**COMPUTING SUBJECT:** Restful WCF-services

**TYPE:** Assignment

**IDENTIFICATION:** RestWCFCustomerService

**COPYRIGHT:** *Michael Claudius*

**LEVEL:** Medium

**TIME CONSUMPTION:** 2-4 hours

**EXTENT:** 50 lines

**OBJECTIVE:** Restful services based on WCF

**PRECONDITIONS:** WCF-services

**COMMANDS:**

**IDENTIFICATION:** RestWCFCustomer /MC

Mission

You are to make and use a restful services based on the WCF service by setting up a server and a client using the services provided. RESTful service provider, test it and finally publish it in Azure. The services support the classic GET, POST, PUT and DELETE requests. This we shall do in eight steps:

1. Create a project with a service
2. Configure Web.config to make use of restful WCF-services, tedious
3. Change the operation contract for methods and test the services
4. Create a customer class
5. Create and provide customer oriented services based on JSON
6. Test the services using Browser Fiddler/Postman
7. Create a client consumer utilizing the services
8. Publish the service in Azure

This assignment holds all 8 steps

Purpose

The purpose of this assignment is to be able to provide and consume restful WCF services.

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>

Configuration of web.config file   
<http://stackoverflow.com/questions/17644392/configuring-wcf-rest-services-in-web-config>

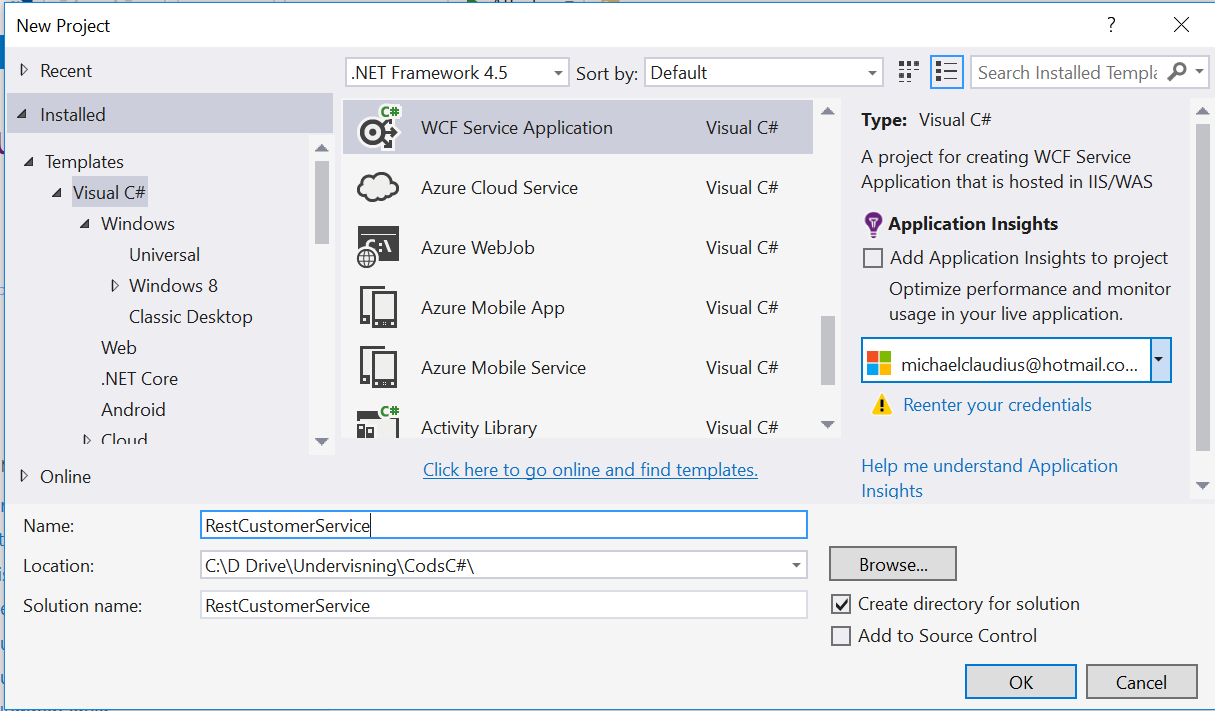
CRUD-Operations  
<http://www.topwcftutorials.net/2014/01/crud-wcf-restful-service.html>

Assignment 1: Restful WCF-service provider

You are to make a Rest Service provider.

Create a new WCF Service Application project, RestCustomerService.

Choose: WCF Service Application and click OK



Now you have to wait a while…

Notice the Service1.svc class and the interface IService1.svc  
Normally one would refactor and rename them to CustomerService and ICustomerService **BUT** it might cause trouble in Visual Studio 2017, so don’t.

Assignment 2: Web.config file

The Web.config file has to be manually adjusted to provide restful web-services based on WCF. This is tedious so be careful. More information is on:

<http://stackoverflow.com/questions/17644392/configuring-wcf-rest-services-in-web-config>

Open the Web.config file.

Inside the file, find the right location and declare the service, behavior and its end point.

Just after <system.serviceModel> declare and add the service:

<services>

<service name="RestCustomerService.Service1" behaviorConfiguration="ServiceBehavior">

<endpoint address="" binding="webHttpBinding" contract="RestCustomerService.IService1" behaviorConfiguration="webHttp"/>

</service>

</services>

Just after <serviceBehaviors> declare and add the service behavior:

<behavior name="ServiceBehavior">

<serviceMetadata httpGetEnabled="false"/>

<serviceDebug includeExceptionDetailInFaults="true"/>

</behavior>

Notice that the httpGetEnabled is set to false to disable the generation of a WSDL-file.

Just after </serviceBehaviors> and before </behaviors> add the endpoint behavior:

<endpointBehaviors>

<behavior name="webHttp">

<webHttp/>

</behavior>

</endpointBehaviors>

Finally change the asp-compability to false

<serviceHostingEnvironment aspNetCompatibilityEnabled="false"

So when the service is executed from a browser it will not try to lookup the wsdl-file.

*Check if the project can execute!?*

*Why not ? Read on….*

Assignment 3: Defining the operation contract

*The program does not work, because we forget to make changes in the IService1. The operation contracts must be defined as WebRole*

In IService1.cs change the operation contract for GetData to the following:

[OperationContract]

[WebInvoke(Method = "GET", ResponseFormat = WebMessageFormat.Json,   
UriTemplate = "data/")]

string GetData();

As the method now is a GET-method, in Service1.svc change the GetData method to the following

public string GetData()

{ return "Hej my name is rest";}

Also in Service1.svc out-comment the method GetDataUsingDataContract()

1. Execute the Service1 by viewing it in a local Browser. This will start the Azure emulator.

From the browser call the customers:

http://localhost:49972/Service1.svc/data

*All fine?!.*

Assignment 4: Model Customer class Customer

To the project add a class, Customer, with the data fields:

ID, FirstName, LastName, Year

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

Create the constructors:

Customer(int id, string first, string last, int year)

Intializing all the data fields

Customer() {} //empty constructor needed for JSON transfer.

Assignment 5: GET service GetCustomers

In *Service1* declare a static list holding two customers:

private static List<Customer> cList = new List<Customer>() //how to add customers ??

and a method returning a list of all customers:

public IList<Customer> GetCustomers()

Then in *IService1* define the operation contract for this method

[OperationContract]

[WebInvoke(Method = "GET", ResponseFormat = WebMessageFormat.Json,

UriTemplate = "customers/")]

IList<Customer> GetCustomers();

1. Execute the Service1 by viewing it in a local Browser. This will start the Azure emulator.

From the browser call the customers:

http://localhost:49972/Service1.svc/customers

1. Try to invoke the method from Fiddler/Postman

Assignment 6: Consumer RestCustomerClient

Create a simple Console Application project, with the Customer class, and a special method:

private static async Task<IList<Customer>> GetCustomersAsync()

{

using (HttpClient client = new HttpClient())

{ string content = await client.GetStringAsync(CustomersUri);

IList<Customer> cList = JsonConvert.DeserializeObject<IList<Customer>>(content);

return cList;

} }

where the CustomerUri is the URI pointing to your service and method (/customers)

In main show how to find and print out:

* All customers

Assignment 7: More methods in service and used by client

You must now extend the service with more methods

* String GetCustomer(int id)  
  Return the customer information with the specified id.
* DeleteCustomer(int id)   
  Delete the customer with the specified id.
* InsertCustomer(Customer c, int id)  
  Insert the customer object in the list
* UpdateCustomer(Customer c, int id)  
  Retrieves a specified customer, replace the old customer object in the list a new customer object, c.

For each method show how to use it from Fiddler/Postman and in the consumer RestCustomerClient.

Remember for each method you must carefully think about

* Which HTTP method/verb to use?
* What should the URI (UriTemplate) look like? Any parameters to the URI, like {id}?
* What is the request format (RequestFormat)? JSON or XML, or not necessary.
* What is the response format (ResponseFormat)? JSON or XML, or not necessary.

Assignment 8: Publish in Azure

1. Publish your service in Microsoft Azure.
2. Use a browser to show the API and the methods.
3. Use Postman or Fiddler to show requests and responses.
4. Show how to use the Azure service instead of the local URI in your consumer program.