After adding the Controller and View now let us move back towards the Web API and make some changes that we have already created with the name “CarDetailsController”.

Let's get to the first method in CarDetailsController.

1. **GET IEnumerable**

```
[HttpGet]
  public IEnumerable<CarsStock> GetAllcarDetails()
  {
     CarsStock ST = new CarsStock();
      CarsStock ST1 = new CarsStock();
     List<CarsStock> li = new List<CarsStock>();

     ST.CarName = "Maruti Waganor";
     ST.CarPrice = "4 Lakh";
     ST.CarModel = "VXI";
     ST.CarColor = "Brown";

     ST1.CarName = "Maruti Swift";
     ST1.CarPrice = "5 Lakh";
     ST1.CarModel = "VXI";
     ST1.CarColor = "RED";

     li.Add(ST);
     li.Add(ST1);
     return li;
  }
```

This method is used to get a list of data.

In this method I have used the Model CarsStock and created a list of CarsStock “List<CarsStock>“.

And returning it.

GET by id

```
public IEnumerable<CarsStock> Get(int id)
{
   CarsStock ST = new CarsStock();
   CarsStock ST1 = new CarsStock();
```

```
25.    List<CarsStock> li = new List<CarsStock>();
26.    if (id == 1)
27.    {
28.        ST.CarName = "Maruti Waganor";
29.        ST.CarPrice = "4 Lakh";
30.        ST.CarModel = "VXI";
31.        ST.CarColor = "Brown";
32.        li.Add(ST);
33.    }
34.    else
35.    {
36.        ST1.CarName = "Maruti Swift";
37.        ST1.CarPrice = "5 Lakh";
38.        ST1.CarModel = "VXI";
39.        ST1.CarColor = "RED";
40.        li.Add(ST1);
41.    }
42.    return li;
43. }
```

In this GET method you can retrieve records for the database by passing an id.

POST

```
44. [HttpPost]
45.  public void PostCar([FromBody] CarsStock cs)
46.  {
47.
48.  }
```

In this POST method you can post data (CREATE) to the database. In this I am using the Carstock model to post the data.

PUT

```
2.    [HttpPut]
3.   public void Putcar(int id, [FromBody]CarsStock cs)
4.   {
5.
```

```
6.      }
```

In this PUT method you can UPDATE the data (UPDATE) to the database. I am using the Carstock model to update the data.

DELETE

```
7.  [HttpDelete]
8.      public void Deletecar(int id)
9.       {
10.
11.     }
```

In this DELETE method you can delete data (DELETE) from the database. I am using an id to delete the data.

Here is a snapshot of all the methods and models after adding the attributes to it.

```
using System.Web.Http;
using WebAPI_Basic.Models;
namespace WebAPI_Basic.Controllers
{
    public class CarDetailsController : ApiController
    {

        [HttpGet] // GET api/cardetails
        public IEnumerable<CarsStock> GetAllcarDetails()...

        // GET api/cardetails/5
        public string Get(int id)...


        [HttpPost] // POST api/cardetails
        public void PostCar([FromBody] CarsStock cs)...


        [HttpPut]  // put api/cardetails/5
        public void Putcar(int id, [FromBody]CarsStock cs)...


        [HttpDelete] // put api/cardetails/5
        public void Deletecar(int id)...

    }
}
```

Now let's move to the view and do CRUD operations from there.

For getting a list of data I have created a function in jQuery.

1. Calling GET IEnumerable List from Ajax and getting data from the Web API.

```
1.  <script lang="ja" type="text/javascript">
2.
3.      function AllcarDetails() {
4.          $.ajax({
5.              type: "GET",
6.              url: "http://localhost:32359/api/Cardetails", //URI
7.
8.              dataType: "json",
9.              success: function (data) {
10.                 debugger;
11.                 var datadatavalue = data;
12.                 var myJsonObject = datavalue;
```

```
            contentType: "application/json";
            $.each(myJsonObject, function (i, mobj) {
                $("#Cartbl").append('<tr><td width="50px">' + mobj.CarName +
                 '</td><td width="50px">' + mobj.CarModel +
                '</td><td width="50px">' + mobj.CarPrice +
                '</td>' + '</td><td width="50px">'
                + mobj.CarColor + '</td></tr>');

            });

        },
        error: function (xhr) {
            alert(xhr.responseText);
        }
    });

}
```

2. Calling PostCar Method using Ajax and posting data to the Web API.

```
function PostData()
 {

 var cardetails =
 {
CarName: "Ertiga",
CarModel: "LXI",
CarPrice: "5000000",
CarColor: "blue"
};

    $.ajax({
        type: "POST",
        data: JSON.stringify(cardetails),
        url: "http://localhost:32359/api/Cardetails",
        dataType: "json",
        contentType: "application/json",
    });

  }
```

3. Calling the PUTcar method using Ajax and updating the data of the Web API.

```
function PutData() {

```

```
var cardetails =
 {

 CarName: "Ertiga",
 CarModel: "LXI",
 CarPrice: "5000000",
 CarColor: "blue"

 };

      var t = JSON.stringify(cardetails);
      var id = "0";
      $.ajax({
         url: 'http://localhost:32359/api/Cardetails/' + id,
         type: "put",
         contentType: "application/json; charset=utf-8",
         data: t,
         dataType: "json",

      });
   }
```

4. Calling the Delete car method using Ajax and to delete data of the Web API.

```
function deleteData1()
   {
         var id = 0;
         $.ajax({
            url: 'http://localhost:32359/api/CarDetails/' + id,
            type: 'DELETE',
            success: function (data) {

            },
            error: function (data) {
               alert('Problem in deleting car:' + data.responseText);
            }
         });
}
```

5. Calling GET by ID from Ajax and getting data from the Web API by id.

```
function GetCarById() {
      var id = 1;
      $.ajax({
         url: 'http://localhost:32359/api/CarDetails/' + id,
```

```
5.          type: 'GET',
6.          dataType: "json",
7.          success: function (data) {
8.
9.              var datavalue = data;
10.             var myJsonObject = datavalue;
11.
12.             var CarModel = myJsonObject[0].CarModel;
13.             var CarName = myJsonObject[0].CarName;
14.             var CarColor = myJsonObject[0].CarColor;
15.             var CarPrice = myJsonObject[0].CarPrice;
16.
17.             $('<tr><td>' + CarModel + '</td><td>' + CarName +
18.             '</td><td>' + CarColor + '</td>' + '</td><td>' + CarPrice + '</td></tr>')
    .appendTo('#Cartbl');
19.         },
20.         error: function (xhr) {
21.             alert(xhr.responseText);
22.         }
23.     });
24.  }
```

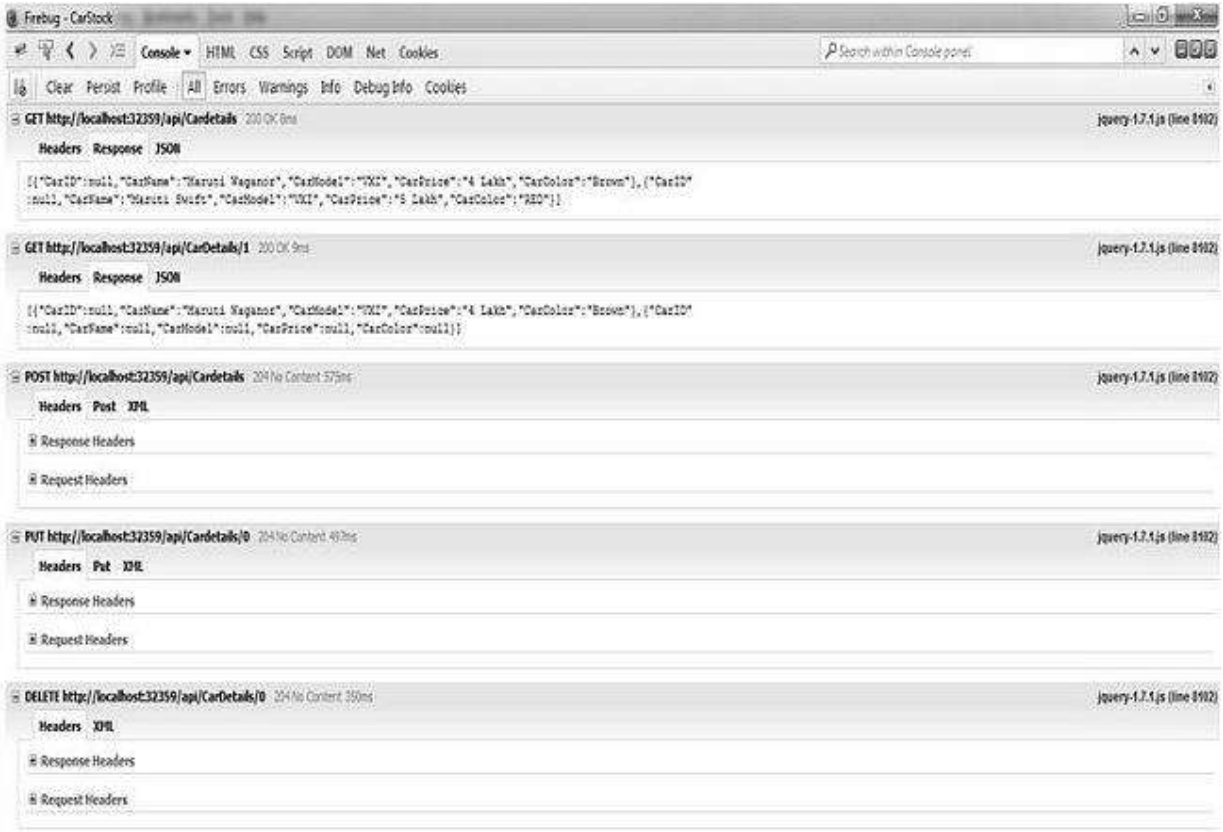
After completing all the functions of Ajax I am now displaying the view “CarStock”.

```
@{ViewBag.Title = "CarStock";}
<h2>CarStock</h2>
<script src="~/Scripts/jquery-1.7.1.min.js"></script>
<script lang="ja" type="text/javascript">
    function GetCarById()...
    function AllcarDetails()...
    function PostData()...
    function PutData()...
    function deleteData1()...
</script>

@using (Html.BeginForm())
{
    <div style="text-align:center;">
     <table id="Cartbl" ...>...</table>
     <table border='1' width="70%" style="color: chocolate" id="Cartbl">
        <tr>
            <td>
                <input id="btnget" type="button" onclick="AllcarDetails();" value="Get_Data" />
                <input id="btngetbyid" type="button" onclick="GetCarById();" value="Get_BYID" />
                <input id="Btnpost" type="button" onclick="PostData();" value="Post_Data" />
                <input id="Btnput" type="button" onclick="PutData();" value="Put_Data" />
                <input id="BtnDelete" type="button" onclick="deleteData1();" value="Delete" />
            </td></tr>
    </table>
    </div>
}
```

In the carstock view I have just added 5 buttons and on the click event of every button a different function of the Web API is called.

The following snippet is debugged in Firebug.



Final Output

Here in this view I have consumed a Web API for Demo.