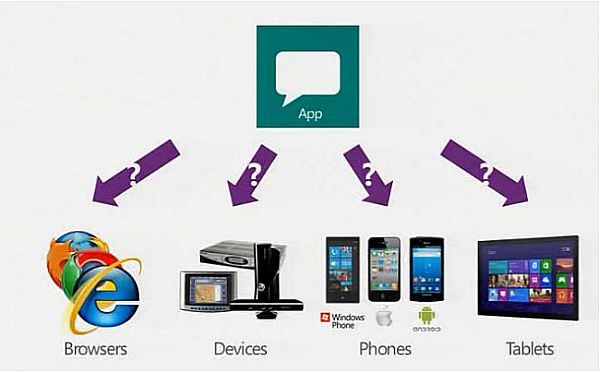
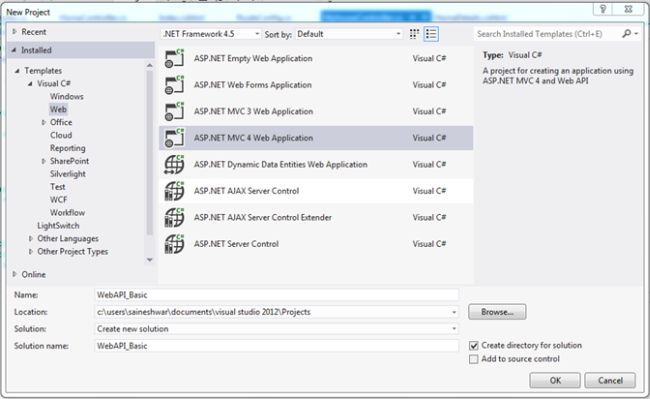
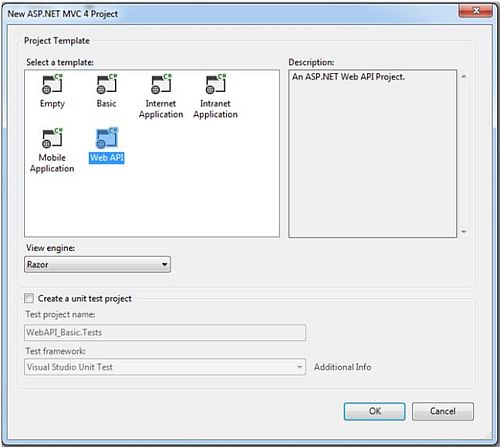
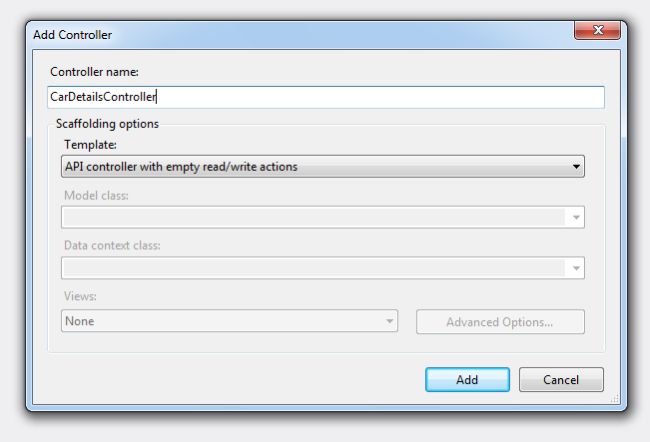
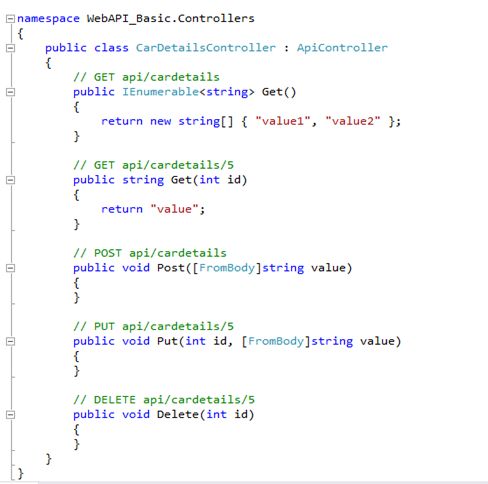
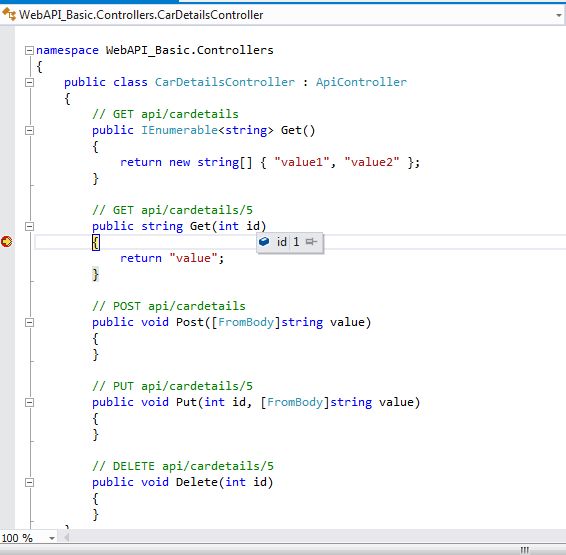
How to Create Web API in ASP.Net MVC

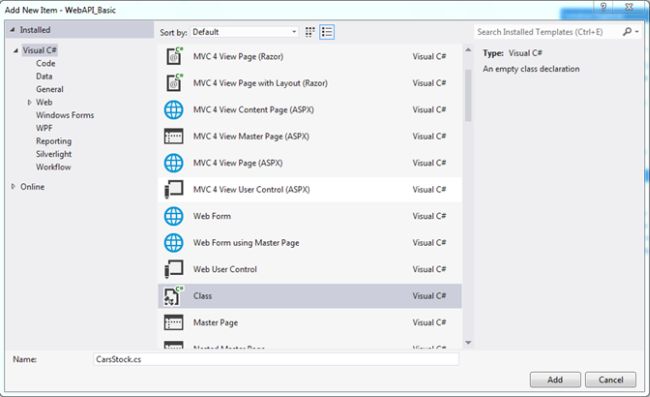
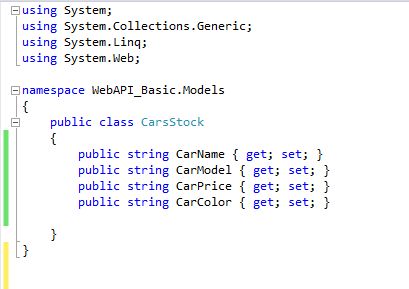
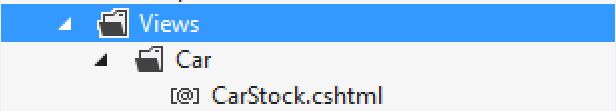
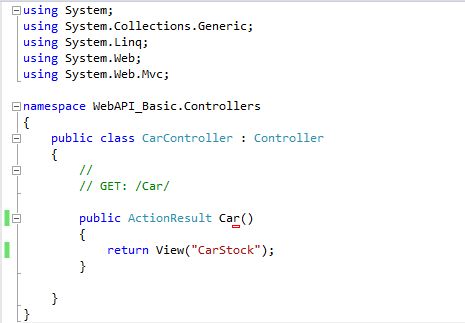
This article explains what the Web API is and its basics.  
  
  
  
**Web API**  
  
The ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices.  
  
The ASP.NET Web API is an ideal platform for building Restful applications on the .NET Framework. Referred from “Microsoft.com”.  
  
**Why to use the Web API**  
  
Currently most mobile devices, browsers and tablets are the medium for accessing most of the internet and in this also people are using mobile apps the most and to provide data to apps we are now going to use the Microsoft new technology called Web API.

**When to use it**  
  
If you want to expose the data/information of your application to your clients and other people then that other people can use your data and interact with the data/information you expose to them.  
  
For example, a mobile application requires a service.  
  
HTML 5 requires a service.  
  
Desktop PC and tablets require services.  
  
Currently most device apps require Web API services.   
  
The ASP.Net Framework leverages both web standards such as HTTP, JSON and XML and it provides a simple way to build and expose REST based data services.  
  
Some core concepts of ASP.Net MVC are similar to the ASP.Net Web API such as routing and controllers.  
  
**Requirements**  
  
We are using Visual Studio 2012 for a demo application.  
  
**Building a Web API**  
  
Let's start with creating a Web API project.  
  
Start Visual Studio and select New project from the Start page or from the File menu select "File" -> "New" -> "Project...".  
  
In the template pane select Installed Templates and expand the Visual C# menu. Inside that Visual C# select Web. In the list of projects select ASP.Net MVC 4 Web Application.  
  
And name the project WebAPI\_Basic.  
  
For reference see the following snapshot.  
  
  
  
After adding, a new dialog will pop-up.  
  
  
  
Inside the project template select Web API and in the view engine select Razor.  
  
A new project is created now.  
  
**Let's begin with adding Web API Controller**  
  
Now let's begin with adding a Web API Controller. It is nearly similar to adding a Controller in ASP.NET MVC.  
  
Right-click on the Controller folder and add a new Web API Controller with the name CarDetailsController and in the template select API Controller with an empty read / write action.  
  
  
  
After adding the Controller you will see the code as in the following snapshot.  
  
  
  
You can keep this Web API controller anywhere in the application.  
  
If you want to follow the convention then create the new folder in the root your of application with the name API.  
  
Inside that you can add a Web API controller.  
  
You have successfully added a Web API controller to your application.  
  
Now you can run the application and test it.  
  
For testing I am passing http://localhost:32359/api/cardetails/1 for calling the method get.  
  
Wow, it's working!  
  
  
  
It's easy to configure it as a Web API.  
  
The ASP.Net MVC  and ASP.Net Web API makes heavy use of convention for configuration to lighten the work load for creating the services.  
  
For example, add a decorating method with attributes to make it easy to do CRUD operations.  
  
Else it will make it difficult to understand and code.

1. [HttpPut]
2. **public** **void** Put(**int** id, [FromBody]**string** value)
3. {
5. }
6. [HttpPost]
7. **public** **void** Post([FromBody]**string** value)
8. {
10. }
11. [HttpDelete]
12. **public** **void** Delete(**int** id)
13. {

The HTTP actions and their corresponding CRUD operations are:

* GET (Read)  
    
  Retrieves the representation of the resource.
* PUT(Update)  
    
  Update an existing resource.
* POST (Create)  
    
  Create new resource.
* DELETE (Delete)  
    
  Delete an existing resource.

Now let's begin with how to create a CRUD operation with the WEB API.  
  
Let's start by adding a Model.  
  
To add the model right-click on the model folder and add a class with the name CarsStock.  
  
  
  
After adding the Model CarsStock.cs now let's start with adding properties to the class.  
  
  
  
After adding the model properties now I will consume the HTTP service developed using the ASP.NET Web API in a simple cshtml page with jQuery and Ajax.  
  
For that in the View folder I will add a folder named Car and inside that folder will add a CarStock named view. To add it just right-click on the View folder and select View.  
  
The following snapshot shows how I had added the view.  
  
  
  
After adding the view you will get a blank view because we are not using any tightly coupled model here.  
  
Then add a Controller with the name CarController. Call this view Carstock for the demo of consuming the Web API.  
  
In this I called the view CarStock.  
  
  
  
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**
   1. [HttpGet]
   2. **public** IEnumerable<CarsStock> GetAllcarDetails()
   3. {
   4. CarsStock ST = **new** CarsStock();
   5. CarsStock ST1 = **new** CarsStock();
   6. List<CarsStock> li = **new** List<CarsStock>();
   8. ST.CarName = "Maruti Waganor";
   9. ST.CarPrice = "4 Lakh";
   10. ST.CarModel = "VXI";
   11. ST.CarColor = "Brown";
   13. ST1.CarName = "Maruti Swift";
   14. ST1.CarPrice = "5 Lakh";
   15. ST1.CarModel = "VXI";
   16. ST1.CarColor = "RED";
   18. li.Add(ST);
   19. li.Add(ST1);
   20. **return** li;
   21. }

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.

1. **GET by id**
   1. **public** IEnumerable<CarsStock> Get(**int** id)
   2. {
   3. CarsStock ST = **new** CarsStock();
   4. CarsStock ST1 = **new** CarsStock();
   5. List<CarsStock> li = **new** List<CarsStock>();
   6. **if** (id == 1)
   7. {
   8. ST.CarName = "Maruti Waganor";
   9. ST.CarPrice = "4 Lakh";
   10. ST.CarModel = "VXI";
   11. ST.CarColor = "Brown";
   12. li.Add(ST);
   13. }
   14. **else**
   15. {
   16. ST1.CarName = "Maruti Swift";
   17. ST1.CarPrice = "5 Lakh";
   18. ST1.CarModel = "VXI";
   19. ST1.CarColor = "RED";
   20. li.Add(ST1);
   21. }
   22. **return** li;
   23. }

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

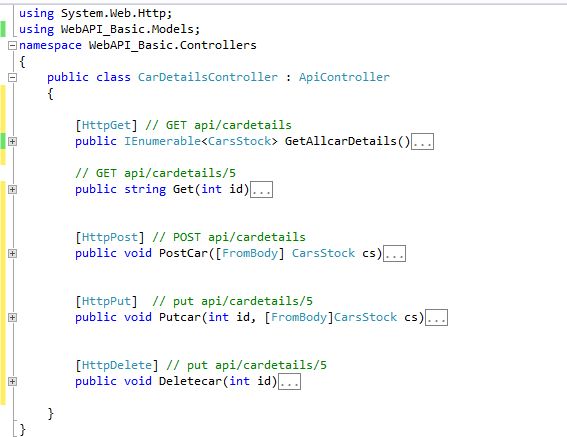
1. **POST**
   1. [HttpPost]
   2. **public** **void** PostCar([FromBody] CarsStock cs)
   3. {
   5. }

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

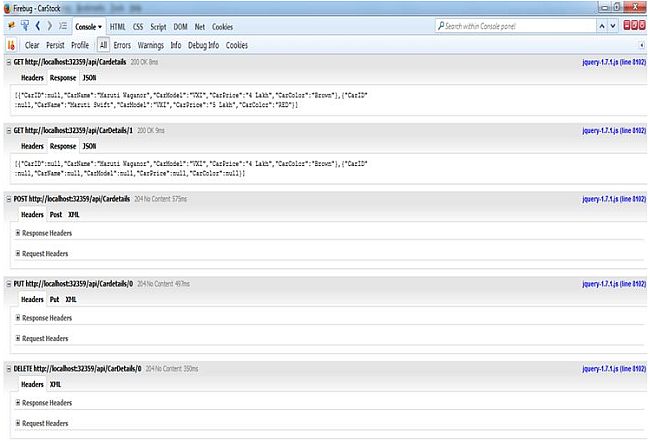
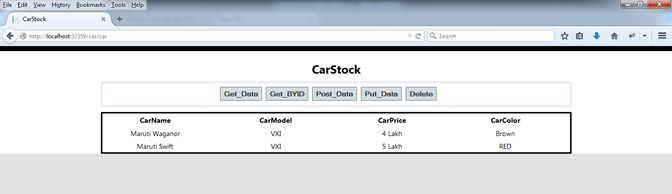
1. **PUT**
   1. [HttpPut]
   2. **public** **void** Putcar(**int** id, [FromBody]CarsStock cs)
   3. {
   5. }

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

1. **DELETE**
   1. [HttpDelete]
   2. **public** **void** Deletecar(**int** id)
   3. {
   5. }

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.  
  
  
  
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">
   3. function AllcarDetails() {
   4. $.ajax({
   5. type: "GET",
   6. url: "http://localhost:32359/api/Cardetails", //URI
   8. dataType: "json",
   9. success: function (data) {
   10. debugger;
   11. var datadatavalue = data;
   12. var myJsonObject = datavalue;
   13. contentType: "application/json";
   14. $.each(myJsonObject, function (i, mobj) {
   15. $("#Cartbl").append('<tr><td width="50px">' + mobj.CarName +
   16. '</td><td width="50px">' + mobj.CarModel +
   17. '</td><td width="50px">' + mobj.CarPrice +
   18. '</td>' + '</td><td width="50px">'
   19. + mobj.CarColor + '</td></tr>');
   21. });
   23. },
   24. error: function (xhr) {
   25. alert(xhr.responseText);
   26. }
   27. });
   29. }
2. Calling PostCar Method using Ajax and posting data to the Web API.
   1. function PostData()
   2. {
   4. var cardetails =
   5. {
   6. CarName: "Ertiga",
   7. CarModel: "LXI",
   8. CarPrice: "5000000",
   9. CarColor: "blue"
   10. };
   12. $.ajax({
   13. type: "POST",
   14. data: JSON.stringify(cardetails),
   15. url: "http://localhost:32359/api/Cardetails",
   16. dataType: "json",
   17. contentType: "application/json",
   18. });
   20. }
3. Calling the PUTcar method using Ajax and updating the data of the Web API.
   1. function PutData() {
   3. var cardetails =
   4. {
   6. CarName: "Ertiga",
   7. CarModel: "LXI",
   8. CarPrice: "5000000",
   9. CarColor: "blue"
   11. };
   13. var t = JSON.stringify(cardetails);
   14. var id = "0";
   15. $.ajax({
   16. url: 'http://localhost:32359/api/Cardetails/' + id,
   17. type: "put",
   18. contentType: "application/json; charset=utf-8",
   19. data: t,
   20. dataType: "json",
   22. });
   23. }
4. Calling the Delete car method using Ajax and to delete data of the Web API.
   1. function deleteData1()
   2. {
   3. var id = 0;
   4. $.ajax({
   5. url: 'http://localhost:32359/api/CarDetails/' + id,
   6. type: 'DELETE',
   7. success: function (data) {
   9. },
   10. error: function (data) {
   11. alert('Problem in deleting car:' + data.responseText);
   12. }
   13. });
   14. }
5. Calling GET by ID from Ajax and getting data from the Web API by id.
   1. function GetCarById() {
   2. var id = 1;
   3. $.ajax({
   4. url: 'http://localhost:32359/api/CarDetails/' + id,
   5. type: 'GET',
   6. dataType: "json",
   7. success: function (data) {
   9. var datavalue = data;
   10. var myJsonObject = datavalue;
   12. var CarModel = myJsonObject[0].CarModel;
   13. var CarName = myJsonObject[0].CarName;
   14. var CarColor = myJsonObject[0].CarColor;
   15. var CarPrice = myJsonObject[0].CarPrice;
   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”.  
  
  
  
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.  
  
  
  
I know that now all my web developer friends must have a basic understanding of what the Web API is.