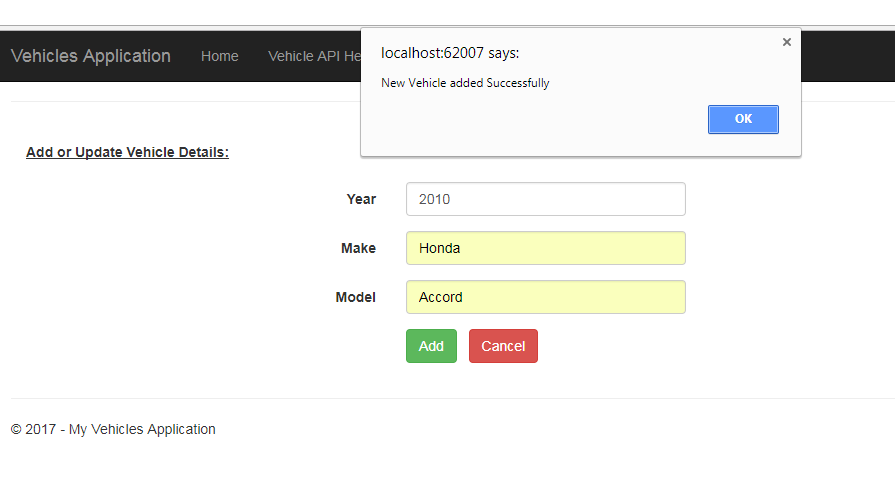
Database Created by Application:

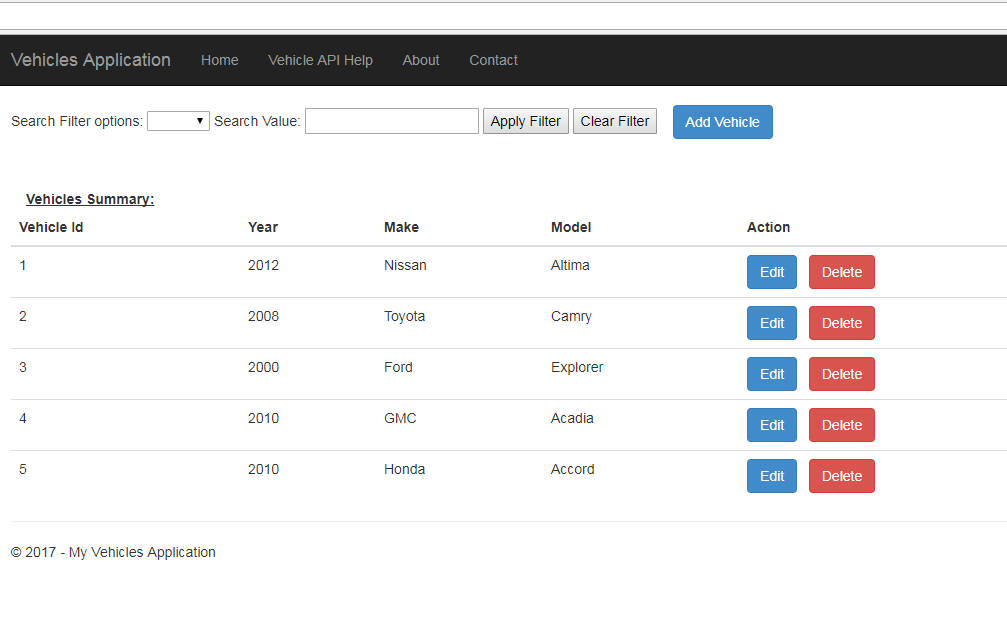


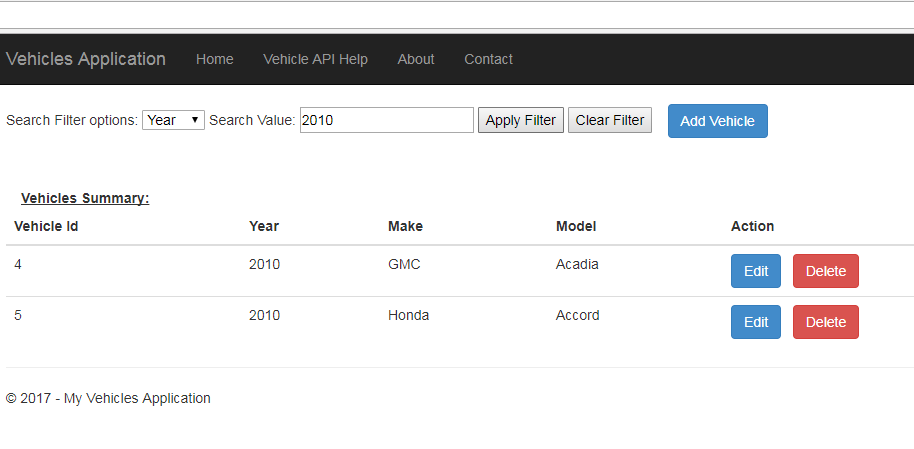
The running application sample pages:

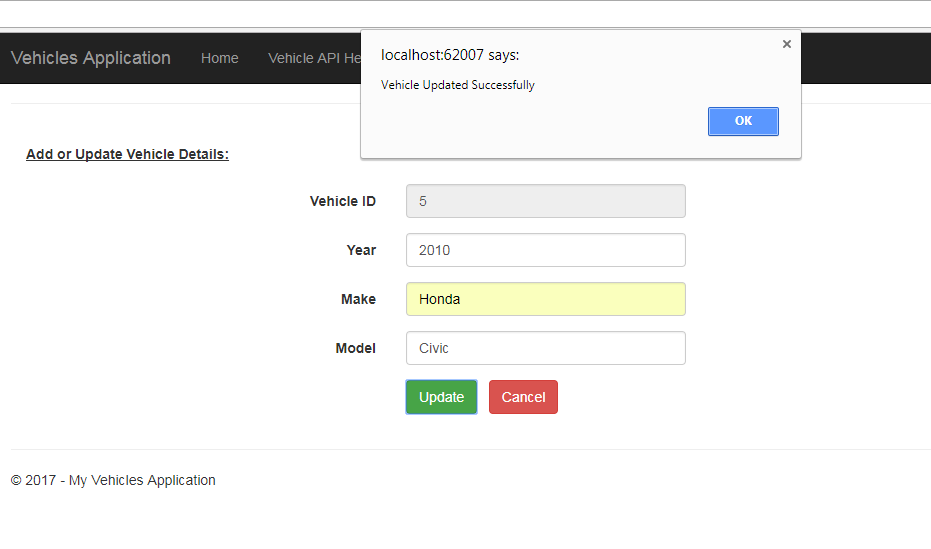


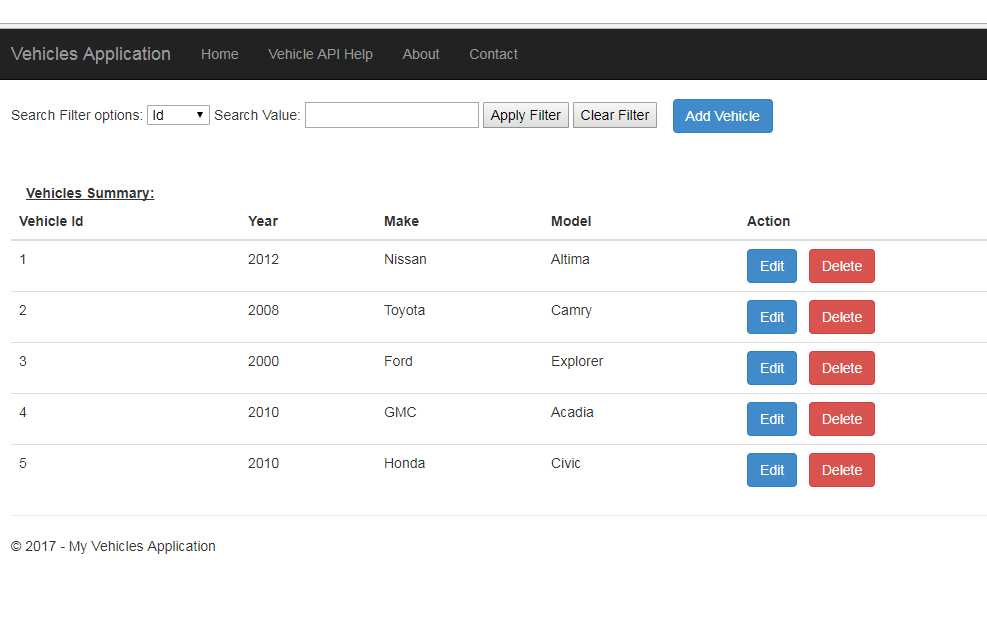


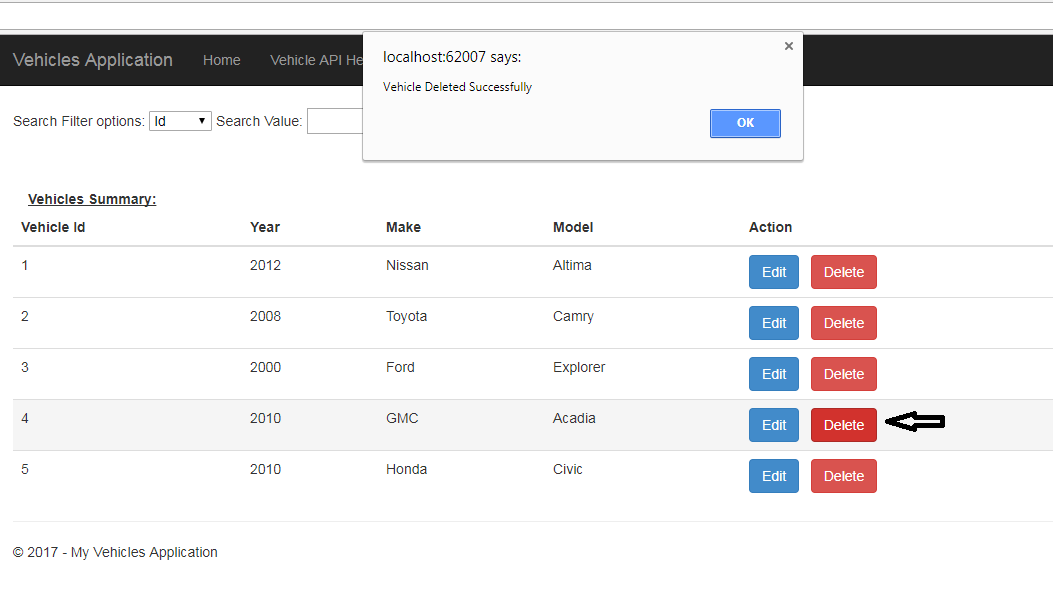


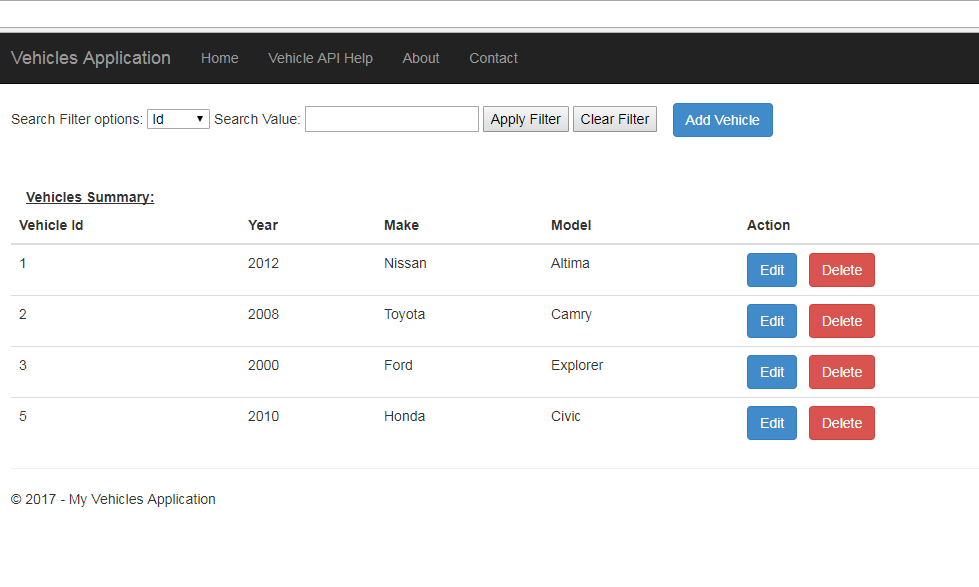




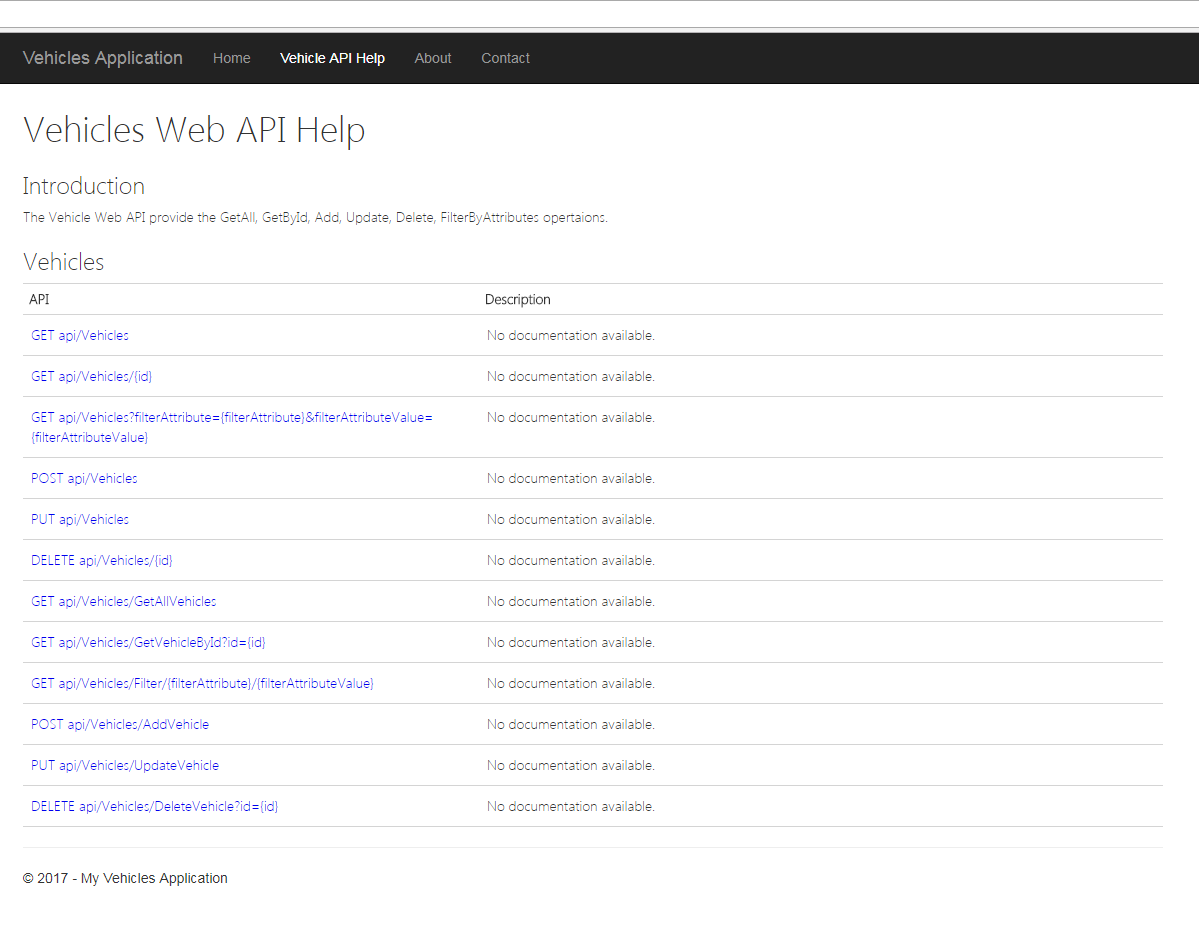


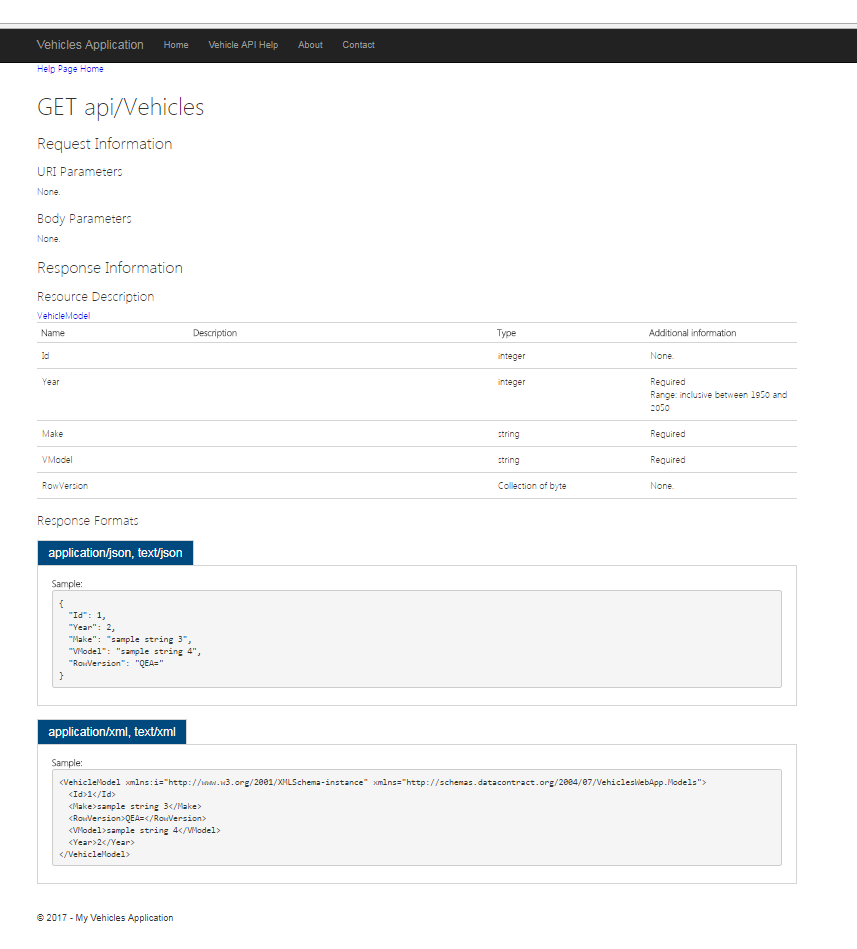






API Help Page: (The API methods repeated due to two Http Routing)





Testing Web API using Advance Rest Client (Chrome Extension):

