CO7215/CO7515

Advanced Web Technologies

Lab 3

Dr Stephan Reiff-Marganiec
Department of Computer Science

Autumn 2015

University of Leicester

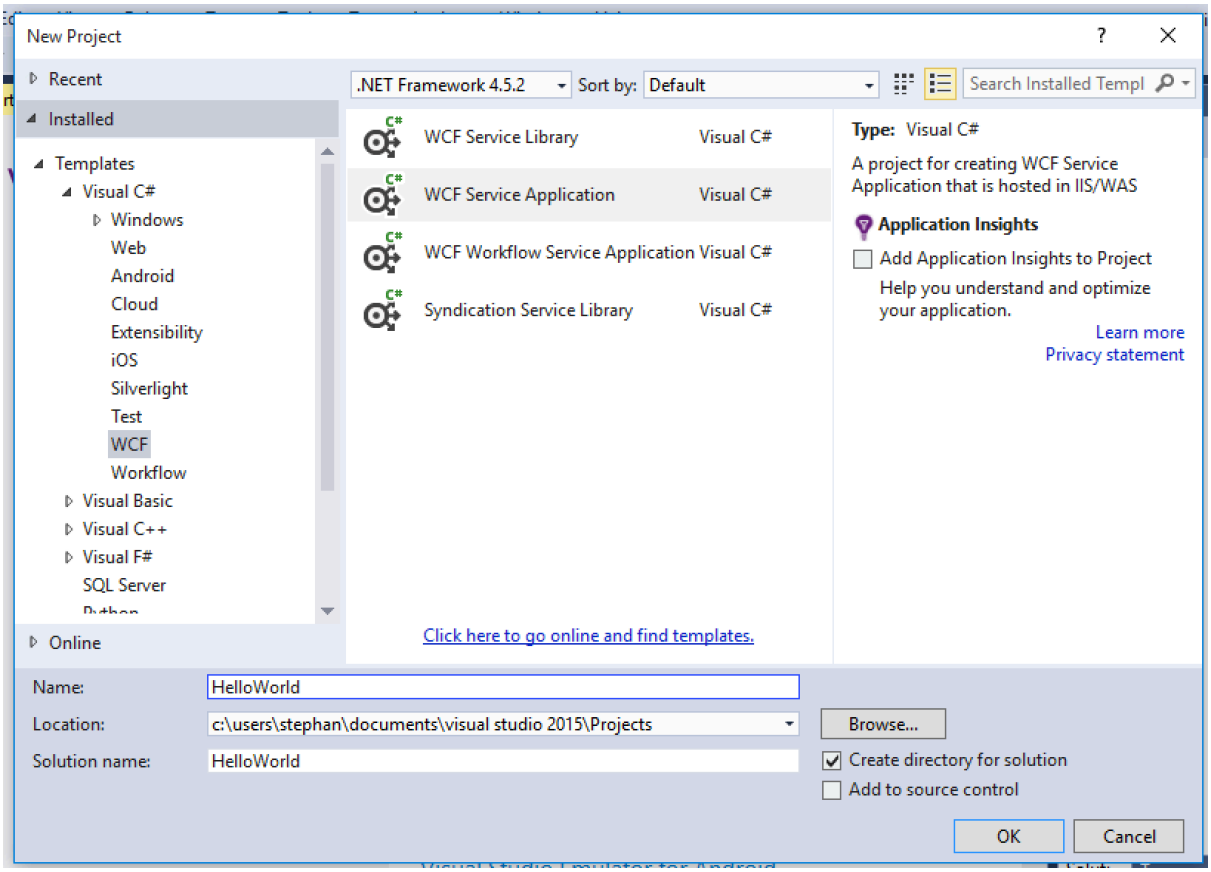
Lab 3: Web Service Introduction, Implementation and Invocation

After this lab, you should

- understand what a Web service is
- be able to implement the HelloWorld Web service
- be able to implement multiple operations in one Web service
- be able to implement a Web service with database backend
- be able to invoke Web Service
- be able to invoke multiple Web services
- be able to implement complex applications by using Web services

A HelloWorld Web Service

In the past web services were being build using ASP, but now the recommended route for building web services using Microsoft's technology stack is to build on WCF (Windows Communication Foundation). This does in fact have several advantages as you can more easily change services to also interact using JSON and is conceptually cleaner as it uses the ideas of 'Design by Contract'. To get started go to File -> New -> Project and select Visual C# and you will see the dialog box shown below.



©Dr S Reiff-Marganiec, University of Leicester

Select WCF Service Application template and rename the project name and solution name to HelloWorld. Finally click OK for continuing. Visual Studio will take a moment to create your solution, and will eventually automatically open a file called IService1.cs – this is the service interface and you can see that some sample method interfaces are shown (one using simple data and one using complex data). Open Service1.svc.cs in addition (from the solutions explorer) and you can see the implementation of the methods for which we have just looked at the interfaces.

> **Q:** Ask yourself where you need to make changes when you want to introduce a new method. What would you change?

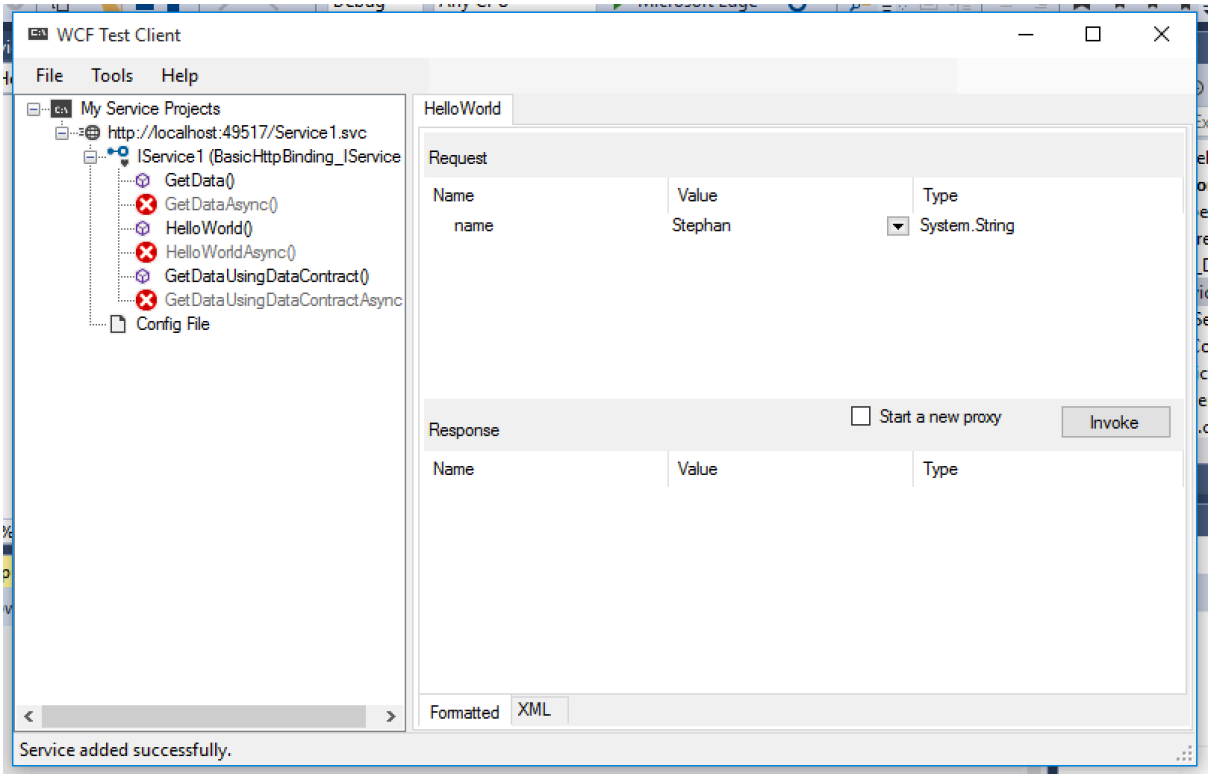
As we are wishing to write a 'hello world' service, we will need to make some changes. You have a choice of either editing the existing GetData method or adding a new method – add an interface to the Interface code and then an implementation to the other class. The hello world method should have the following interface:

```
public string HelloWorld(string name);
```

You should be able to create the implementation yourself without too much trouble.

Once done, build your solution (Ctrl+Shift+B or the respective menu item) and then you should be able to launch the service (F5 or one of the run options with or without debugging).

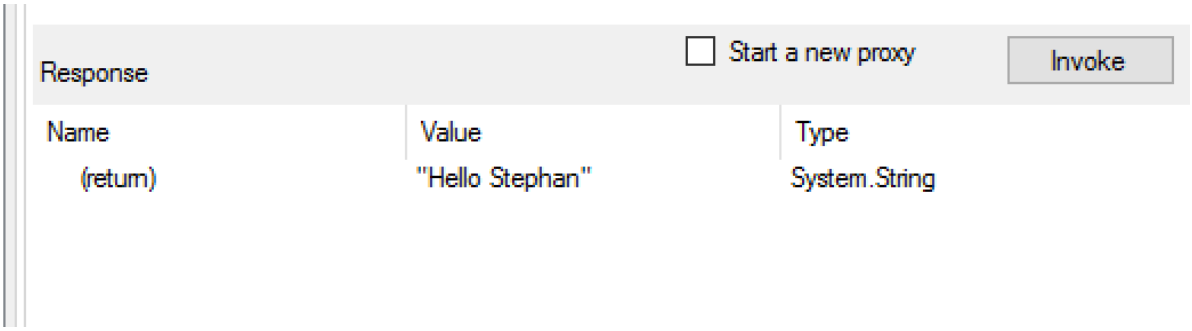
After a short while a screen like the following will appear:



©Dr S Reiff-Marganiec, University of Leicester
Autumn 2015

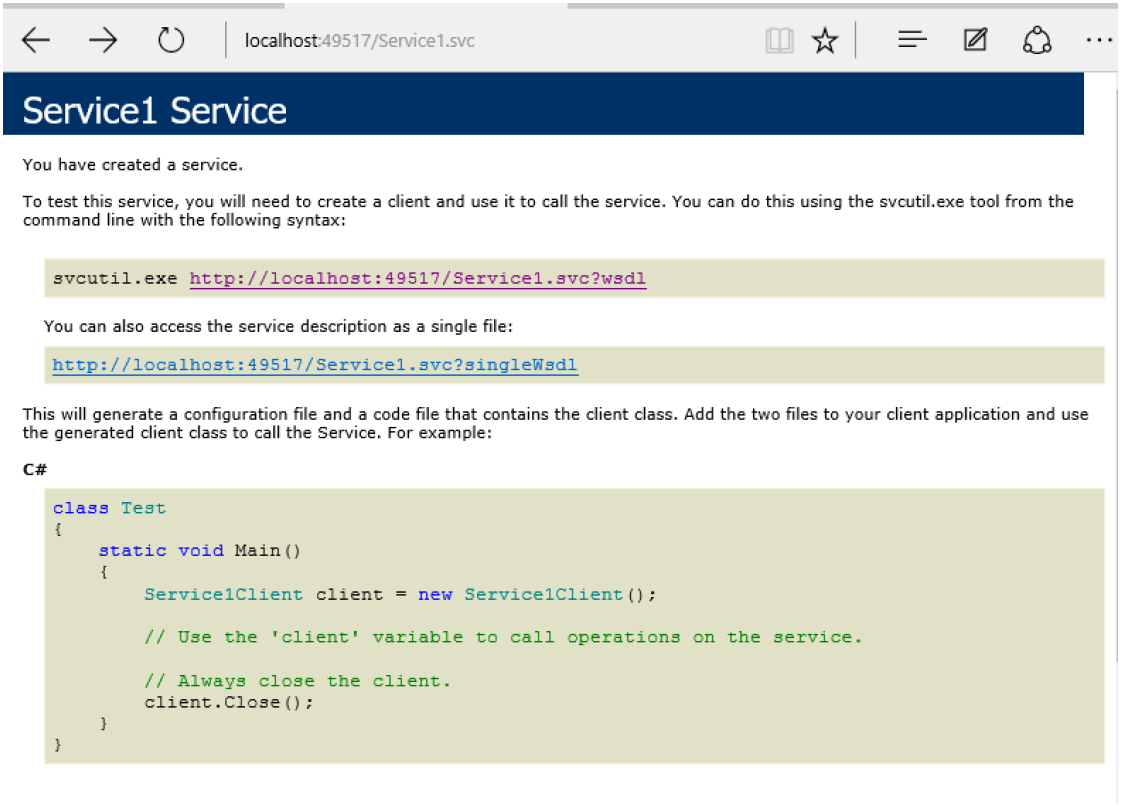
This is a convenient test tool build into VS to test services without having a formal client application. In the background several things happened when starting the service, most notably a web server for running services was started up and the service deployed: the give away is the fact that under the "My Service Projects " you can see a URL to a localhost address and port. We will come back to this in a moment, but first let us see whether the service works: Select the HelloWorld method in the test client and right pane will look like it does above: with an input and output area.

Add a value for your input variable and select invoke – the result will be shown in the results part and should look a bit like this:



All fine? Then let's move on.

Copy the URL to the service endpoint, open a web browser and point to the URL:



©Dr S Reiff-Marganiec, University of Leicester

Select the first link in the page (the one behind svcutil.exe) and you will see the WSDL file of the service. The WSDL file specifies the end point for invoking the service and its description – as you know from the lectures. Have a look around the file and you should find your service as an operation name.

That's it: your first service written and running – I promised that it will be easy!

Build a Calculator Web Service

Now that you have learned how to write a basic web service you should try to build one yourself. The Calculator method takes two double numbers and operators (e.g. +, -) as input values and returns the calculation result as output value.

The main purpose of this exercise is to enable you understand that one Web service may have more than one operator and each operator is a different method or functionality of the service. You are welcome to play around with this exercise and try out alternatives. You might also like to look at the exercise from Lab 1 and see if you can embed some of that functionality into the webservice.

Implement an EmployeeStore Web Service

Now let us implement a Web service which supports functions of query, update and insert to the Employ database table from Lab 2. For connecting to the database and using the database as well as the functionality of the EmployeeStore look at the previous Lab materials if required. For this service you might want to explore the use of the Interface and implementation of the service that uses the data objects (see the sample in the auto generated code!).

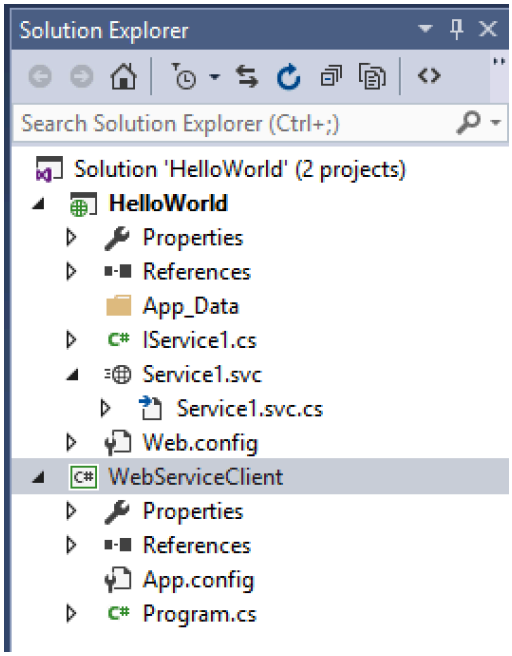
Invoking a Web Service from a Client Application

We will use the calculator service for this exercise, but you can also try it with the Employee store service. First of all, to enable other applications to invoke a service, the service must be available first. So start the Calculator Web Service project that we built before and run it (make sure you start it without debugging).

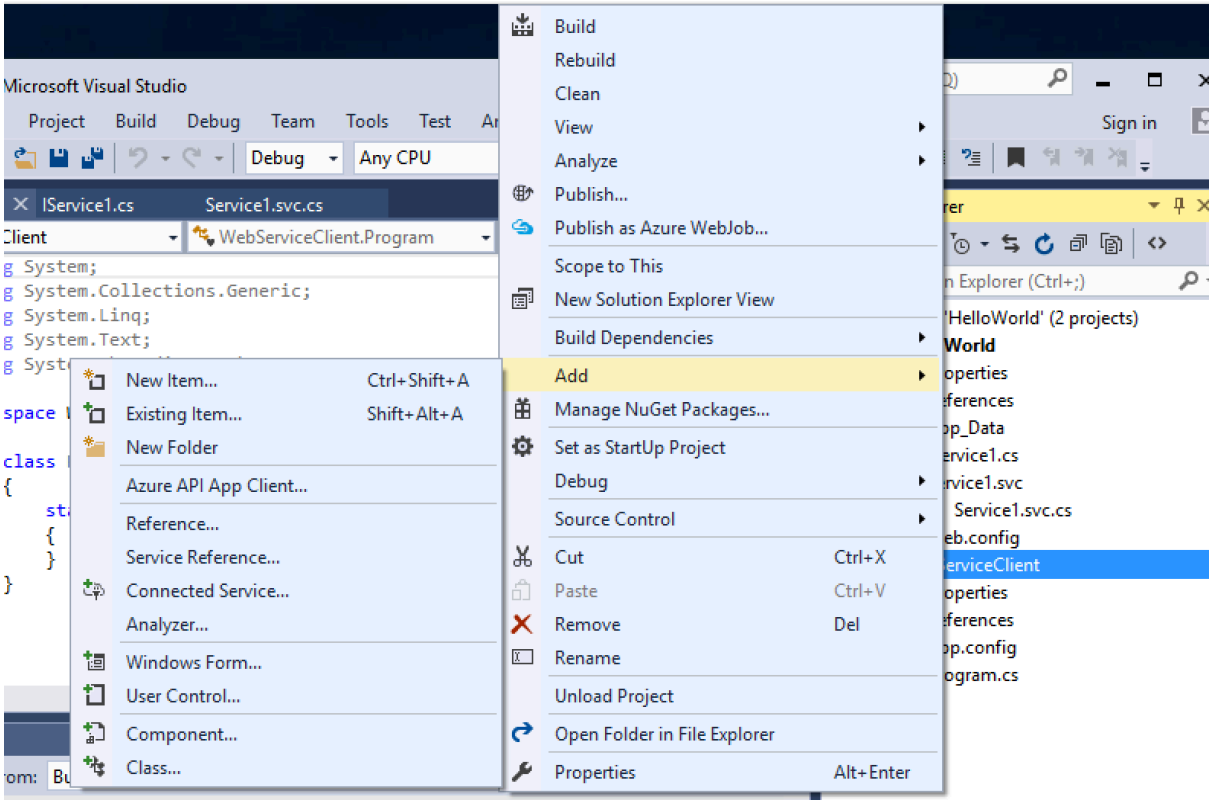
Next, add a new project to your current solution (please do not start a new solution, that will stop the web service tat is running as it will close the current solution). Go to File->New Project and select the Console Application from the C# menu. The in the options change the Solution setting from

©Dr S Reiff-Marganiec, University of Leicester
Autumn 2015

"Create new Solution" to "Add to Solution". Your solution explorer now should look a bit like this (I called my Console Application *WebServiceClient*):

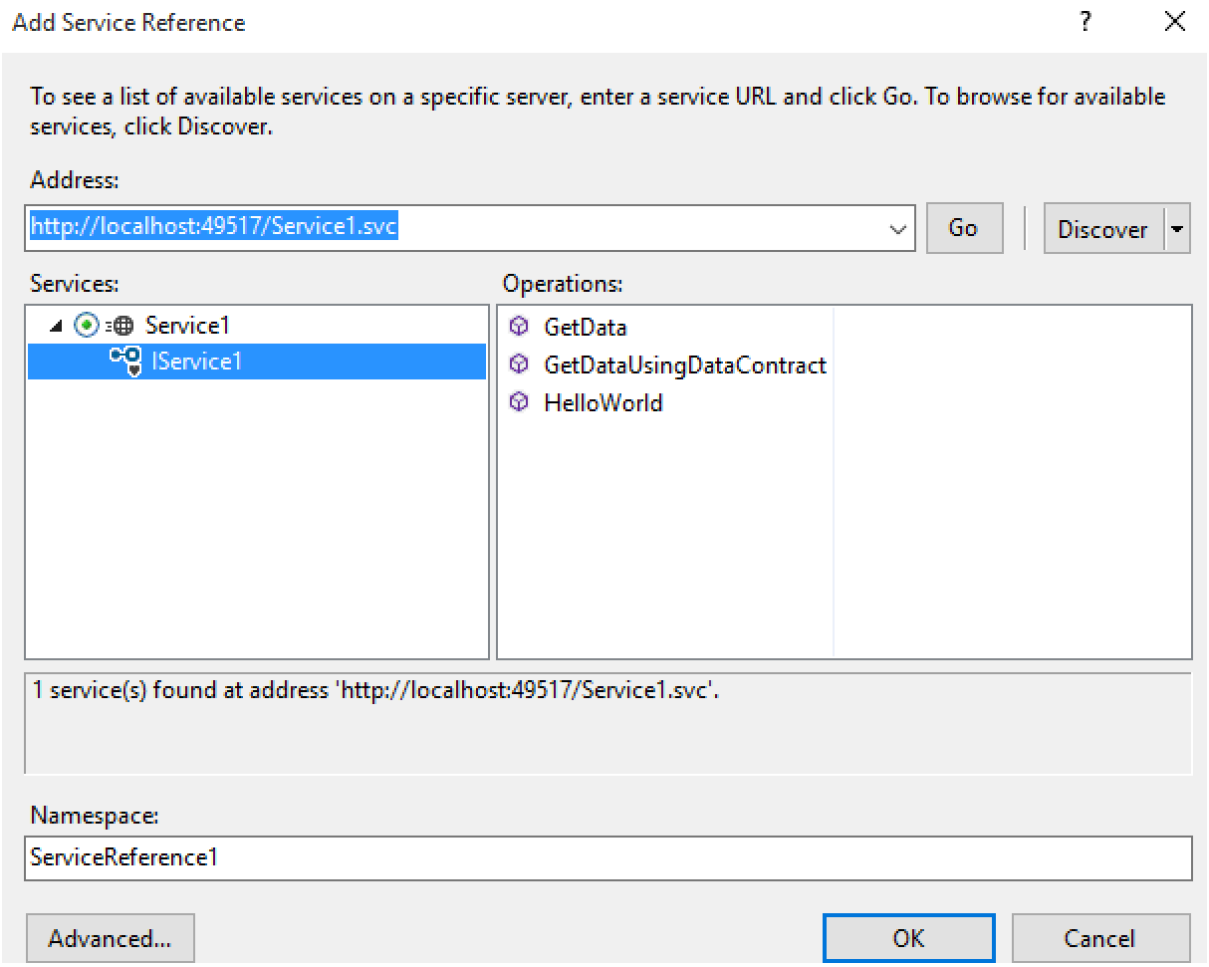


Next, Select the Project and right click; this will open a menu as follows and selecting 'Add' will in a second menu give you the option to add a service reference. Select this.



©Dr S Reiff-Marganiec, University of Leicester
Autumn 2015

Add the URL for your service under Address and click ‘Go’. Your service should be discovered and the dialogue should become something like this (the screenshot is for my HelloWorldService):



You should now click OK. You are back to the code editor and you should locate the Program.cs file in your console application.

Add the following code to the main method:

Add the URL for your service under Address and click ‘Go’. Your service should be discovered and the dialogue should become something like this (the screenshot is for my HelloWorldService):

```
ServiceReference1.Service1Client client = new
      ServiceReference1.Service1Client();
```

Note the exact names might differ depending on what you called things. Essentially it needs to match the ServiceReference and Service name that you created in the previous step.

©Dr S Reiff-Marganiec, University of Leicester
Autumn 2015

The “client” object instance that you have just created can be used to invoke the service methods – and you will see that they look just like any other method in the code. So, create a variable of a suitable type, call the service method with the right input parameters and display the result on screen.

[For the HelloWorld Service this would be e.g.

```
string s = “”;
s=client.HelloWorld(“Stephan”);
Console.WriteLine(s);
Console.ReadLine();
```

]

To run this from one solution, build the solution, then run the service not in debug mode and finally right click the web service client project and in the popup menu select Debug – Start new Instance. You should see the terminal application open and show you the string.

Invoking multiple services

You now have all the tools to build webservices with one or more methods and also to build clients – if a client needs access to more than one service, just add more service references. Give it a go. For this we are providing a service to convert strings to upper case and to reverse them as services at:

https://campus.cs.le.ac.uk/tyche/ws-temp/ReverseOperator.wsdl
https://campus.cs.le.ac.uk/tyche/ws-temp/UpperOperator.wsdl

Play around using with using these in a project. Incidentially, these services are written in Java, but this makes no difference when you use them – however there is a difference in that these are SOAP services so we need to invoke them slightly different to the above (you will also point the Add Service reference to the wsdl file, not the service instance):

You need to include a “using” phrase to “WebServiceClient.XXX” where XXX is the name of your service reference (ServiceReference2 in the code below).

To invoke you will need a new client object, say client 2. For reverse you do the following:

```
ReverseOperatorPortTypeClient c2 =
    new ServiceReference2.ReverseOperatorPortTypeClient(
        “ReverseOperatorSOAP12port_http”);
```

You can then use c2.reverse() as you expect (it takes a string argument and returns a string).
Two notes on the above: To find out what Clients are offered by a SOAP Service reference type the name of the Servicereference followed by a fullstop and see what is offered as code completion. You will also need to pass a parameter to the constructor to identify the port or endpoint used in case that the service offers multiple (most SOAP services do). Compare my code in the last box with the WSDL file content.

©Dr S Reiff-Marganiec, University of Leicester