

ASP.NET Web API – Part 1

Agenda

- How to create a Web API Controller
- How to set route for Web API
- Making HTTP (GET, POST) hits to Web API Controller
- File Posting (AJAX) to Web API

Tools

- Visual Studio 2013

Pre-requisite

- Understanding of how to create basic ASP.NET MVC application in Visual Studio

Version	Last updated	Comments	Modified By
V1.0	16-05-2016		Bilal Shahzad

Brief Introduction

Microsoft has provided “ASP.NET WEB API” technology to write “REST based APIs”. With .Net framework 4.0, you can create an ASP.NET MVC Project + API controllers in it. But with 4.5, you may create separate WEB API project.

In this tutorial, you will learn how to write a basic WEB API Controller.

When content is returned to client, it is formatted. By default two formatters (XML & JSON) are available and XML is default one. It means when you will return result from your WEB API Controller, it will be formatted in XML. So if you want JSON format, just remove XML formatter from the list (you will see in demo how to do this).

Understanding of Model Binder is very important to pass data to WEB API Controller. If action is taking primitive parameters, model binder searches the values from request header (query string) but if action is taking “complex type”, model binder tries to construct the object by searching in request “Body”.

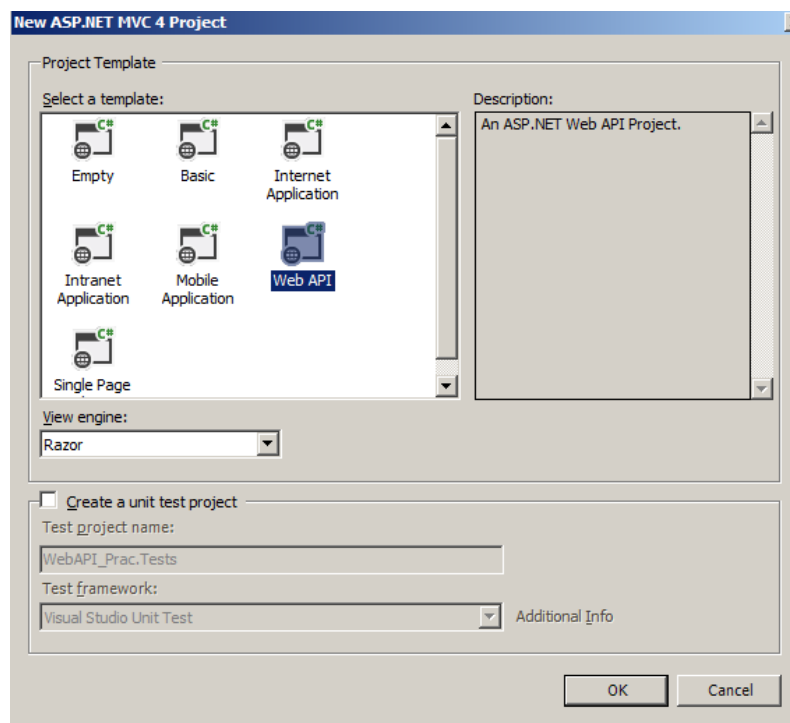
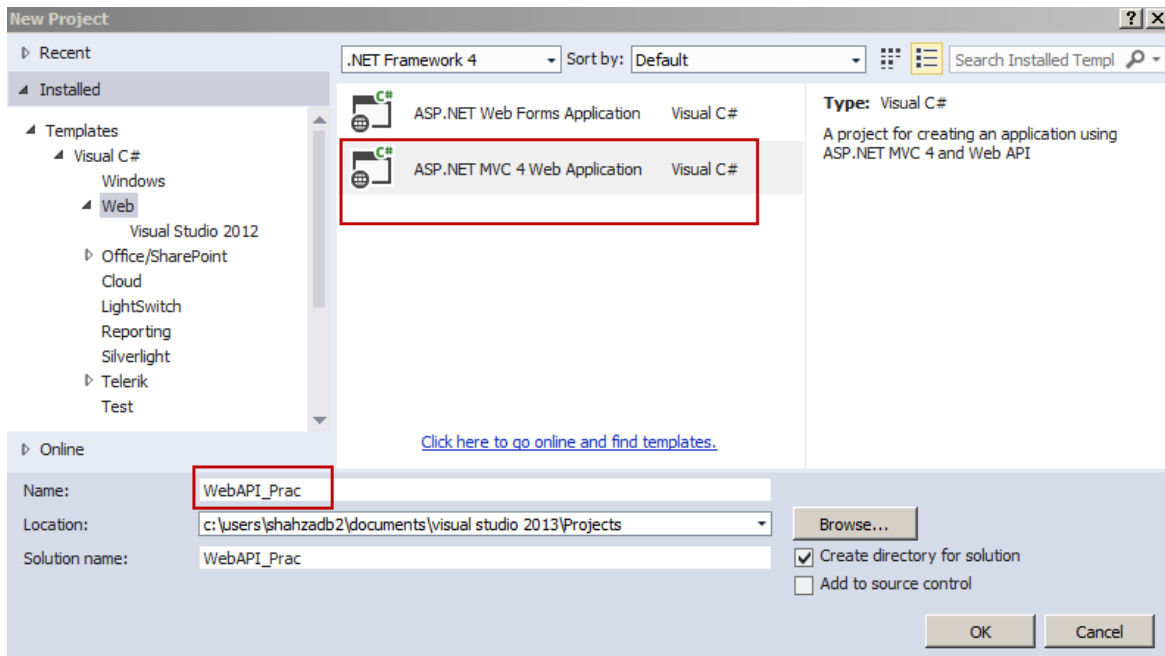
You can guide model binder where to look for values by “[FromUri]” or “[FromBody]” attributes.

Here you don’t need to use “JsonResult” as return value of “action”. Whatever you will return, will be converted to XML or JSON based on your formatters settings.

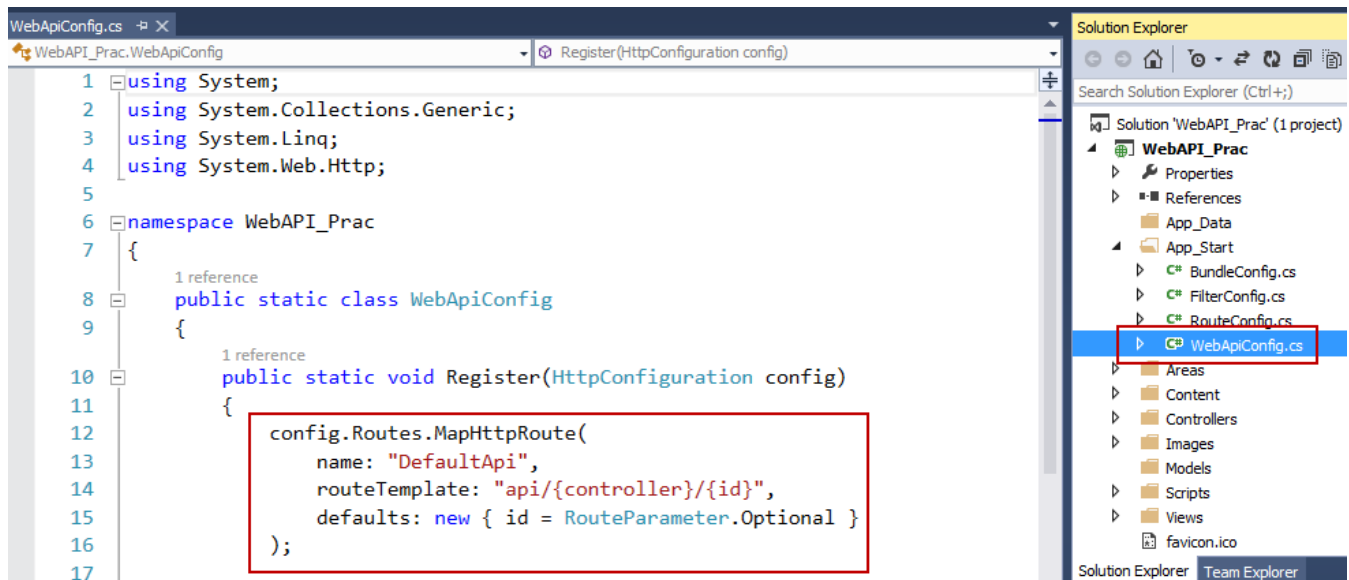
Also check [https://msdn.microsoft.com/en-us/library/jj823172\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/jj823172(v=vs.110).aspx)

Step by Step Walkthrough

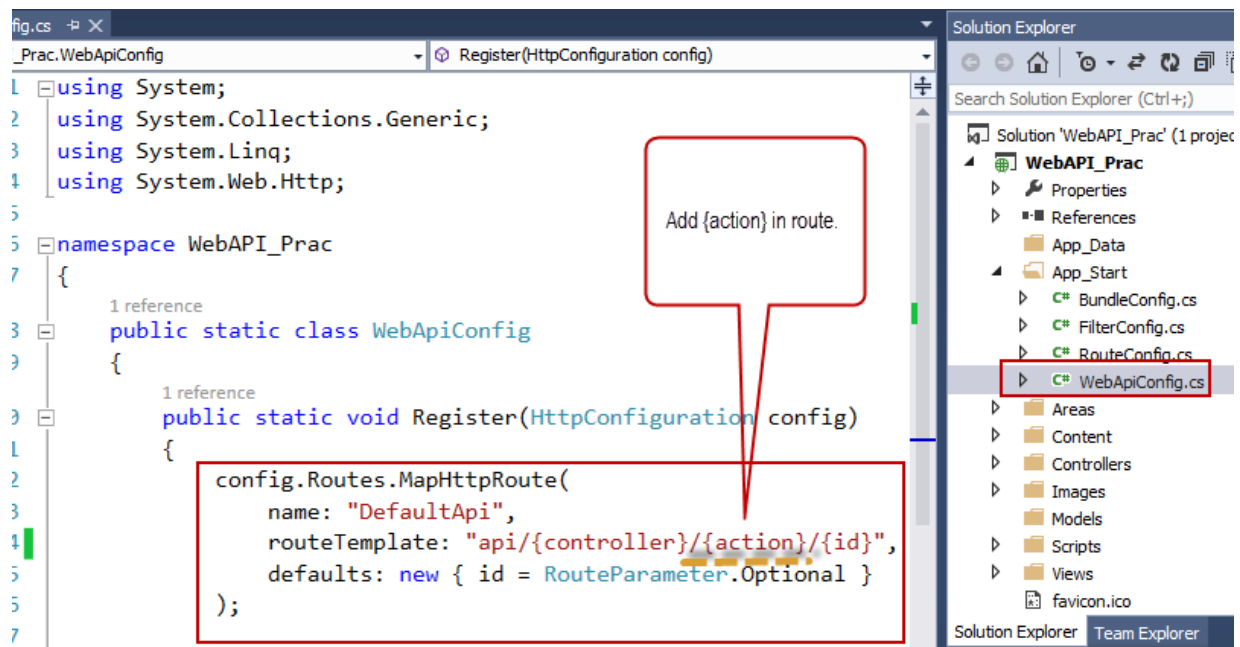
- 1- Create a new ASP.NET MVC application.



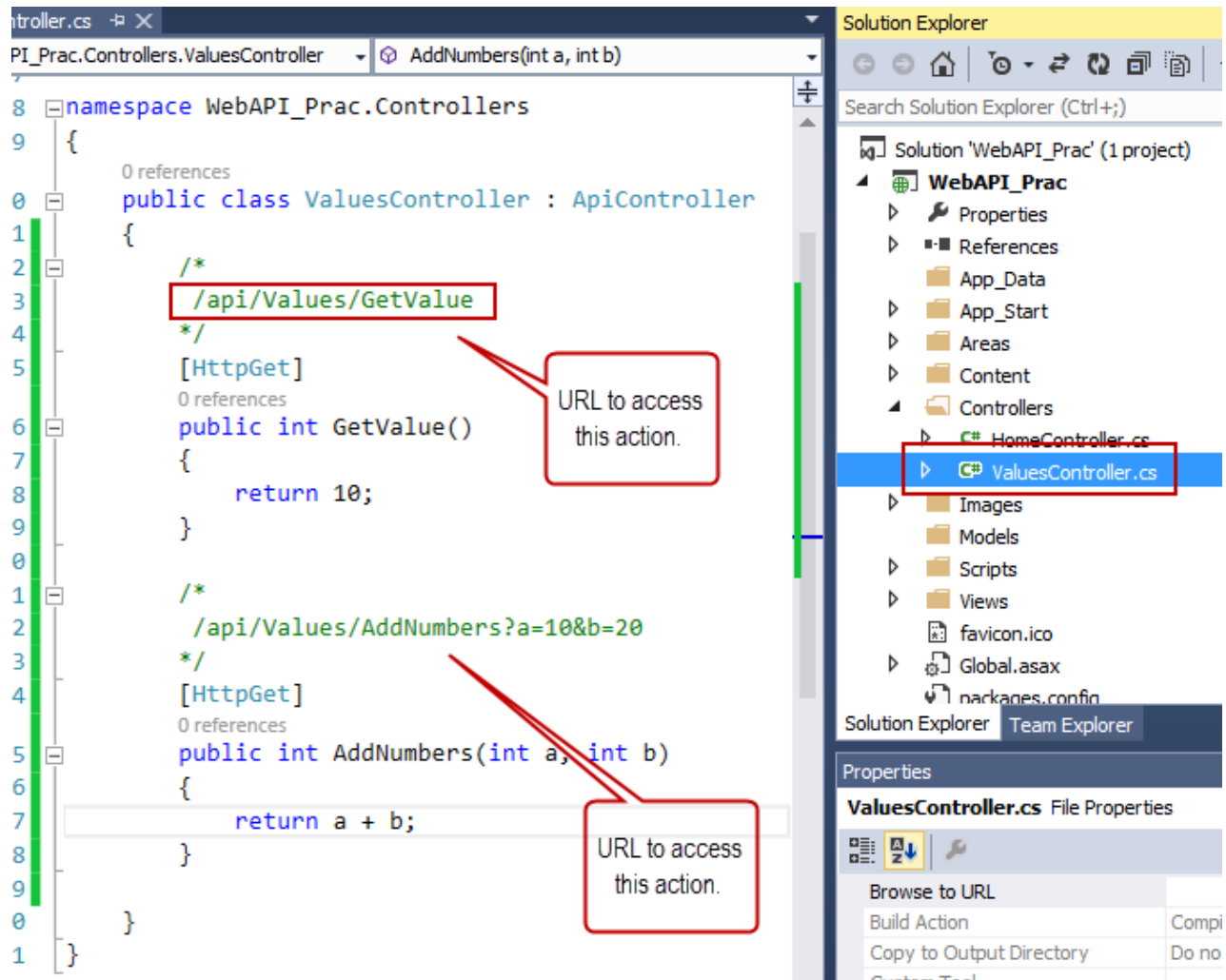
2- In App_Start, you will see “WebApiConfig.cs” file. If you open it, you will see that default route is configured.



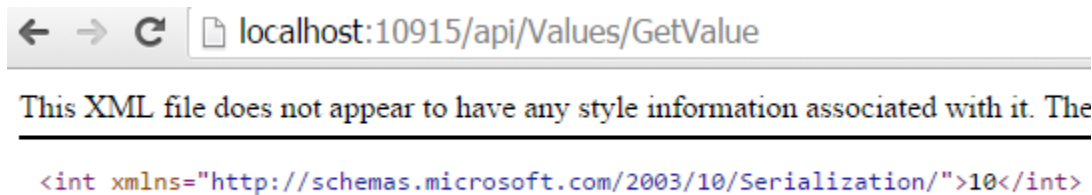
3- Let's change the default route and add {action} in it.



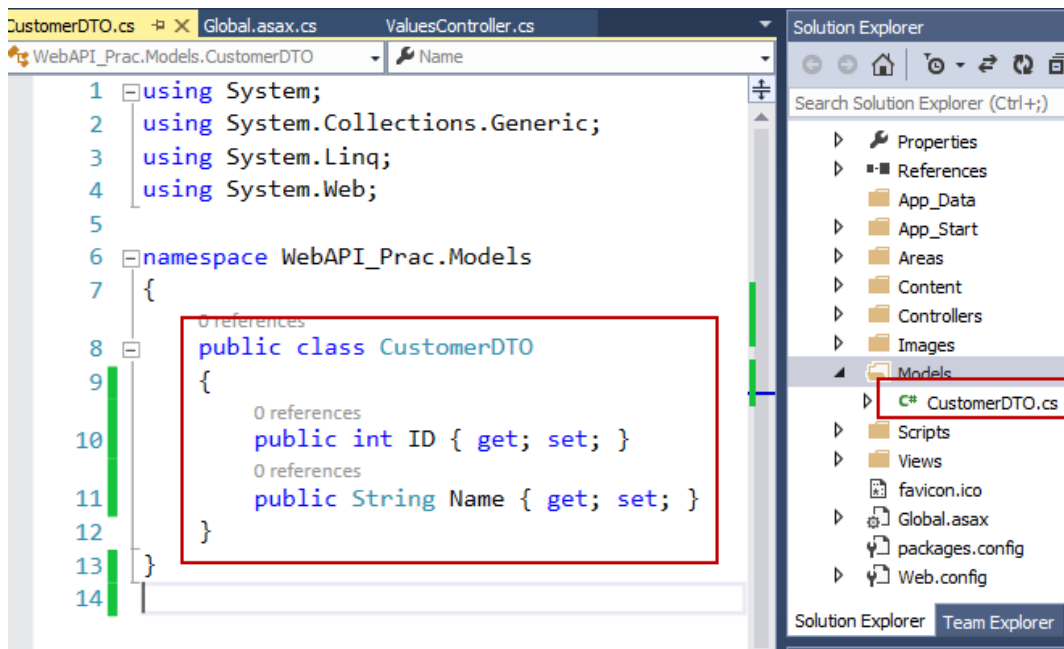
- 4- In Controllers folder, you will see an “API” controller with name “ValuesController”. Open this file and remove all content of “ValuesController” class. Add following methods (as shown in the screenshot). Note that this is just like MVC controller but it is extending “ApiController”. Also you can see the URLs which will be used to access specific actions (based on the routing we’ve configured above).



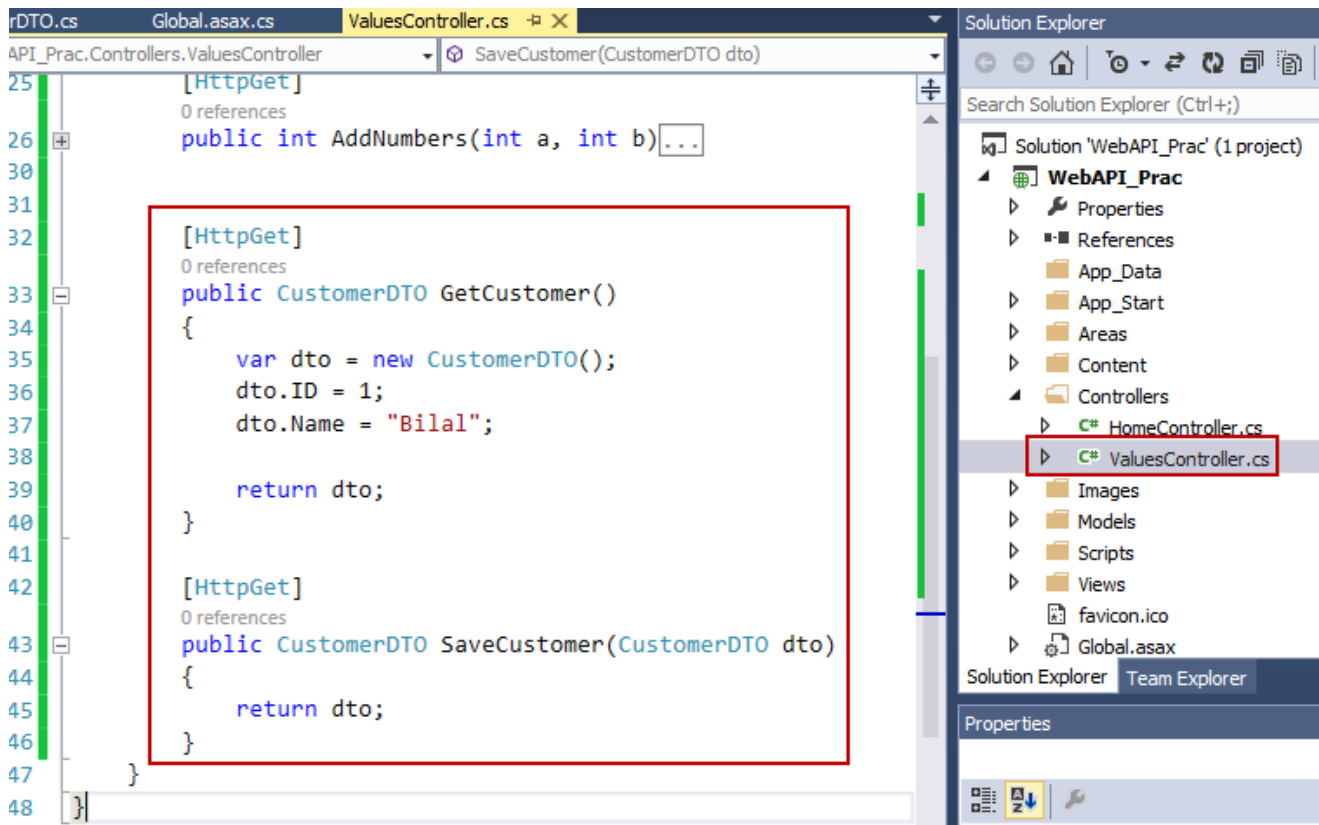
- 5- Run the project and type following URLs in browser. You will see that output but in XML format. Default formatter is XML. Though we can change it. (Change “localhost:10915” to base path on which your application will start running)



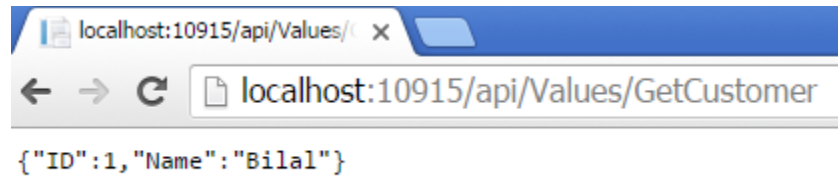
- 8- Add a new class “CustomerDTO” in Models folder and change it as shown in screenshot below.



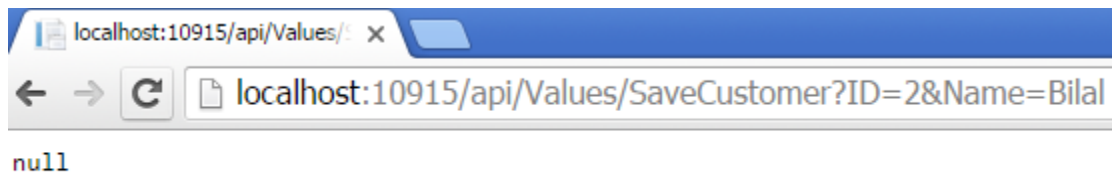
- 9- Add following two methods which are using “CustomerDTO” (complex type) as input/return value. Note: You will have to include reference of “CustomerDTO” namespace.



10- Run project and access following URLs to see if output is appearing as intended.

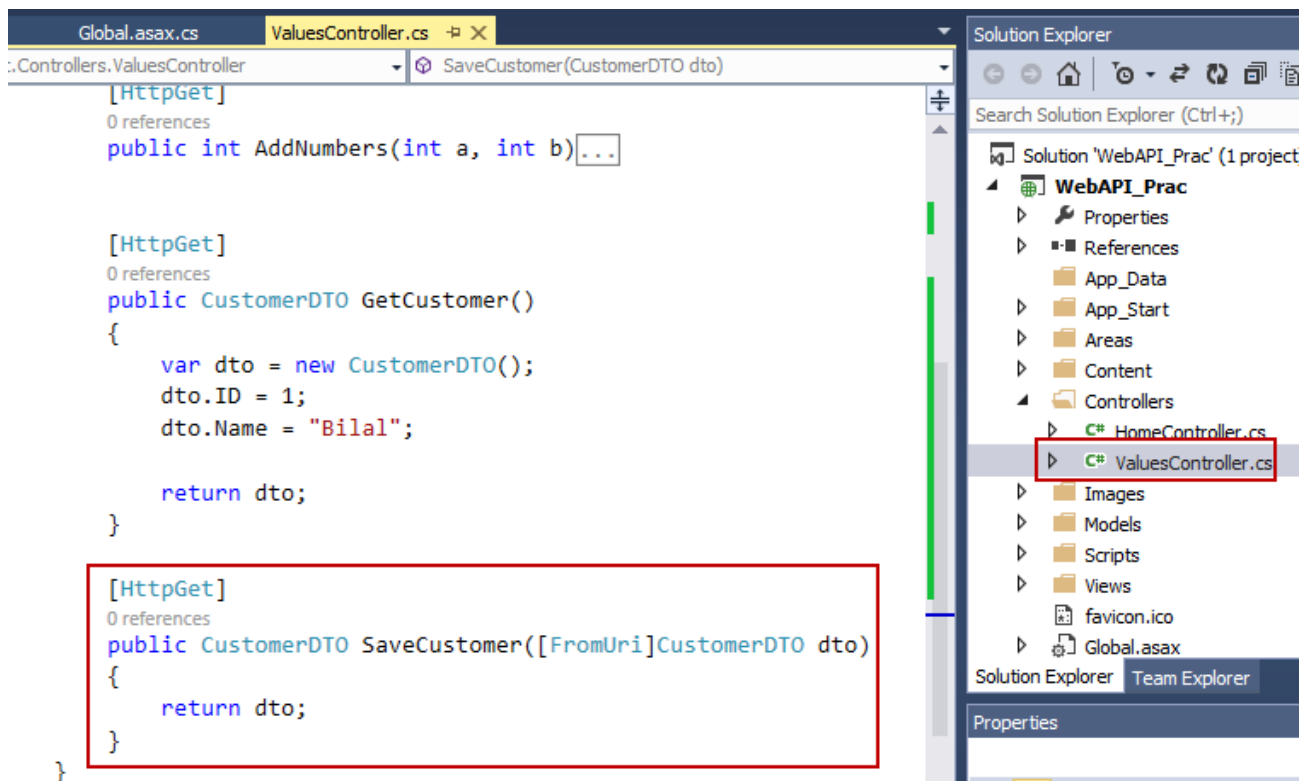


11- As method type is “GET”, parameters will be passed in query string. So let’s execute the following URL by providing parameters as query string. But when you will run this, it will return null (It should have returned same data though).

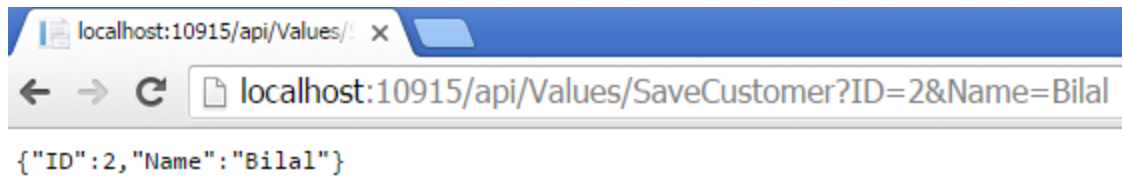


12- What is issue with above? Actually when you use “Complex type” as parameter of the function, API model binder tries to construct object from “Request” body. Here we are passing parameters as query string (i.e. in header) so ‘dto’ in code is receiving null.

13- Now to tell model binder to construct object by reading values from query string, we’ll have to use [FromUri] attribute with our complex type.



14- Now run project and again access following URL. You will see correct output this time.



15- Please check & memorize this detail (with understanding).

Request Method Type	Data is sent to server in?
GET	Header (query string)
POST	Body

Action Parameters Type	Where Does Model Binder Looks for? (By default)
Primitives (e.g. int, string)	Query String (Header)
Complex Type	Body

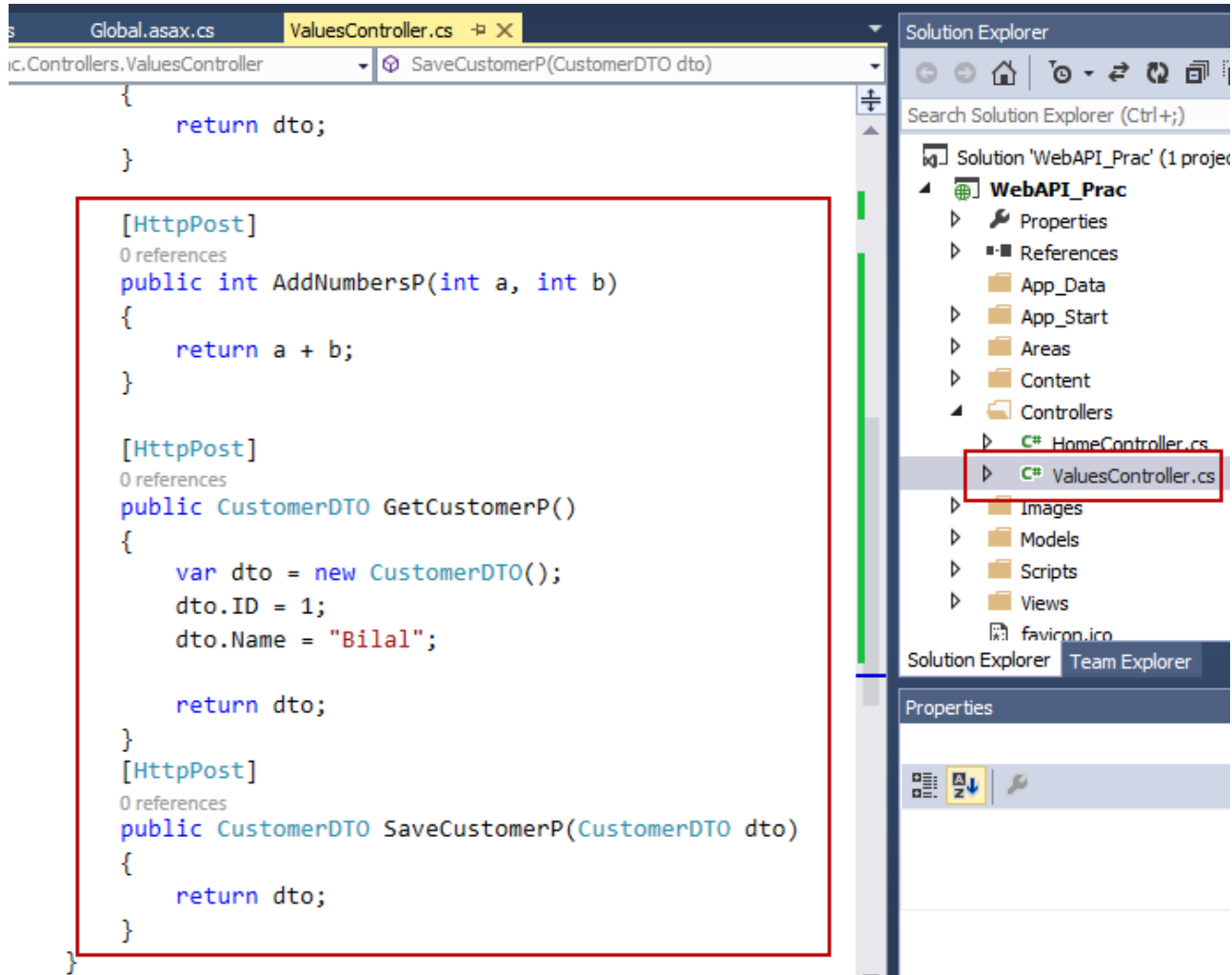
Question: How to enforce model binder to look for values from Query String (header)?

Answer: By using [FromUri] attribute with parameter

Question: How to enforce model binder to look for values from Body?

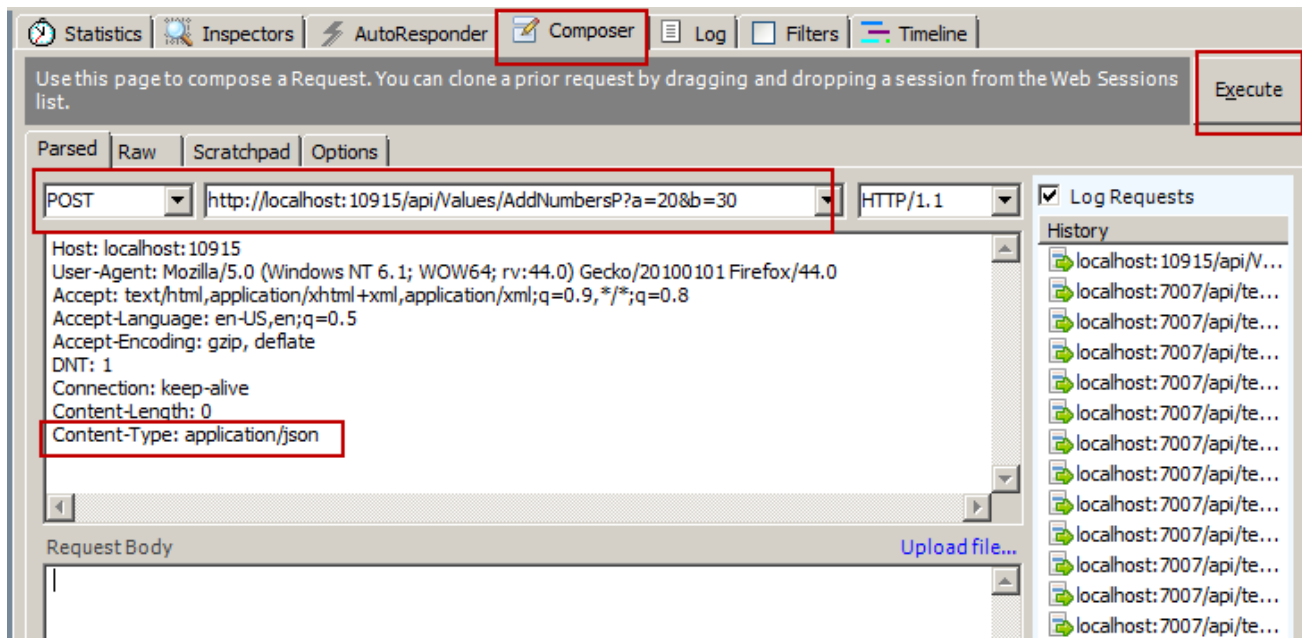
Answer: By using [FromBody] attribute with parameter

16- Let's add following three methods in "ValuesController" class. Note that these methods have method type as POST.



17- Till now, we've seen different scenarios to make hit to API but only by "GET" method. Now let's see how to make "POST" hits to our API controller. As we can't make POST hits directly by using URL bar, so we'll use "Fiddler" here. You may use any other 3rd party or browser plugin which will help you making "POST" hits.

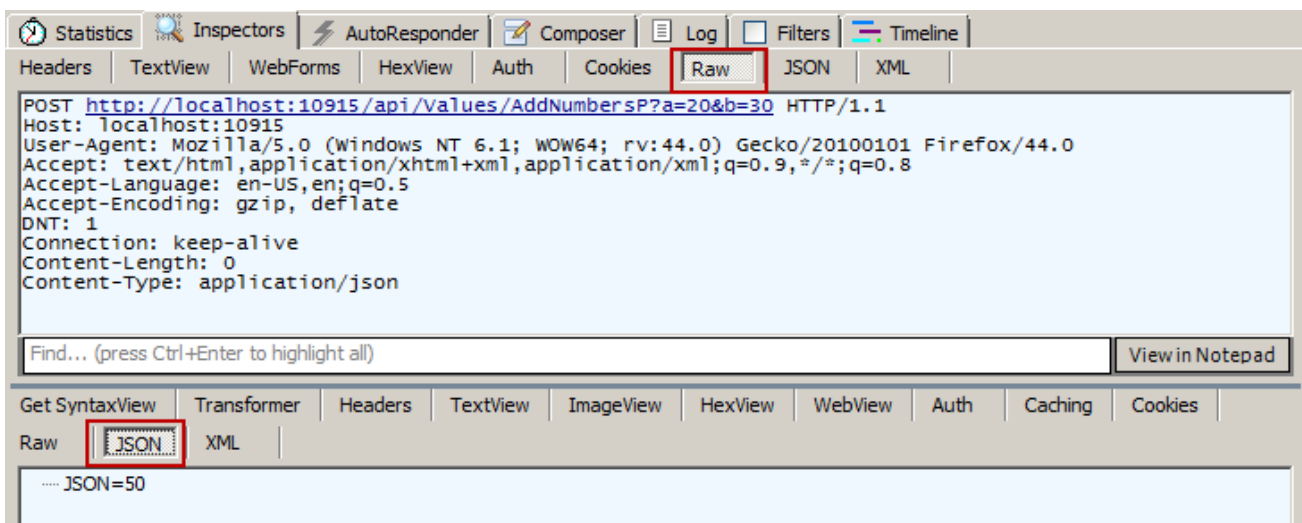
- 18- In Fiddler, Go to “Composer” Tab, choose “POST” and provide URL. As “AddNumbers” method is taking primitive parameter so model binder will expect parameters in query string. Therefore, we are providing parameters with URL. (Body is empty in this case). Also note that we’ve added “Content-Type:application/json”. Click “Execute” button. It will make hit to server.



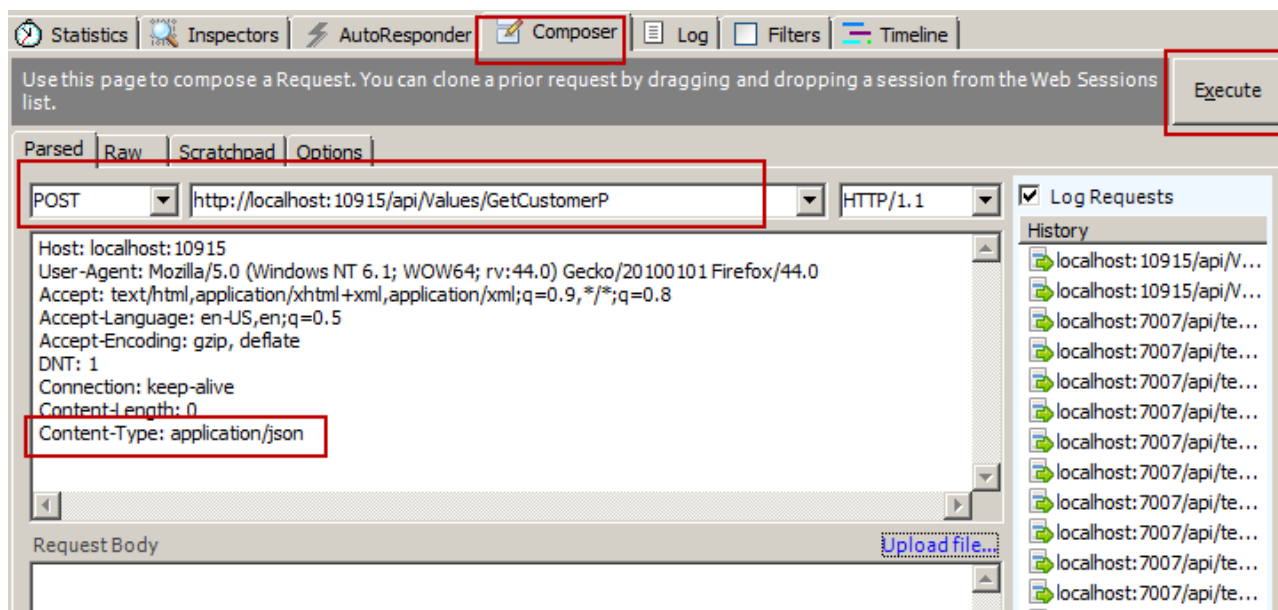
- 19- In left panel, you will see entry of our request. Double click on it to open its detail in right panel.

#	Result	Protocol	Host	URL	Body	Caching	Cc
1	200	HTTP	localhost:10915	/api/Values/AddNumbersP...	2	no-cac...	ap

- 20- In right panel, select “Inspectors=> Raw”. Top panel is showing the detail of request (which was sent to server) and bottom panel is showing the response received from server.



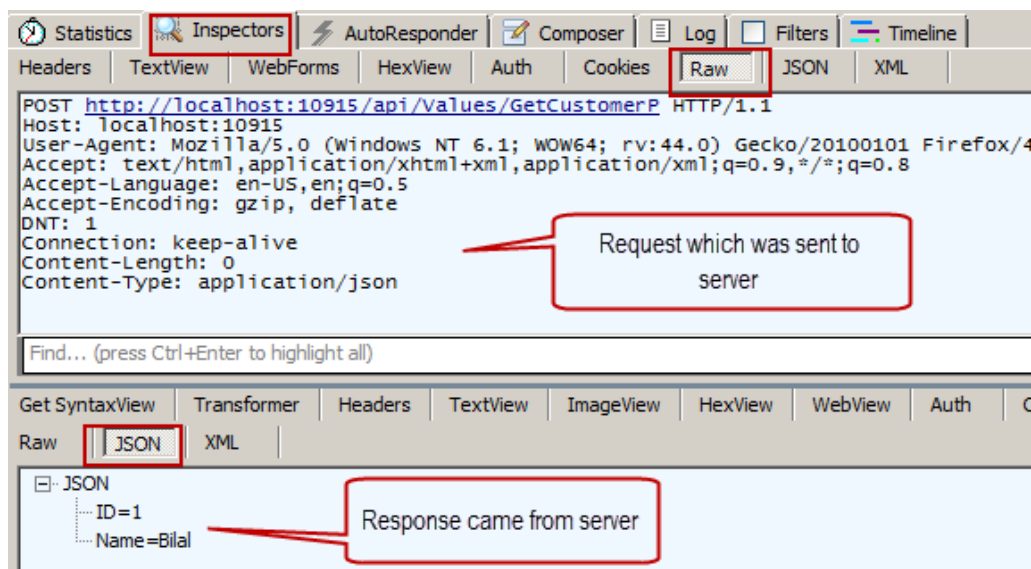
21- Now to make call to second method, again open “Composer” tab. Provide the URL. This action doesn’t take any input. Click on Execute. This will make hit to server.



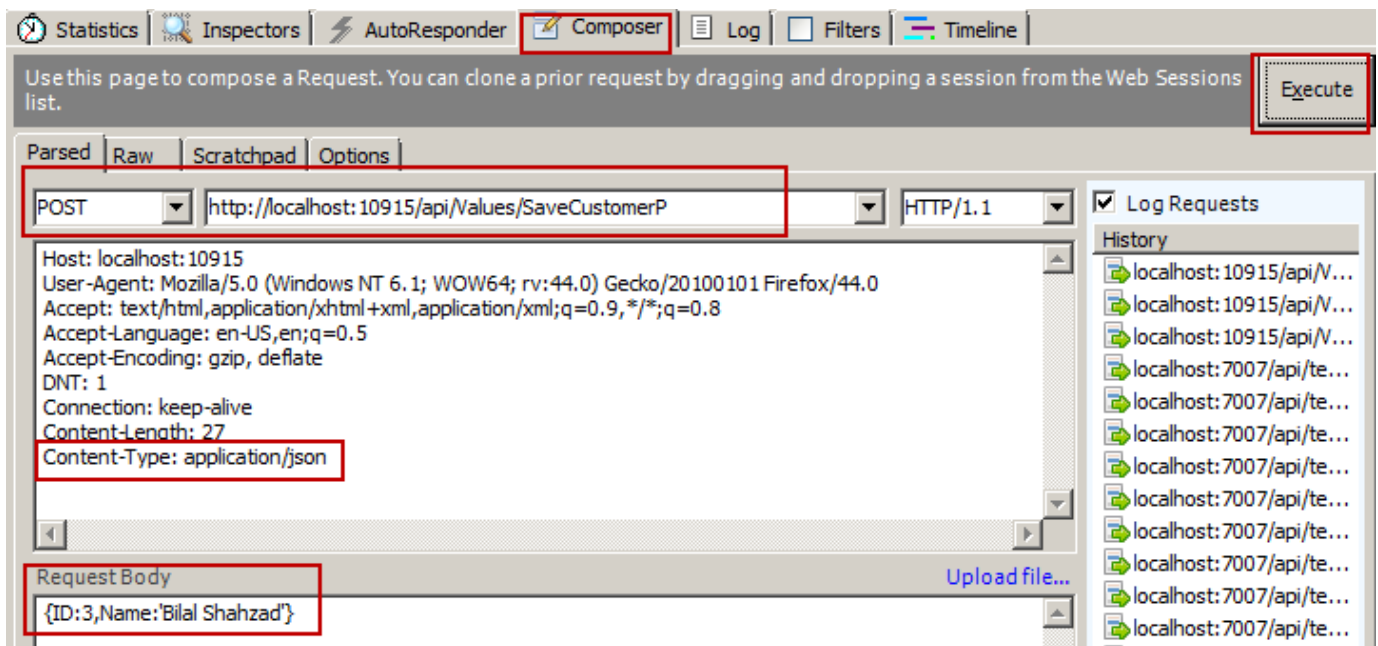
22- In left panel, you will see entry of our request. Double click on it to open its detail in right panel.

#	Result	Protocol	Host	URL	Body	Caching	Co
1	200	HTTP	localhost:10915	/api/Values/AddNumbersP...	2	no-cac...	ap
2	200	HTTP	localhost:10915	/api/Values/GetCustomerP	23	no-cac...	ap

23- Here you can see details of “Request” & “Response”.



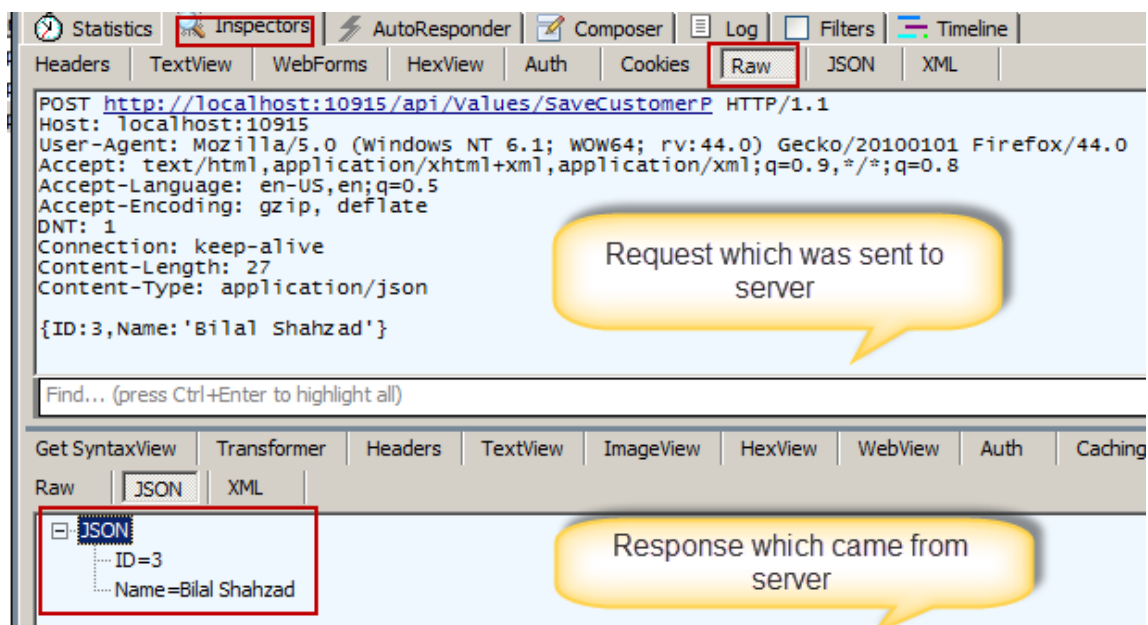
24- Now to make call to third method. Again open “Composer” tab. Provide the URL. This action takes an object of complex type so it would be expecting data in “Body”. Note we’ve provided “content-type” to json and data in JSON format in “Request Body”.



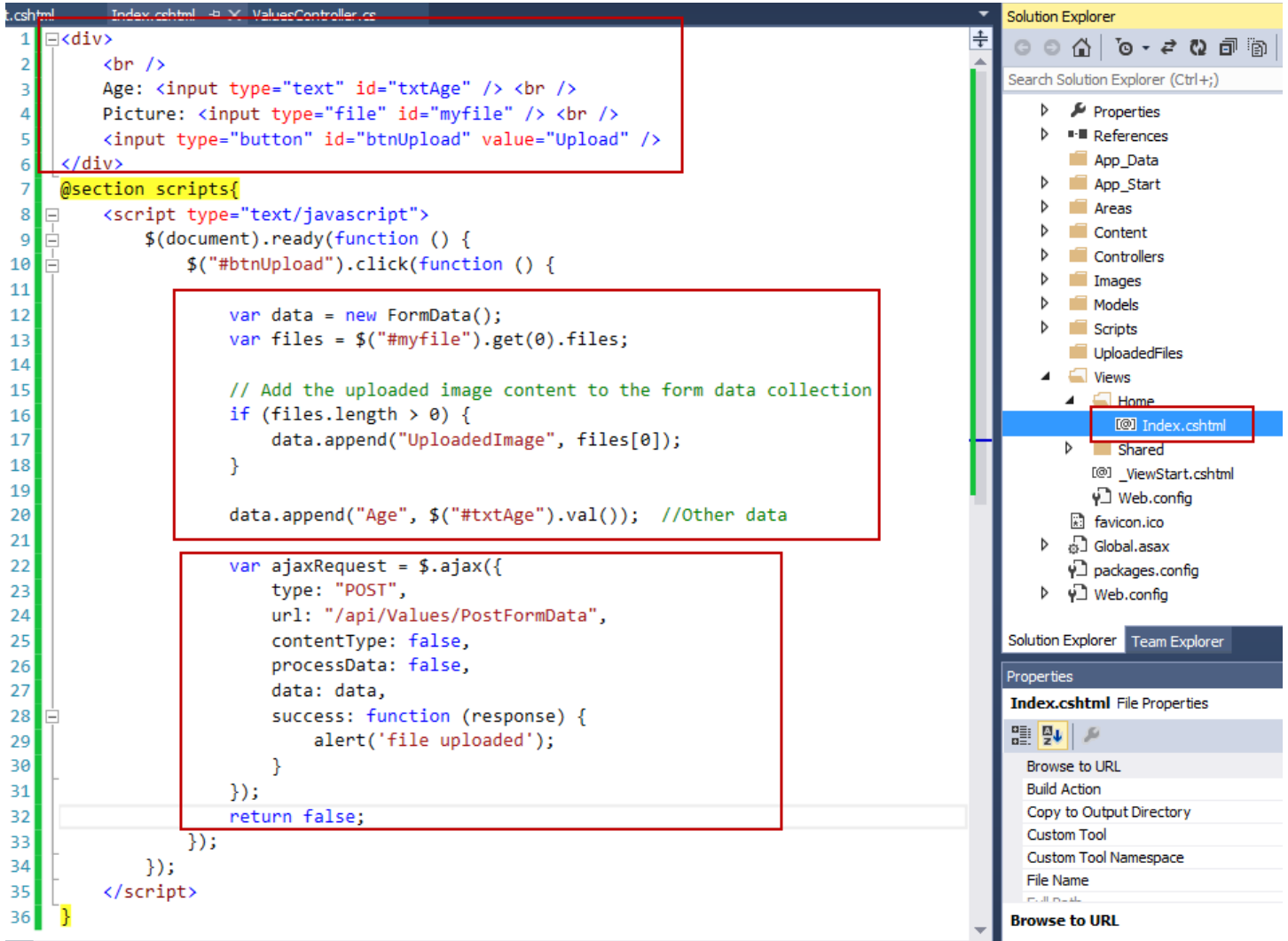
25- In left panel, you will see entry of our request. Double click on it to open its detail in right panel.

#	Result	Protocol	Host	URL	Body	Caching	Cc
1	200	HTTP	localhost: 10915	/api/Values/AddNumbersP...	2	no-cac...	ap
2	200	HTTP	localhost: 10915	/api/Values/GetCustomerP	23	no-cac...	ap
12	200	HTTP	localhost: 10915	/api/Values/SaveCustomerP	31	no-cac...	ap

26- Here you can see details of “Request” & “Response”.



27- Now let's see how to POST a file to "WEB API" controller using AJAX. Open "Index.cshtml" file as shown file and remove all its content. Add following content in this file. Here we are creating a "text" control, a "file" control and a "button" control. User will provide some value for "age", will select a file to upload and will click on "Upload" button. A "FormData" object will be constructed and content will be placed in it. Then using AJAX, data will be posted to WEB API controller.



```
1 <div>
2   <br />
3   Age: <input type="text" id="txtAge" /> <br />
4   Picture: <input type="file" id="myfile" /> <br />
5   <input type="button" id="btnUpload" value="Upload" />
6 </div>
7 @section scripts{
8   <script type="text/javascript">
9     $(document).ready(function () {
10       $("#btnUpload").click(function () {
11
12         var data = new FormData();
13         var files = $("#myfile").get(0).files;
14
15         // Add the uploaded image content to the form data collection
16         if (files.length > 0) {
17           data.append("UploadedImage", files[0]);
18         }
19
20         data.append("Age", $("#txtAge").val()); //Other data
21
22         var ajaxRequest = $.ajax({
23           type: "POST",
24           url: "/api/Values/PostFormData",
25           contentType: false,
26           processData: false,
27           data: data,
28           success: function (response) {
29             alert('file uploaded');
30           }
31         });
32         return false;
33       });
34     });
35   </script>
36 }
```

Solution Explorer

- Properties
- References
- App_Data
- App_Start
- Areas
- Content
- Controllers
- Images
- Models
- Scripts
- UploadedFiles
- Views
 - Home
 - Index.cshtml
 - Shared
- _ViewStart.cshtml
- Web.config
- favicon.ico
- Global.asax
- packages.config
- Web.config

Solution Explorer Team Explorer

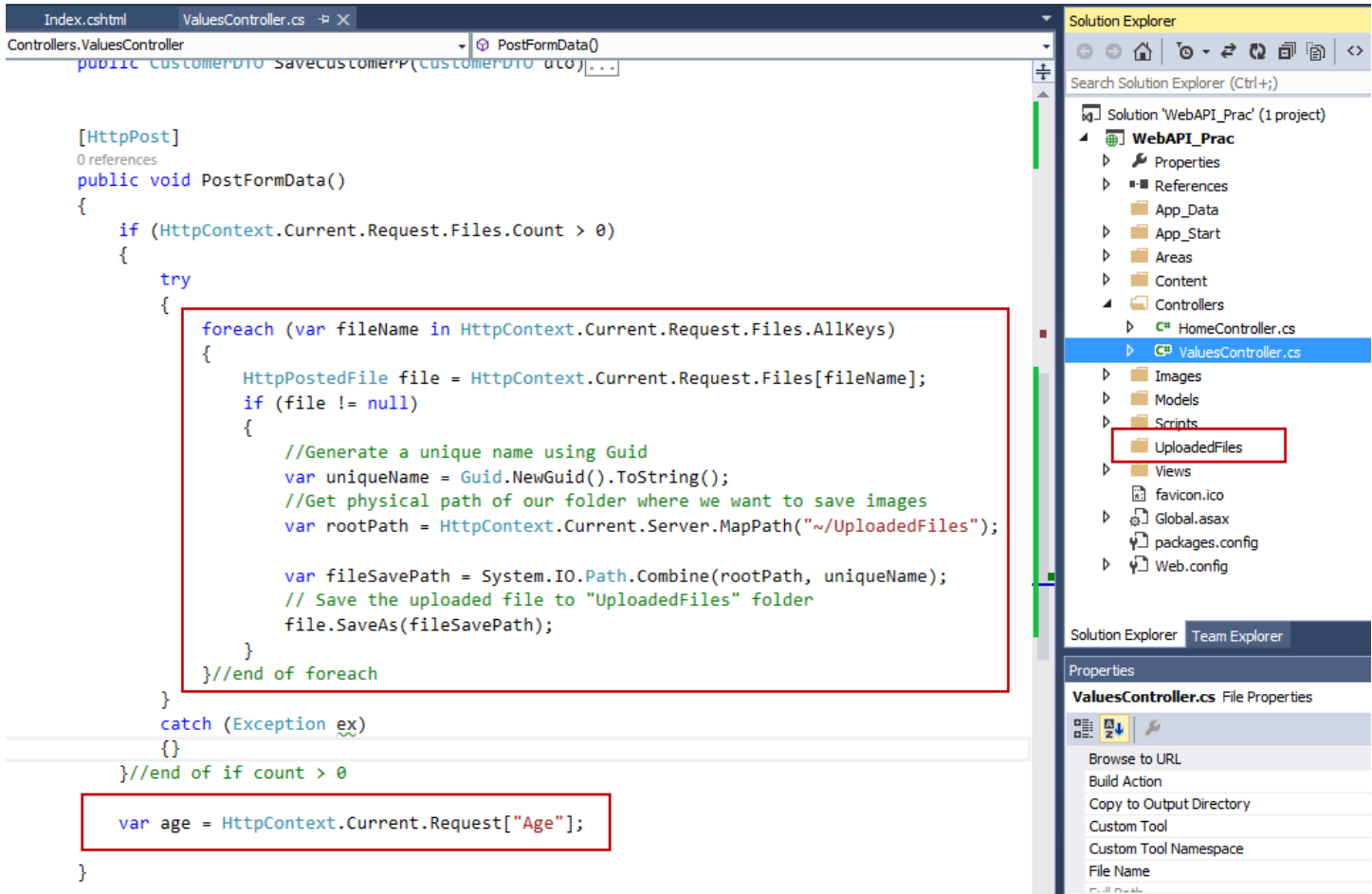
Properties

Index.cshtml File Properties

- Browse to URL
- Build Action
- Copy to Output Directory
- Custom Tool
- Custom Tool Namespace
- File Name
- Full Path

Browse to URL

28- Add following action in “Values” controller. Here we are not using default Model binder as that will not work. Instead, we are accessing our “Request” object directly. As this is not MVC controller so “Request” object will not be available as part of class. We are accessing it by “System.Web” (namespace) and then “HttpContext.Current.Request”. Remaining code is same which we’ve already seen while uploading file in ASP.NET web forms and ASP.NET MVC. Also note that, we are receiving other data as well using “Request”. We are saving our files in “UploadedFiles” folder so you will have to create a folder with this name in your project as shown in screenshot below.



```
Index.cshhtml  ValuesController.cs  PostFormData()
Controllers.ValuesController
public CustomerDTO SaveCustomerP(CustomerDTO dto)...

[HttpPost]
0 references
public void PostFormData()
{
    if (HttpContext.Current.Request.Files.Count > 0)
    {
        try
        {
            foreach (var fileName in HttpContext.Current.Request.Files.AllKeys)
            {
                HttpPostedFile file = HttpContext.Current.Request.Files[fileName];
                if (file != null)
                {
                    //Generate a unique name using Guid
                    var uniqueName = Guid.NewGuid().ToString();
                    //Get physical path of our folder where we want to save images
                    var rootPath = HttpContext.Current.Server.MapPath("~/UploadedFiles");

                    var fileSavePath = System.IO.Path.Combine(rootPath, uniqueName);
                    // Save the uploaded file to "UploadedFiles" folder
                    file.SaveAs(fileSavePath);
                }
            }
        }
        catch (Exception ex)
        {
        }
    }
}

var age = HttpContext.Current.Request["Age"];
}
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'WebAPI_Prac' (1 project)

- WebAPI_Prac
 - Properties
 - References
 - App_Data
 - App_Start
 - Areas
 - Content
 - Controllers
 - HomeController.cs
 - ValuesController.cs
 - Images
 - Models
 - Scripts
 - Views
 - favicon.ico
 - Global.asax
 - packages.config
 - Web.config

Solution Explorer Team Explorer

Properties

ValuesController.cs File Properties

- Browse to URL
- Build Action
- Copy to Output Directory
- Custom Tool
- Custom Tool Namespace
- File Name

29- Run the project and verify if page is working as expected.

Tasks for Practice

Once you are done with above tutorial. Try to work on these tasks.

- 1- Make hits to above actions of “Values” controller through “AJAX”. You may add your HTML & JavaScript in “Views->Home->Index.cshtml”.
- 2- Make changes in “Index.cshtml” file and upload 2 images (+ age) instead of one image. Check what changes you will have to make on server side.
- 3- Explore how to enable “Cross-Origin Requests” (CORS) in ASP.NET Web API
 - a. <http://www.asp.net/web-api/overview/security/enabling-cross-origin-requests-in-web-api>

Useful Links

<http://www.asp.net/web-api>

<http://www.asp.net/web-api/overview/formats-and-model-binding>

<http://www.asp.net/web-api/overview/formats-and-model-binding/json-and-xml-serialization>

<http://stackoverflow.com/questions/27863434/upload-file-using-webapi-ajax>