**COMPUTING SUBJECT:** Restful WCF-services

**TYPE:** Assignment

**IDENTIFICATION:** RestCustomer

**COPYRIGHT:** *Michael Claudius/small modification Peter Levinsky*

**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 publish it in Azure. The services support the classic GET, POST, PUT and DELETE requests. This we shall do in four steps:

1. Create a project with service and a class Customer
2. Configure web.config to make use of restful WCF-services
3. Create and provide services based on JSON
4. Test the services using Unit test and Fiddler/Postmand)
5. Create a client consumer utilizing the services
6. Publish the service in Azure

This assignment holds all 6 steps

Purpose

The purpose of this assignment is to be able to provide and consumer 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>

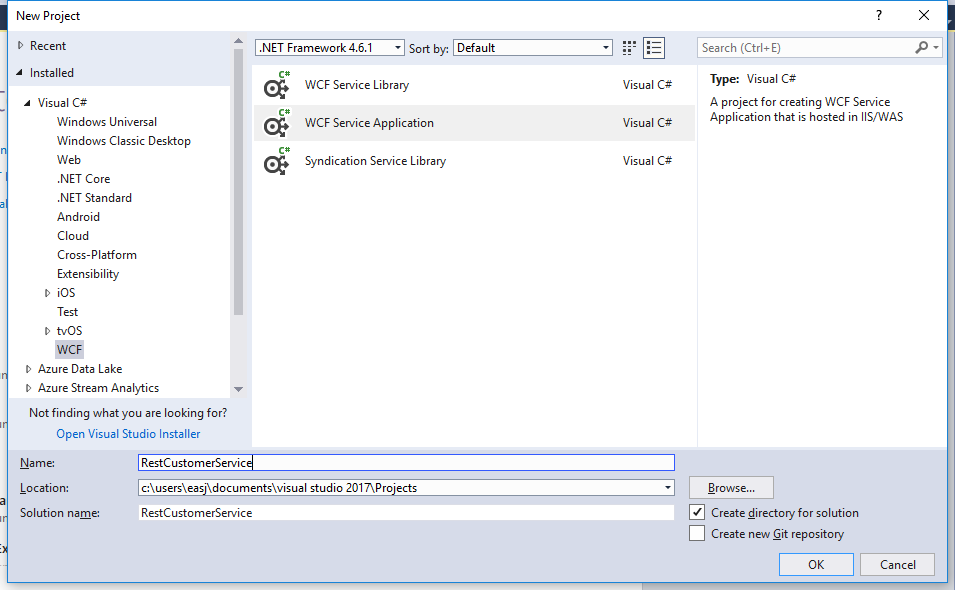
Configuration of web.config file   
<http://stackoverflow.com/questions/17644392/configuring-wcf-rest-services-in-web-config>

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

Assignment 1: Restful WCF-service provider

Create a new project, RestCustomerService,

Choose: WCF -> Web Service Application and click OK



Now you have to wait a while…

1. Rename the IService1 to *ICustomerService* and the Service1 to *CustomerService*.
2. Have the CustomerService (kode) tab active – try to run the application – what happen?
3. NOW Have the ICustomerService (interface) tab active – try to run the application – what happen now?
4. To the project add a model class, Customer, with the data fields:

ID, FirstName, LastName, Year

with get/set method for all the fields.

Create the constructors:

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

Intializing all the data fields

Customer() {} //empty constrsuctor – MUST BE THERE

Assignment 2: Web.config file

You have now a SOAP service. The **Web.config** file has to be adjusted manually to provide a **restful** wcf services.

For more information see:

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

Open the **Web.config** file.

You have to specify you will like to have this End-Point to be a RESTFUL service

Declare the service and its endpoint – after the ‘<system.serviceModel>’ insert

<services>

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

<endpoint address="" binding="webHttpBinding"

contract="RestCustomerService.ICustomerService" behaviorConfiguration="webHttp"/>

</service>

</services>

Declare and add the service behavior after the ‘<serviceBehaviors>’  
*(Note you could stick to the default behavior and refactor it)*:

<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.

Declare and add the endpoint behavior (after the **‘< / s**erviceBehaviors>’) to use HTTP:

<endpointBehaviors>

<behavior name="webHttp">

<webHttp/>

</behavior>

</endpointBehaviors>

Finally you can 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.

Assignment 3: Define the GET REST service GetCustomers

In *ICustomerService* define the operation contract for this method

[OperationContract]

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

UriTemplate = "customers/")]

IList<Customer> GetCustomers();

Then in *CustomerService* declare a static list holding two customers:

private static List<Customer> cList = new List<Customer>()

and a method returning a list of all customers:

public IList<Customer> GetCustomers()

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

From the browser call the customers: (Note: The port number could be different at your computer)

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

1. Try to invoke the method from Fiddler/Postman
2. Make a structured test of your system i.e. create a unit test of your CustomerService-class.  
   Then you know the functionality at server site is ‘correct’.

Assignment 4: 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 5: 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.

Refactor your Unit Test to cover these new methods.

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 6: 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.