Creating and Consuming REST Services in .NET

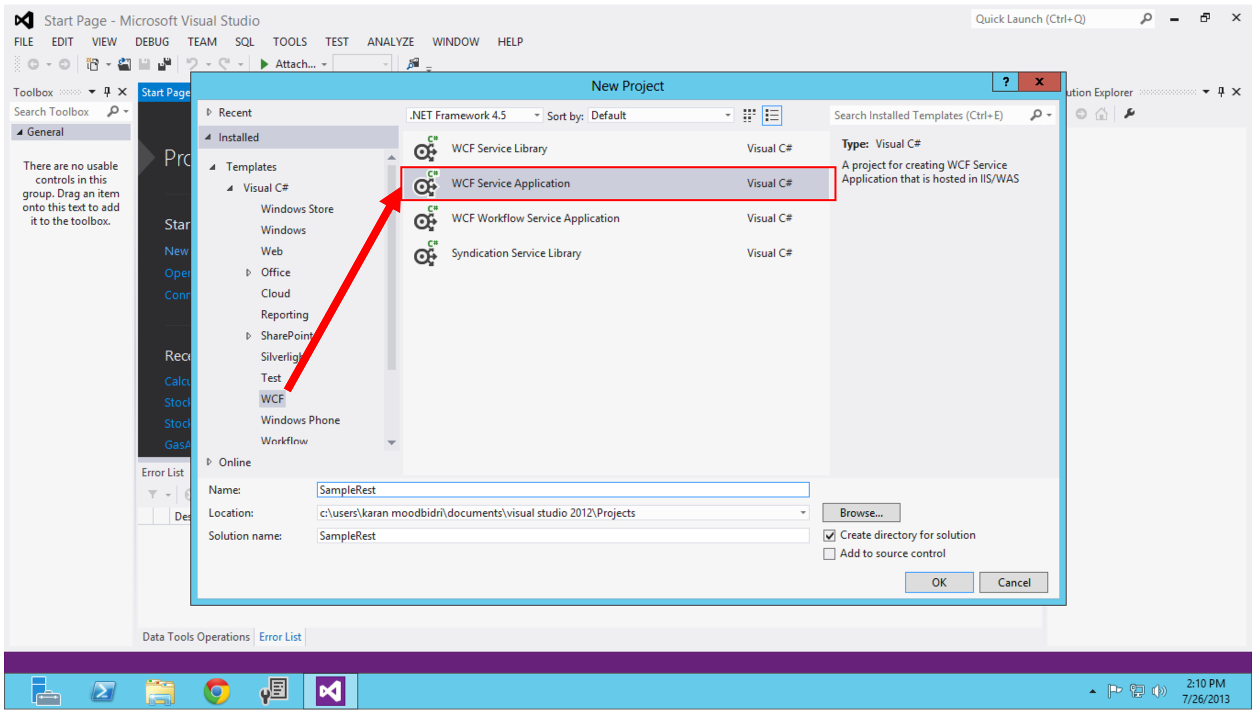
Cheat Sheet to create REST Services in .NET

This document will provide a user the capability to create a sample REST service in C# using the following technologies
1> >.NET Framework and Visual Studio 2010 and above
2> Configured IIS

Karan Moodbidri and Amey Gawde
7/26/2013

Creating REST Service Project in Visual Studio

➢ Create a New Project and Click on WCF and then select WCF Service Application



.NET Framework 4.5 — Sort by: Default

Type: Visual C#

A project for creating WCF Service Application that is hosted in IIS/WAS

Name: SampleRest

Location: c:\users\karan moodbidri\documents\visual studio 2012\Projects

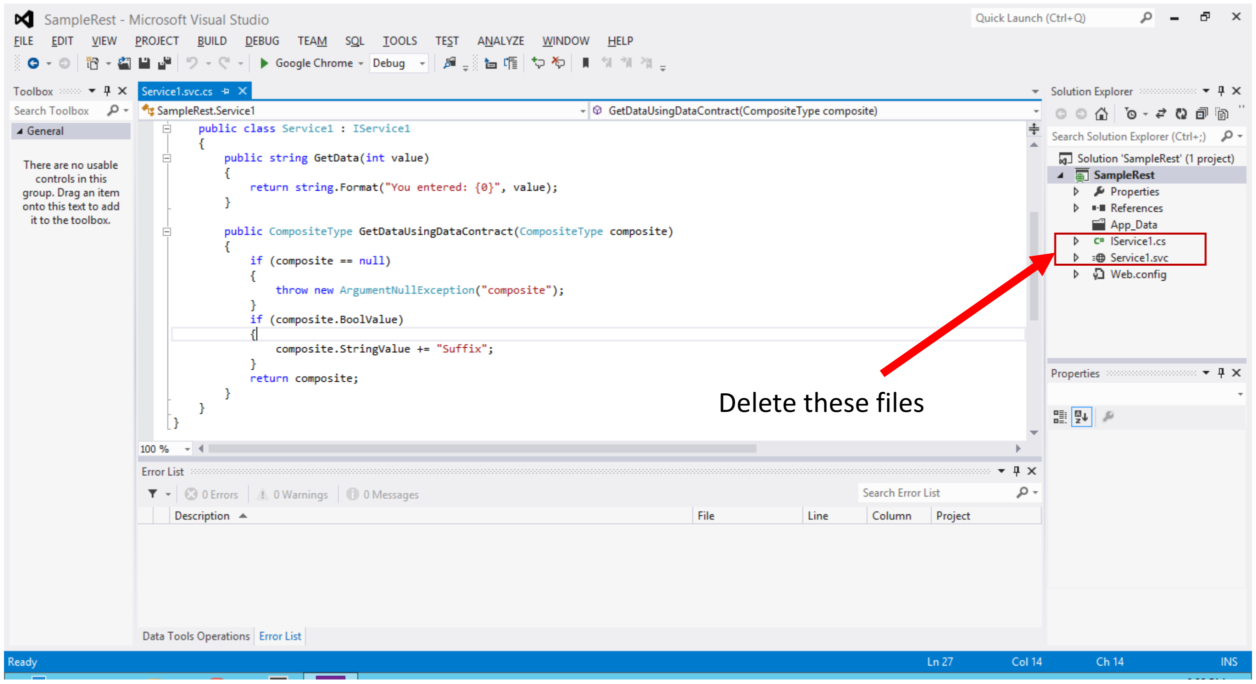
Solution name: SampleRest

☑ Create directory for solution

☐ Add to source control

➢ The Project Structure Will Open

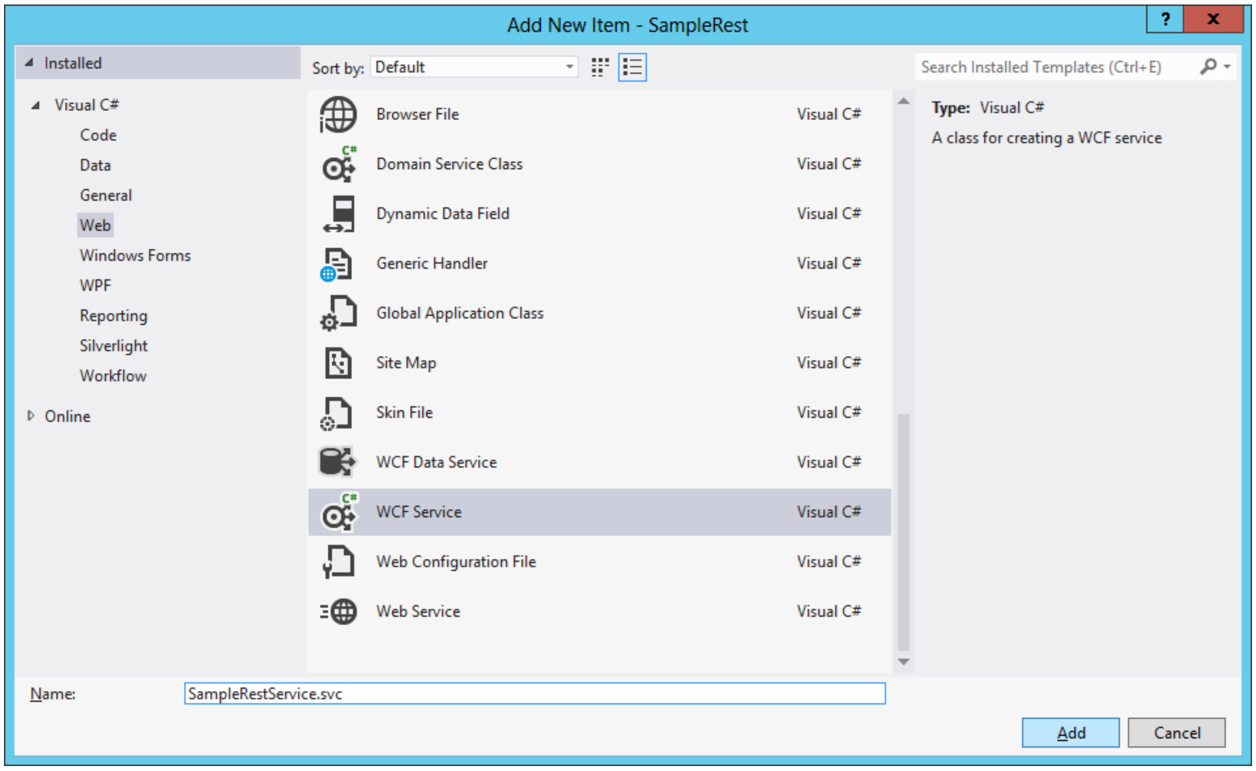
➢ Delete Service1.svc and IService1.cs



```
public class Service1 : IService1
{
    public string GetData(int value)
    {
        return string.Format("You entered: {0}", value);
    }

    public CompositeType GetDataUsingDataContract(CompositeType composite)
    {
        if (composite == null)
        {
            throw new ArgumentNullException("composite");
        }
        if (composite.BoolValue)
        {
            composite.StringValue += "Suffix";
        }
        return composite;
    }
}
```

- ➢ Click on project structure
- ➢ Click Add
- ➢ Click on New Item
- ➢ Select WCF Service



Browser File	Visual C#
Domain Service Class	Visual C#
Dynamic Data Field	Visual C#
Generic Handler	Visual C#
Global Application Class	Visual C#
Site Map	Visual C#
Skin File	Visual C#
WCF Data Service	Visual C#
WCF Service	Visual C#
Web Configuration File	Visual C#
Web Service	Visual C#

Type: Visual C#

A class for creating a WCF service

Name: SampleRestService.svc

- ➢ Click Add
- ➢ Now we Added the WCF Service SampleRestService to the project.
- ➢ In the ISampleRestService.cs Add the this code

```
[OperationContract]
[System.ServiceModel.Web.WebInvoke(Method = "GET",
ResponseFormat = WebMessageFormat.Json, BodyStyle = WebMessageBodyStyle.Wrapped,
UriTemplate = "json/?name={name}")]

string sayHelloJSON(string name);
```

UriTemplate should be the Uri structure of how you want to call the service. For example, In the above example we wanted to call the function to be called as json/?name = {name} where name is the query parameter passed by the function sayHelloJSON().Return type of this function would be JSON.

- Similarly for XML Return type. Add the code Below

```
 [OperationContract]
 [System.ServiceModel.Web.WebInvoke(Method = "GET",
 ResponseFormat = WebMessageFormat.Xml, BodyStyle = WebMessageBodyStyle.Wrapped,
UriTemplate = "/xml/?name={name}")]
 string sayHelloXML(string name);
```

- In the SampleRestService.svc provide functionality for the above given functions for example:

```
public string sayHelloJSON(string name)}{

       return "Hello "+name;

}

public string sayHelloXML(string name)}{

       return "Hello "+name;

}
```

- Now Open the Web.config File
- Add The following code below <system.serviceModel>

```
<services>
      <service name="SampleRest.SampleRestService"
behaviorConfiguration="ServiceBehaviour">
        <!-- Service Endpoints -->
        <!-- Unless fully qualified, address is relative to base address supplied above -
->
        <endpoint address ="" binding="webHttpBinding" contract="
SampleRest.SampleRestService " behaviorConfiguration="web">
          <!--
              Upon deployment, the following identity element should be removed or
replaced to reflect the
              identity under which the deployed service runs.  If removed, WCF will infer
an appropriate identity
              automatically.
          -->
        </endpoint>
      </service>
    </services>
```

- Add the following code also in Web.config file under the tag <behavior>.Remove previously written code under tag <Behavior> :

```
<behaviors>
      <serviceBehaviors>
        <behavior name="ServiceBehaviour">
          <!-- To avoid disclosing metadata information, set the value below to false and
remove the metadata endpoint above before deployment -->
          <serviceMetadata httpGetEnabled="true"/>
          <!-- To receive exception details in faults for debugging purposes, set the
value below to true.  Set to false before deployment to avoid disclosing exception
information -->
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
      <endpointBehaviors>
        <behavior name="web">
          <webHttp/>
        </behavior>
      </endpointBehaviors>
    </behaviors>
```

- Add the following code below Behavior.

```
    <serviceHostingEnvironment aspNetCompatibilityEnabled="true"
multipleSiteBindingsEnabled="true" />
```

- Rest Service is completed. Host it on IIS. Perform an HTTP Get operation on the URL you created above as the URI Template.

<u>Consuming a REST service in C# without using WebClient</u>

- Click on File
- Click on New
- Select Console Application under Visual C#
- Now I would help you go through the code for the Consuming the Service
- We create a WebRequest in C# and provide the URL to the REST service and the Method to connect to the URL for example : perform a 'PUT ' or a 'GET' Request

Example Code:

```
WebRequest restRequest = 
WebRequest.Create(@"http://vhost3.cs.rit.edu/CalculatorRest/RestServiceImpl.svc/Add/xml/?
n1=10&n2=20");

            restRequest.Method = "GET";
```

- Now we create a HttpWebResponse to get the response that was provided from the REST service

Example Code:

```
  HttpWebResponse restResponse = restRequest.GetResponse() as HttpWebResponse;
```

- Now we check if the status of the HTTPRequest performed was completed or not. For that we create a condition loop i.ie an if condition and chech if the status code was an ok.

Example code :

```
if (restResponse.StatusCode == HttpStatusCode.OK)
            {}
```

- If the status code was an OK then we can go ahead and read from the response stream and print it on the console.

Example Code :

```
  using (Stream restResponseStream = restResponse.GetResponseStream())
                {
                    StreamReader streamReader = new StreamReader(restResponseStream, 
Encoding.UTF8);
                    Console.WriteLine(streamReader.ReadToEnd());
                }
```

- If there are any errors then we will print the error on the console.

Example Code:

```
   Console.WriteLine(string.Format("Status Code: {0}, Status Description: {1}", 
restResponse.StatusCode, restResponse.StatusDescription));
```

- ➢ We can succesfully Consumed a Service in C#.

The whole code is as below

```
  WebRequest restRequest = 
WebRequest.Create(@"http://vhost3.cs.rit.edu/CalculatorRest/RestServiceImpl.svc/Add/xml/?
n1=10&n2=20");

            restRequest.Method = "GET";

            HttpWebResponse restResponse = restRequest.GetResponse() as HttpWebResponse;
            if (restResponse.StatusCode == HttpStatusCode.OK)
            {
                using (Stream restResponseStream = restResponse.GetResponseStream())
                {
                    StreamReader streamReader = new StreamReader(restResponseStream, 
Encoding.UTF8);
                    Console.WriteLine(streamReader.ReadToEnd());
                }
            }
            else
            {
                Console.WriteLine(string.Format("Status Code: {0}, Status Description: 
{1}", restResponse.StatusCode, restResponse.StatusDescription));
            }
```