COMPUTING SUBJECT:	Restful ASP.Net Core-services
TYPE:	Assignment
IDENTIFICATION:	RestCalculatorService
COPYRIGHT:	*Michael Claudius & Peter Levinsky*
LEVEL:	Medium
TIME CONSUMPTION:	2-5 hours
EXTENT:	60 lines
OBJECTIVE:	Restful services based on ASP.Net Core
PRECONDITIONS:	Rest service theory. Http-concepts Computer Networks Ch. 2.2
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.

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

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, 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#:

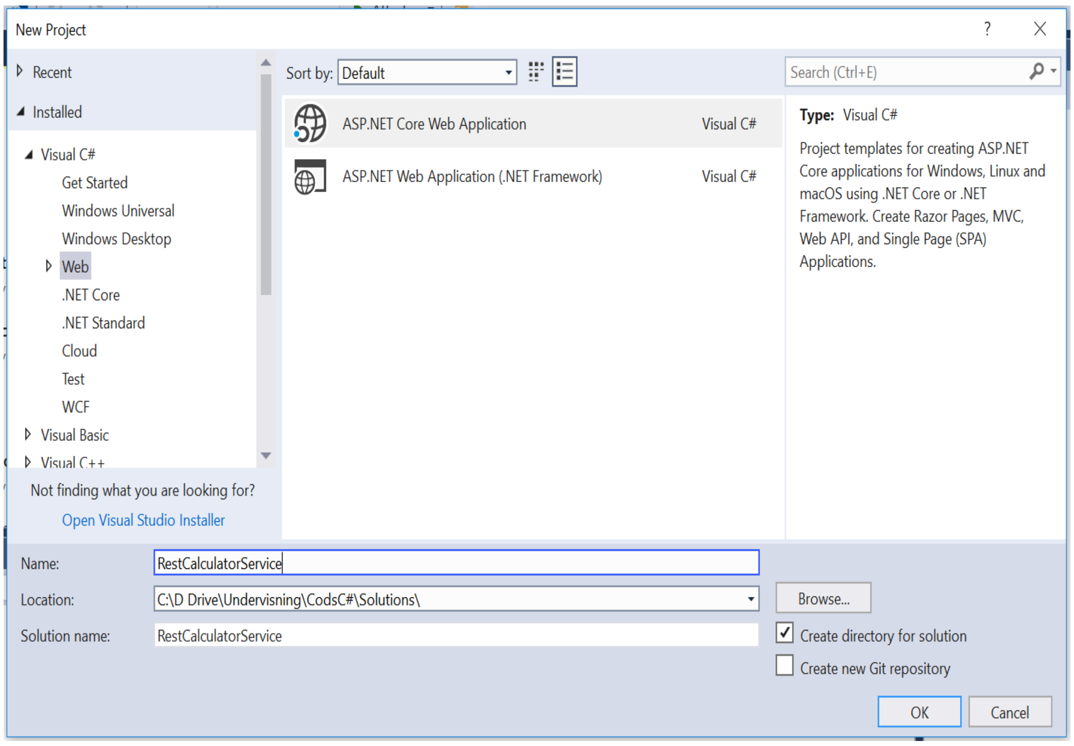
Serializable Class
https://msdn.microsoft.com/en-us/library/4abbf6k0(v=vs.110).aspx

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

Building ASP web services
https://docs.microsoft.com/en-us/aspnet/web-api/overview/web-api-routing-and-actions/create-a-rest-api-with-attribute-routing

Cors – Cross reference support
https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/enabling-cross-origin-requests-in-web-api (from the middle enable CORS)

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.

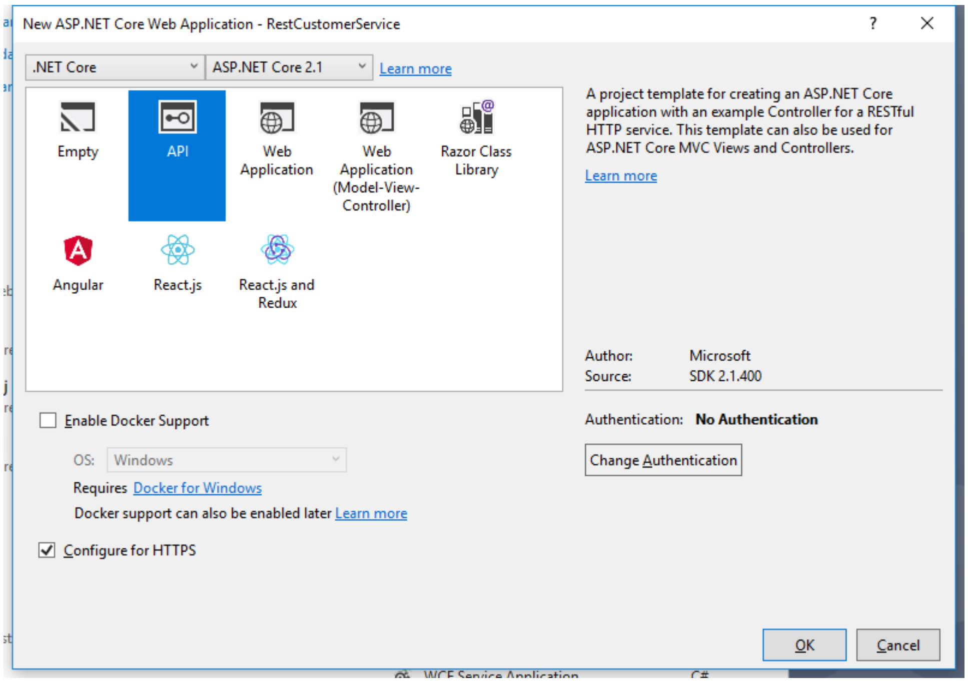


ASP.NET Core Web Application — Visual C#
ASP.NET Web Application (.NET Framework) — Visual C#

Type: Visual C#
Project templates for creating ASP.NET Core applications for Windows, Linux and macOS using .NET Core or .NET Framework. Create Razor Pages, MVC, Web API, and Single Page (SPA) Applications.

Name: RestCalculatorService
Location: C:\D Drive\Undervisning\CodsC#\Solutions\
Solution name: RestCalculatorService
[✓] Create directory for solution
[ ] Create new Git repository

Click OK.
Now a project template with options are given.


New ASP.NET Core Web Application - RestCustomerService

.NET Core | ASP.NET Core 2.1 | Learn more

A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Learn more

Author: Microsoft
Source: SDK 2.1.400

Authentication: **No Authentication**

☐ Enable Docker Support

OS: Windows

Requires Docker for Windows

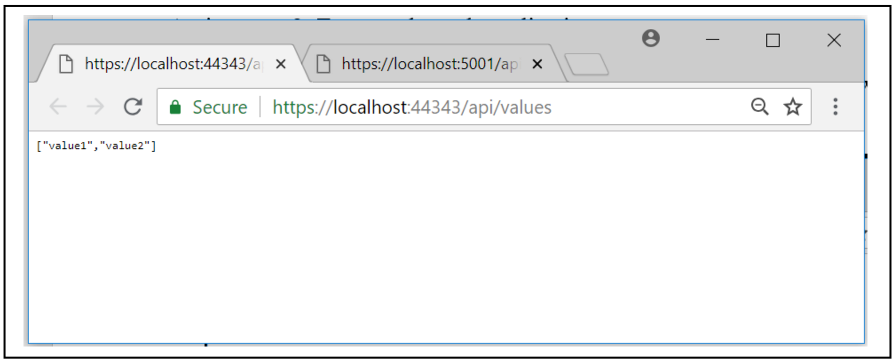
Docker support can also be enabled later Learn more

☑ Configure for HTTPS

Choose the API. **DON'T tick Docker support.**
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.



https://localhost:44343/api/values

```
["value1","value2"]
```

As you can see the application is running on port 44343 on my computer (it will be different on yours).
Try to give the url:

http://localhost:44343/api/values/5

and

http://localhost:44343/api/values/17

What do you get ?

Assignment 3: Fiddler/Postman

Try to invoke the methods from Fiddler/Postman using the same url's as before.

Assignment 4: Exploring the set up

Some questions arise:

Why does the project start with api/values?
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 -> ValuesController
Propeties -> 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:

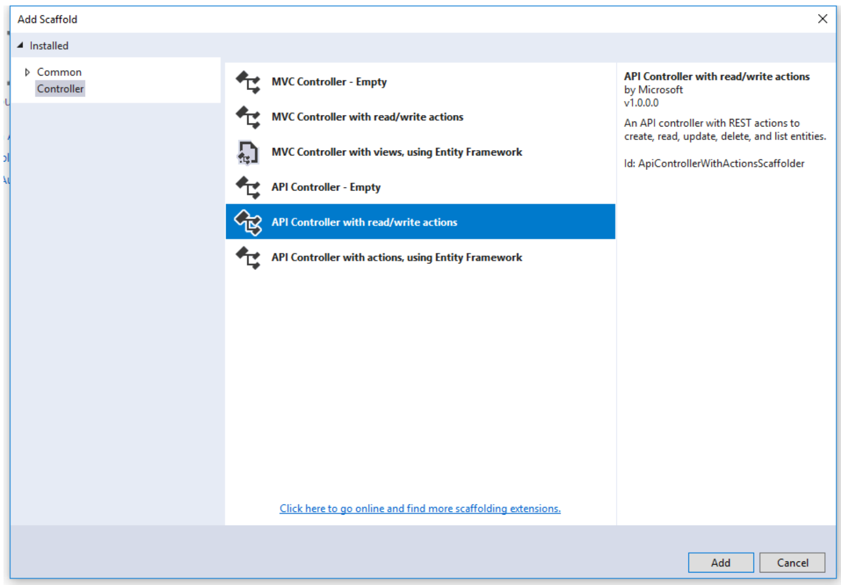
    Data(int a, int b)
    Intializing all the data fields

    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 ValuesController

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**’.


Add Scaffold
Installed
Common
Controller
MVC Controller - Empty
MVC Controller with read/write actions
MVC Controller with views, using Entity Framework
API Controller - Empty
API Controller with read/write actions
API Controller with actions, using Entity Framework
Click here to go online and find more scaffolding extensions.
Add
Cancel

API Controller with read/write actions
by Microsoft
v1.0.0.0
An API controller with REST actions to create, read, update, delete, and list entities.
Id: ApiControllerWithActionsScaffolder

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/api/calculator

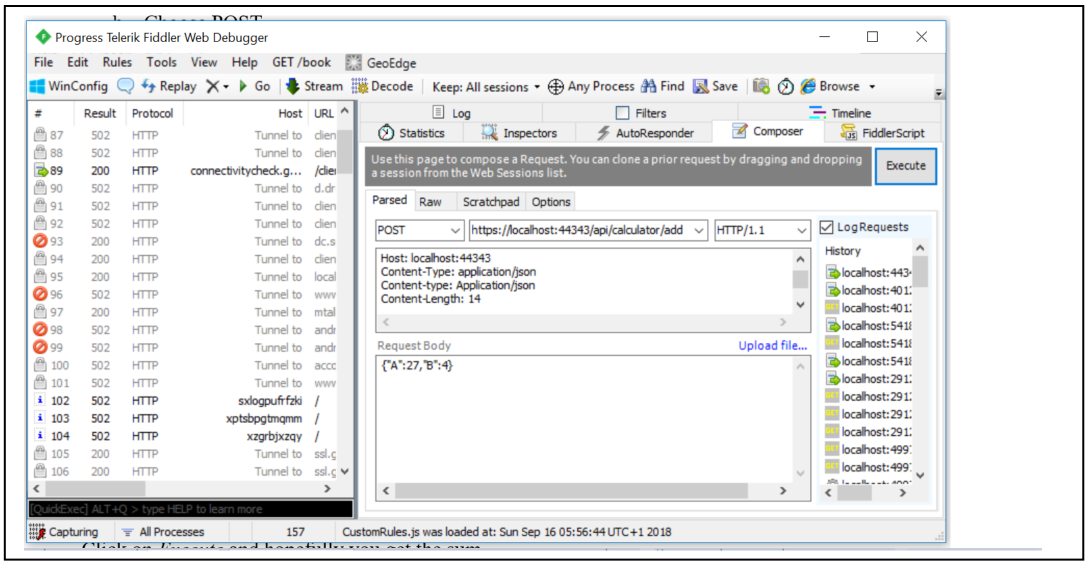
http://localhost:44343/api/calculator/add

Note probably another port number and note also what you indicate as route is part of your URL to the Rest Service.
All fine?!.
Nopre…Really boring right!!?

Assignment 8: Running from Fiddler/Postman
Try to invoke the method from Fiddler/Postman
Be aware that you must:

a. Click on Composer
b. Choose POST
c. Define the Content-Type: application/json
d. Request body must hold the data as a Json-string

It will look something like this:


Progress Telerik Fiddler Web Debugger
File Edit Rules Tools View Help GET /book GeoEdge
WinConfig Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse
Log Filters Timeline
Statistics Inspectors AutoResponder Composer FiddlerScript
Execute
Parsed Raw Scratchpad Options
Log Requests
History
Upload file...
Capturing All Processes 157

#	Result	Protocol	Host	URL
87	502	HTTP	Tunnel to	clien
88	502	HTTP	Tunnel to	clien
89	200	HTTP	connectivitycheck.g...	/clie
90	502	HTTP	Tunnel to	d.dr
91	502	HTTP	Tunnel to	clien
92	502	HTTP	Tunnel to	clien
93	200	HTTP	Tunnel to	dc.s
94	200	HTTP	Tunnel to	clien
95	200	HTTP	Tunnel to	local
96	502	HTTP	Tunnel to	www
97	200	HTTP	Tunnel to	mtal
98	502	HTTP	Tunnel to	andr
99	502	HTTP	Tunnel to	andr
100	502	HTTP	Tunnel to	accc
101	502	HTTP	Tunnel to	www
102	502	HTTP	sxlogpufrfzki	/
103	502	HTTP	xptsbpgtmqmm	/
104	502	HTTP	xzgrbjxzqy	/
105	200	HTTP	Tunnel to	ssl.g
106	200	HTTP	Tunnel to	ssl.g

Use this page to compose a Request. You can clone a prior request by dragging and dropping a session from the Web Sessions list.

POST https://localhost:44343/api/calculator/add HTTP/1.1

```
Host: localhost:44343
Content-Type: application/json
Content-type: Application/json
Content-Length: 14
```

Request Body

```
{"A":27,"B":4}
```

[QuickExec] ALT+Q > type HELP to learn more

CustomRules.js was loaded at: Sun Sep 16 05:56:44 UTC+1 2018

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

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..

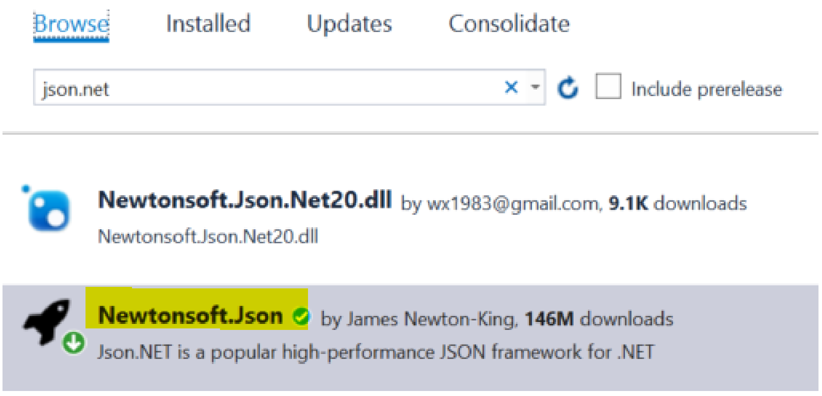
And write down your arguments.

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

Assignment 10: Program for consuming services: RestCalculatorConsumer

Create a simple Console 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 Newtosoft.json.



Newtonsoft.Json.Net20.dll by wx1983@gmail.com, **9.1K** downloads
Newtonsoft.Json.Net20.dll

Newtonsoft.Json by James Newton-King, **146M** downloads
Json.NET is a popular high-performance JSON framework for .NET

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 the URI pointing to your service and method (api/calculator).

a. In *main* show how to use the method and print out the sum of two integers.
   Execute the program

b. Carefully explain the code line-by-line what goes on.

c. 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.

d. 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?!