

COMPUTING SUBJECT: Restful ASP.Net Core-services

TYPE: Assignment

IDENTIFICATION: RestCalculatorService

COPYRIGHT: *Michael Claudius & Peter Levinsky*
Pictures upgraded to ASP.Net 3.0 by Zuhair Khan

LEVEL: Medium

TIME CONSUMPTION: 2-5 hours

EXTENT: 60 lines

OBJECTIVE: Restful services based on ASP.Net Core 3.1

PRECONDITIONS: Rest service theory. Http-concepts
Computer Networks Ch. 2.2
Telerik Fiddler installed

COMMANDS:

IDENTIFICATION: RestCalculator /MICL&PELE

Purpose

The purpose of this assignment is to be able to provide and consume restful ASP.Net Core web services.

Secondary purpose is to utilize the web debugging tool Fiddler. Before start reading more you must watch the video and install Fiddler:

Video: <https://www.telerik.com/videos/fiddler>

Download: <https://www.telerik.com/fiddler>

Mission

You are to make and use restful web services based on the ASP.Net Core services by setting up a server (provider), test the services by use of Fiddler/Postman and create a client (consumer) using the services provided. On the way you will publish the service to the cloud (Azure). The service supports the classic GET, POST, PUT and DELETE requests. This we shall do in 12 steps:

1. Create a project with auto generated service: api/values
2. Test the services using Browser
3. Test the services using Fiddler/Postman
4. Investigate various possibilities in the Solution Explorer and Execution mode.
5. Create a model class Data for holding numbers
6. Create a controller CalculatorController to provide a REST services
7. Create and provide a controller-oriented service in CalculatorController
8. Test the service using Fiddler/Postman
9. More services and testing by Fiddler/Postman
10. Create a client/consumer utilizing the services
11. Publish the service to Azure
12. Refactoring the consumer code
13. Appendix A on a small technical issue

This assignment holds all 12 steps, where the first 4 is more like a tutorial done and guided by your teacher and the last ones you are doing more independently.

Later in another assignment, RestCustomerService, you will create a Rest service for management and administration of customer-objects, and in last assignment, RestCustomerService No. 2, learn elementary unit testing and how to support Cross Origin Resource Sharing (CORS).

Domain description

First (1-4), we shall just utilize the simple auto generated web service defined by ValuesContolller. Second (5-12), we shall program web services for the four the basic arithmetic calculation of two integers (a, b):

Addition(a+b), Subtraction(a-b), Multiplication(a*b) and division(a/b).

When surfing on the net it is easy to find many descriptions more or less useful, and in more or less updated versions. Here are some:

Useful links for C#:

Serializable Class

[https://msdn.microsoft.com/en-us/library/4abbb6k0\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/4abbb6k0(v=vs.110).aspx)

CRUD-Operations and routing

<https://docs.microsoft.com/en-us/aspnet/web-api/overview/web-api-routing-and-actions/attribute-routing-in-web-api-2>

Useful links for Fiddler:

Video

<https://www.telerik.com/videos/fiddler>

Download

<https://www.telerik.com/fiddler>

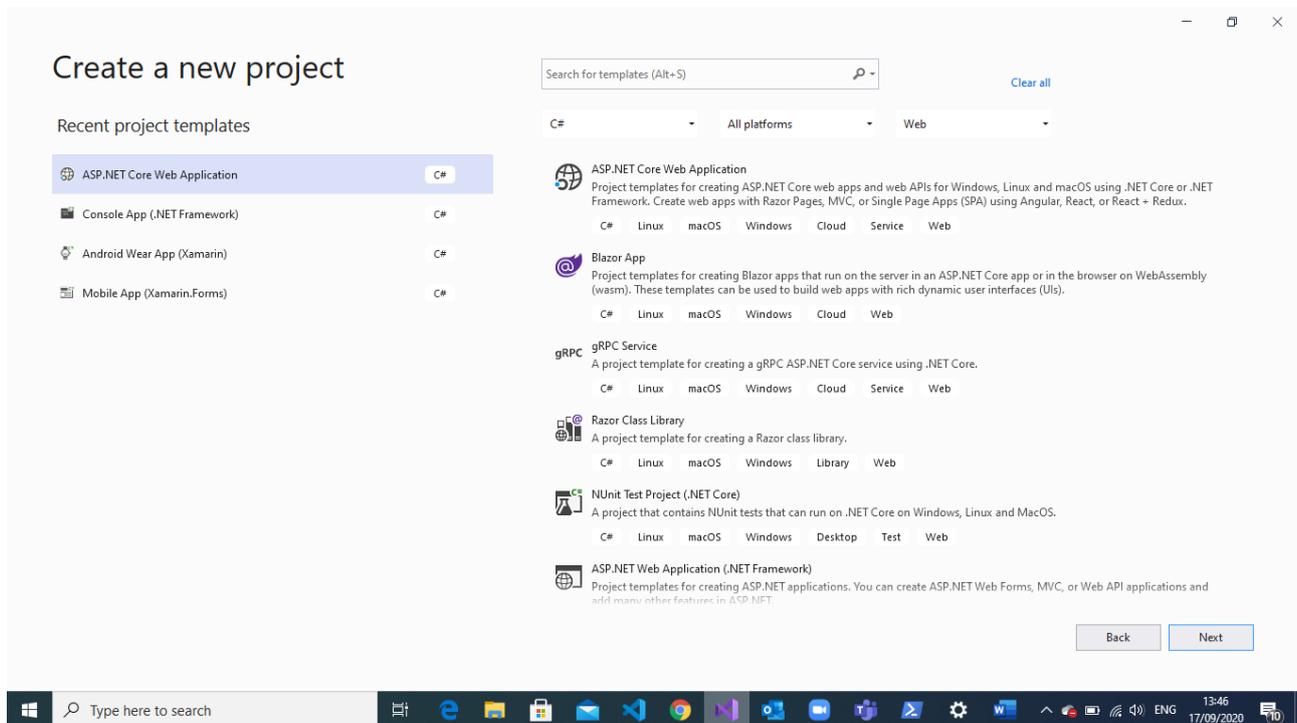
Assignment 1: Restful ASP.Net Framework-service provider

You are to make a Rest Service provider RestCalculatorService.

Start Visual Studio:File -> New -> Project.

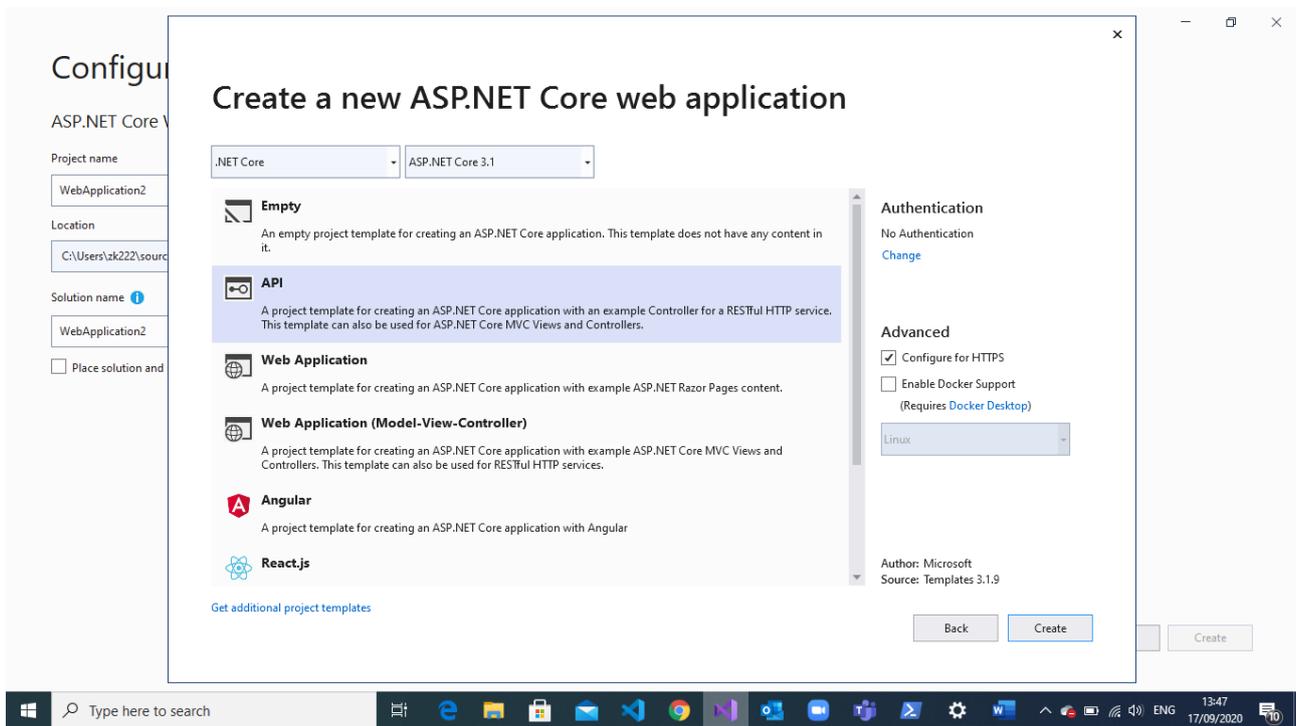
Choose: Web -> ASP.NET Core Web Application (**not .Net Framework**).

Browse to a convenient location and give the name RestCalculatorService.



Click OK.

Now a project template with options is given.



Choose the API.

DON'T tick Docker support.

Tick HTTPS if you intend to use GoogleChrome or MicrosoftEdge as browser. And Fiddler as a tester-client. This has been tested by me (Michael Claudius) and it all works fine.

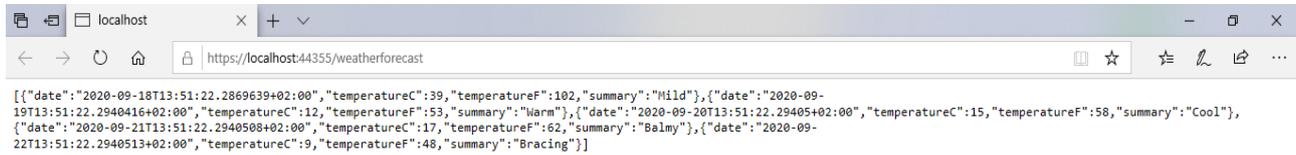
Tick HTTPS might cause problems if you intend to use Firefox as browser. And maybe, maybe not Postman, as it can have problems using https. But then of course https will **NOT** be supported later on.

Now you have to wait a while...

Then we are ready to investigate the created project.

Assignment 2: Execute the web application

Execute the web application (click the green arrow as usually) and it will open a local Browser.



Discuss what happened and try figure it out how it happened.

Assignment 3: Fiddler/Postman

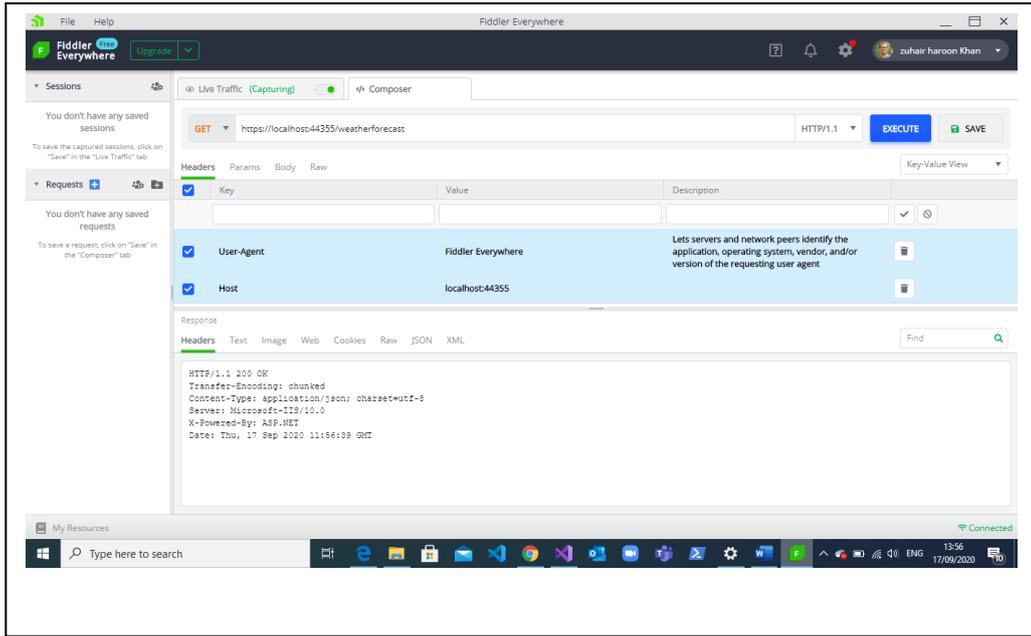
Now we will try to invoke the methods from Fiddler/Postman using the same url's as before.

Start Fiddler

Be aware that you must:

- a. Click on Composer
- b. Choose GET
- c. Copy and paste the url from Browser (<http://localhost:44355/weatherforstcast>) into the text field

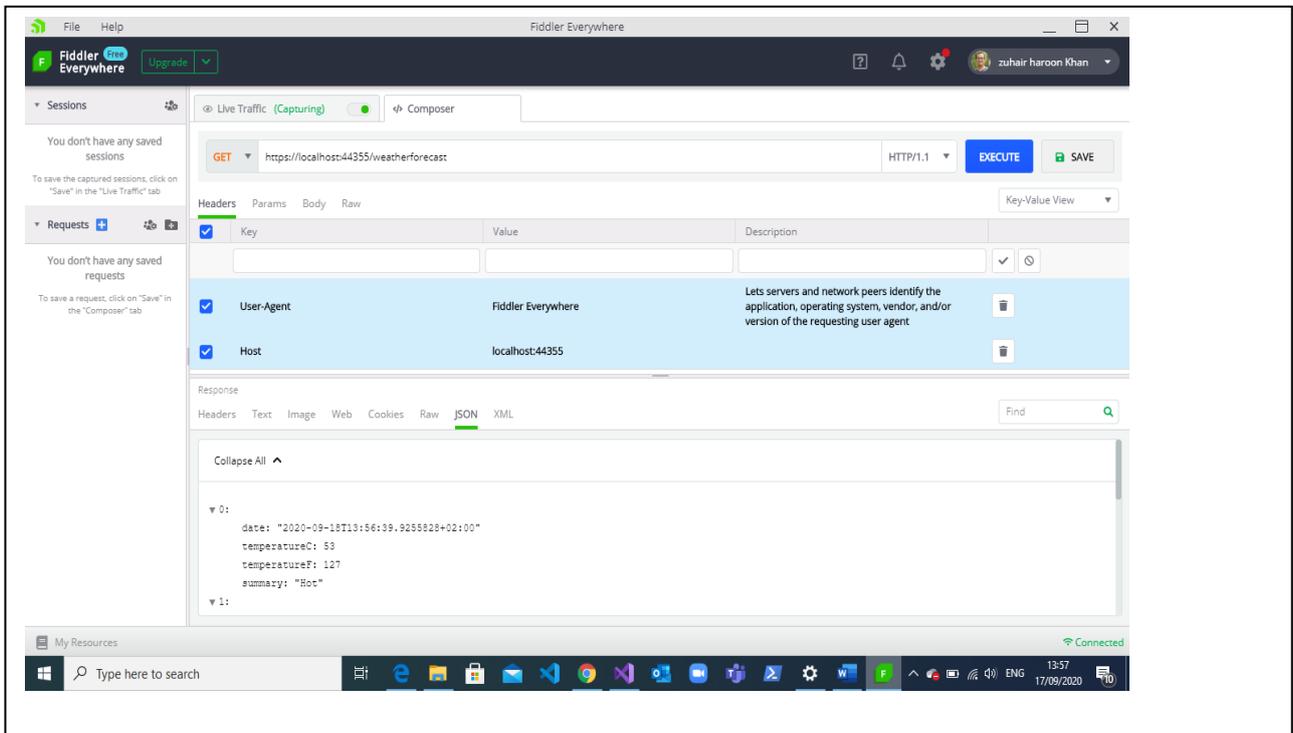
It will look something like this:



Then

- a. Click on Execute
- b. Double Click the “200 HTTPS” response line in the left column

And hopefully you will see something like this:



Finally click on Headers and Raw you can see the HTTP response.
Notice the “Content-Type: application/json; charset=utf-8”
Notice the JSON body, some array data
How are the ta display here?

Assignment 4: Exploring the set up

Some questions arise:

Where are the methods defined?
Why is it port 44343?
How can it be executed in console mode on a different port?

To find the answers take a look in Solution Explorer:

Controller -> WeatherForecastController
Properties -> LaunchSettings.json

Try to change the Get method to

```
[HttpGet("{id}")]  
public ActionResult<int> Get(int id)  
{return id;}
```

Execute again. Any difference? What happens?

Finally, change IISExpress to RestCalculatorService and run the application again.
What could be the purpose of doing this?

*This ends the investigation of the auto generated service.
In the next assignments you shall create your own service.*

Assignment 5: Model class Data

We need a class with two integers so arithmetic operations can be performed on objects of this class. Therefore to the project add (right click project, choose Add -> Folder) a folder named “Model” and in this folder add a public class, “Data”, with the data fields:

```
int A; int B
```

with get/set method for all the fields; i.e. they are properties.
Create the constructors:

```
Data(int a, int b)  
    Initializing all the data fields
```

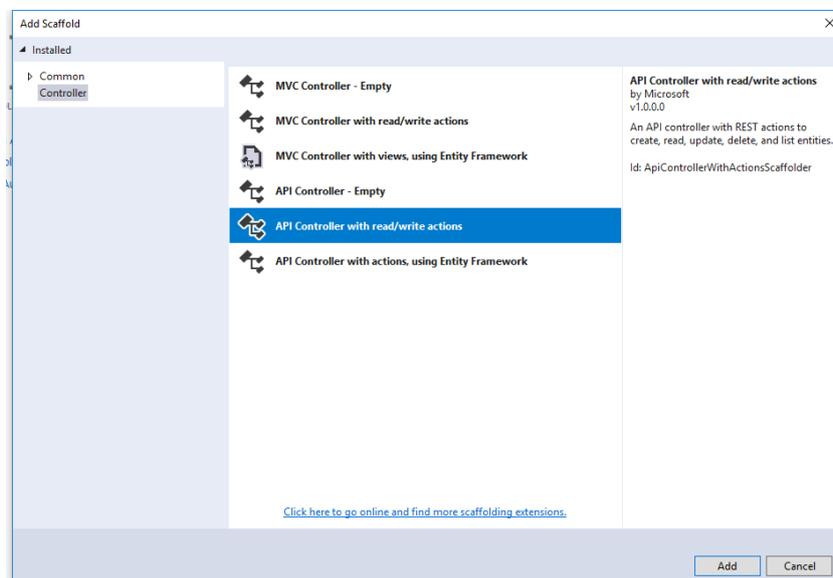
```
Data() {}  
    //empty constructor needed for JSON transfer. Serializable objects.
```

Assignment 6: REST API operation by creating a controller

You are to create a controller where the operation contracts must be defined as REST API routes and methods similar to WeatherForecast controller

In the solution in the controller folder, add (i.e. Right click) a new controller named “CalculatorController”.

Choose ‘Web API Controller **with read/write actions**’.



Click Add and you can see the new controller.

Assignment 7: Defining Service Add

In *CalculatorController* define a new operation contract to support a Rest API POST request:

```
// POST: api/Calculator
[HttpPost("Add", Name = "Add")]
public int Post([FromBody] Data data)
{
    return data.A + data.B;
}
```

And implement the method to return the sum of the integers in the data object.

What is the full route to the Add operation?

Why did we choose to change HttpPost?

Execute the Application by viewing it in a local Browser. This will start the Azure emulator.

From the browser call the calculator:

<http://localhost:44343/calculator>

<http://localhost:44343/calculator/add>

Note probably another port number and note also what you indicate as route is part of your URL to the Rest Service.

All fine?!

Nope...Really boring right!!?

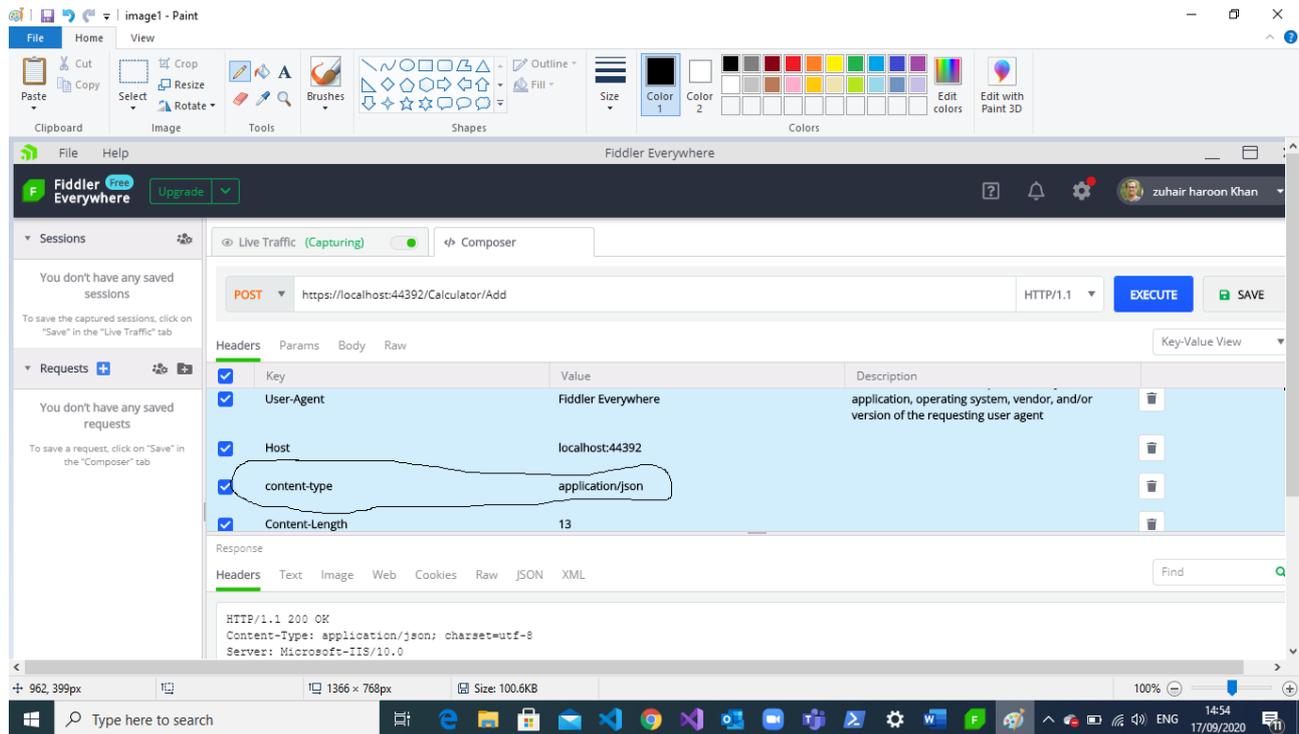
Assignment 8: Running from Fiddler/Postman

Try to invoke the method from Fiddler/Postman

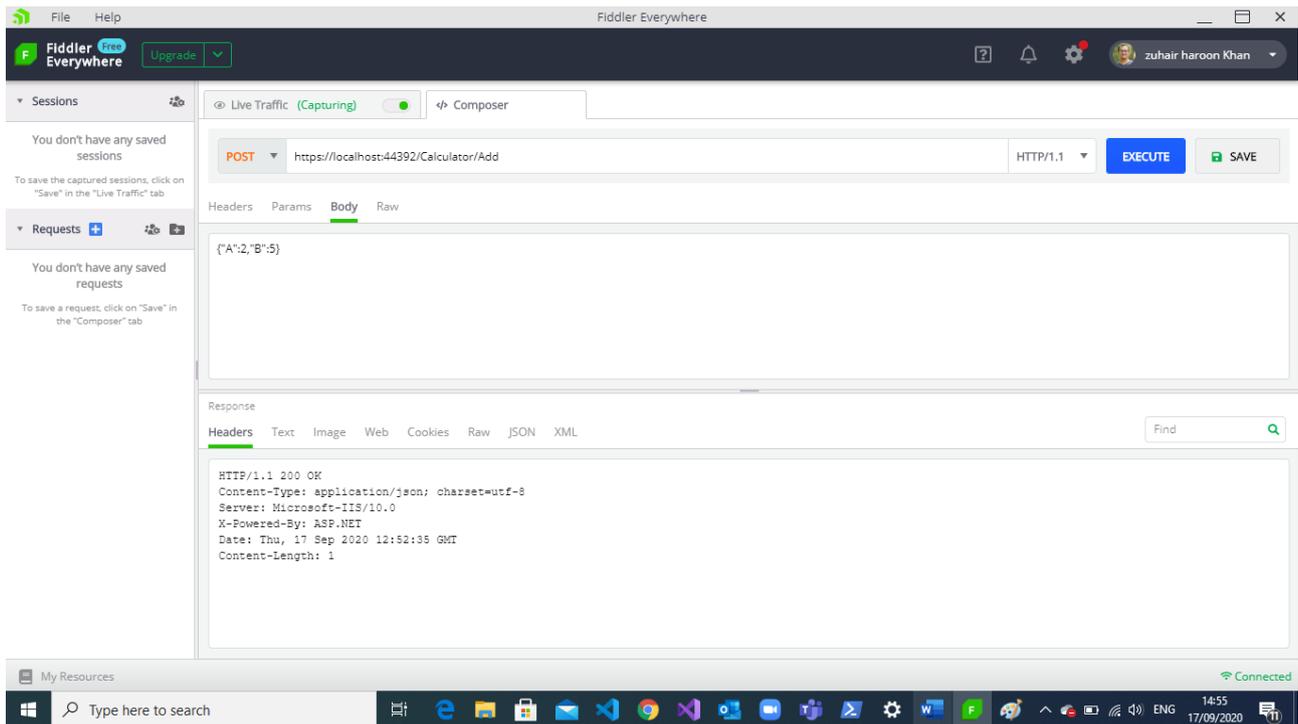
Be aware that you must:

- d. Click on Composer
- e. Choose POST
- f. In the header write/define the Content-Type: application/json
- g. Request body must hold the data as a json-string like: {"A":27,"B":4}

It will look something like this:



- h. Click on *Execute* and hopefully you get the sum.



Note: Fiddler might be limited to only use the IISExpress version (one student out of 50 will encounter this problem ☹️)

Assignment 9: More services

You must now extend the service (i.e. your controller) with more methods. In CalculatorController define more operations handling

- a. Subtraction a-b
- b. Multiplication a*b
- c. Division a/b

Remember for each method you must carefully think about

- Which HTTP method/verb to use?
- What should the URI (Route) look like? Any parameters to the URI, like {id}?
- Return type: true/false, customer object, id etc..

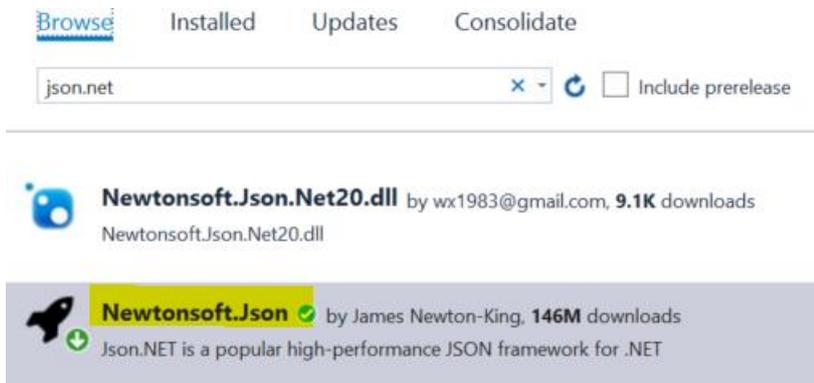
And write down your arguments.

For each method, show how to use it from Fiddler/Postman.

Assignment 10: Program for consuming services: RestCalculatorConsumer

Create a simple Console Core Application project. Add Data class to the project, similar to the one you used in the provider.

In order to serialize/deserialize objects, you must from NuGet install the package Newtonsoft.json.



Now to consume the “Add” service, you in Program class (i.e. **Not inside main**) make a very special method:

```
public static async Task<string> AsyncAdd(Data data)
{
    using (HttpClient client = new HttpClient())
    {
        Console.WriteLine("Data " + data);

        var jsonString = JsonConvert.SerializeObject(data);
        Console.WriteLine("json string: " + jsonString);

        StringContent content = new StringContent(jsonString, Encoding.UTF8, "application/json");
        Console.WriteLine("content: : " + content.ToString());
        Console.WriteLine("CalculatorUri: " + CalculatorUri);

        HttpResponseMessage response = await client.PostAsync(CalculatorUri + "Add", content);
        string str = await response.Content.ReadAsStringAsync();
        //Int32 sumStr = JsonConvert.DeserializeObject<Int32>(str);
        return str;
    }
}
```

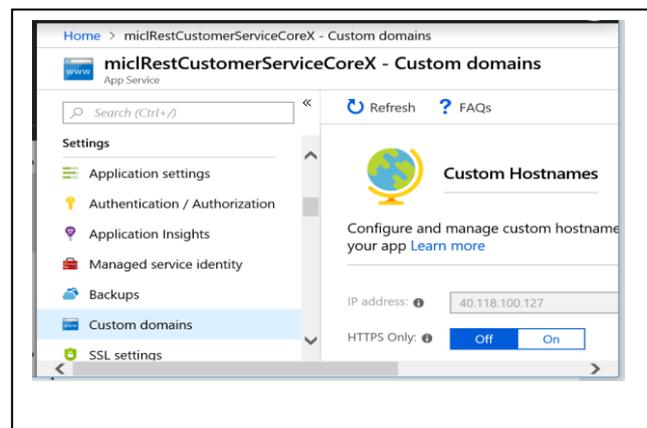
Where the CalculatorUri is a string with the URI to your service and method (api/calculator).

- In *main* show how to use the method and print out the sum of two integers.
Execute the program.
Tip: You need to get the Result value of the asynchronous Task and then change the type from string to integer.
- Carefully explain the code line-by-line what goes on.
- More calculations
Extend the program with similar methods handle the other arithmetic calculations.
Show also how to use these methods in *main*.

Assignment 11: Publish in Azure

- a. Publish your service in Microsoft Azure.
- b. Use a browser to show the API and the methods.
- c. Use Postman or Fiddler to show requests and responses.
*Unfortunately you probably get a 301/502 error security error.
Why?
The issue is that if your project was created it was configured for Https and Fiddler uses Http-scheme for Azure. Read on*

- d. Go to your Azure Portal
 1. Open your Web-App project
 2. Find *Custom domains* in the left scroll-bar
 3. Set *Https-Only* to OFF
 4. Click *Refresh*



- e. Now try again to use Postman or Fiddler to show requests and responses. *Viola ! Works!*
- f. Show how to use the Azure service instead of the local URI in your consumer program.

Assignment 12: Refactor the consumer code

Refactoring is about making the code either smarter or downsizing the number of code lines!

Take a look at your consumer code.

There is a lot of redundant code.

Can you do something about it?!

Appendix A Technical Issue to be resolved

One issue you might face is the problem of no response, when you are executing assignment no: 8 (Calculator controller). Route naming convention (below in highlighted circle) can not run your controller in the browser. All in all, you just need to replace with these code-lines.

```
[ApiController]  
[Route("[controller]")]
```

The reason why we need to replace this in the controller is that Asp.Net core 2.0 version is using calling API through the style: [Route("api/controller")].

Whereas the new version 3.0 just uses: [Route("[controller]")].

