■ ECE/CS 4984: Wireless Networking and Mobile Systems ■

At-home Exercise 4 (E4)

Part I – Objectives and Lab Materials

Objective:

The objectives of this lab are to:

- Introduce designing a web service server
- Introduce using full C#

After completing the assignment, you should be able to:

- Design and implement web service clients and servers.
- Understand the role of middleware (XML and SOAP in this context) and how they enhance communication.

Hardware to be used in this lab assignment:

- Notebook, iPAQ, 802.11b access point, 802.11b card

Software to be used in this lab assignment:

- Visual Studio .NET 2003, .NETcf

Overview:

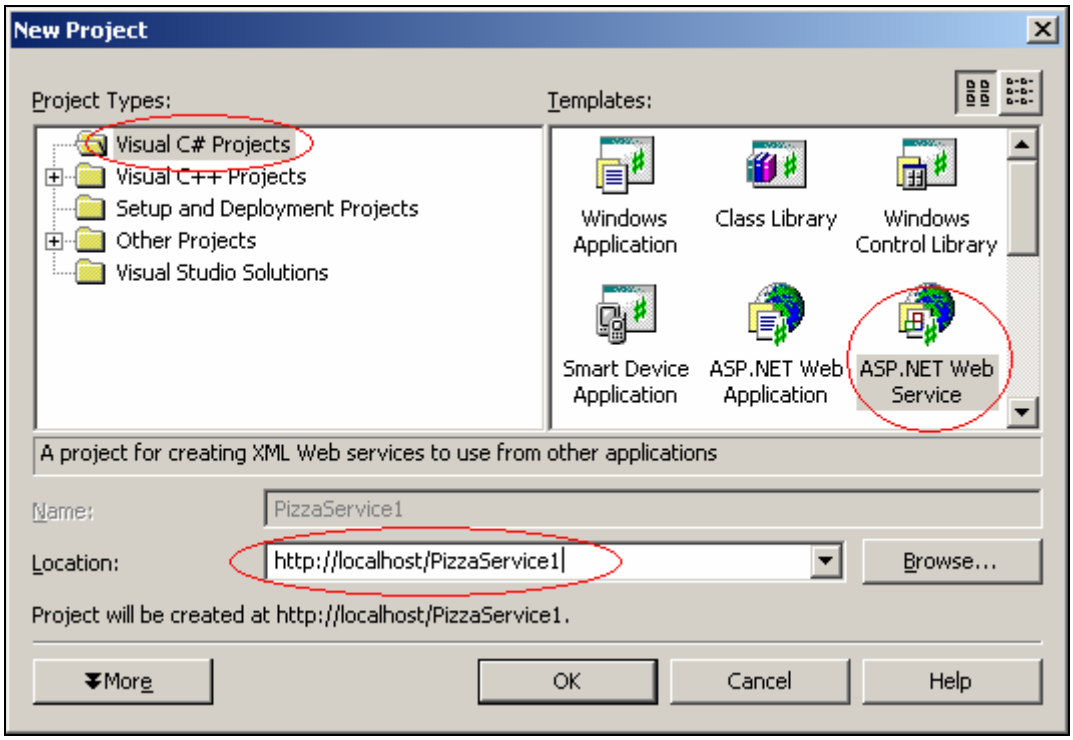
In this section of the lab you will be designing a web service server application. In this context, a web service is an application running on the notebook that offers some sort of 'service' to the client; the iPAQ in this case. Using the pizza client you designed in the earlier part of the lab, you will write the service back-end of the client/server pair.

Part II – At-home lab assignment

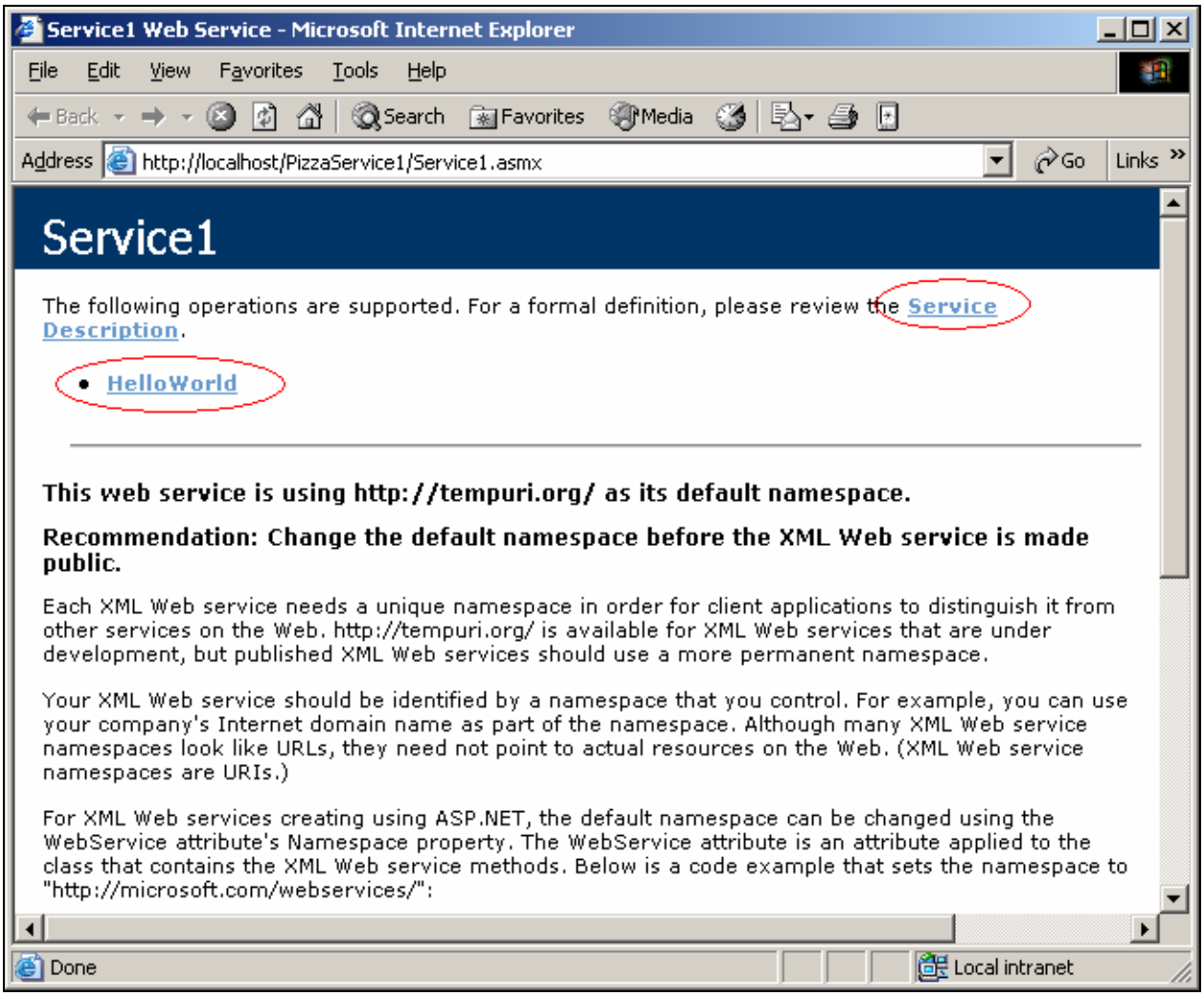
Part A: Creating a Web Service

The first step is creating a new web service using VS.NET.

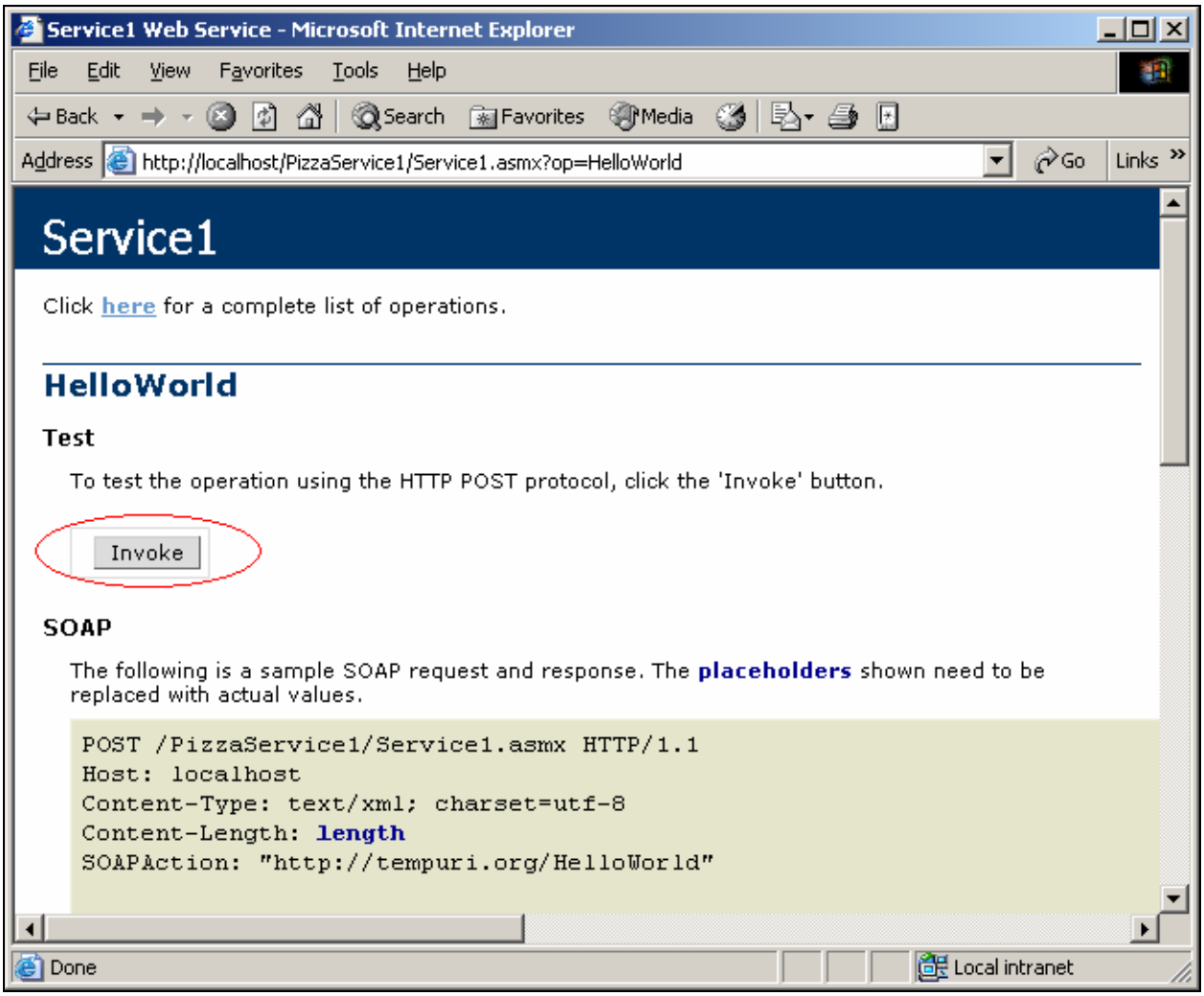
1. Start a new project in VS using a Visual C# Project template of ASP.NET Web Service (as shown below). Name the project something meaningful.



2. Find the web service. Find the directory that contains this project; it is typically in the 'Visual Studio' directory in 'My Documents'. Inside the project directory there should be 2 files. This should be questionable, since after all, VS projects typically have a lot more than 2 files. The reason for this is that the rest of the project is in the web server directory tree. Open the directory 'C:\Inetpub\wwwroot'. In here you will find a directory named the same as your project. Opening that directory will reveal something that looks more like a VS project. This step has nothing to do with the creation of the project, it just provides some possibly useful information.
3. (Back to VS). Click the 'click here to switch to code view' link in the middle of the page. This is the main file of the project. Scroll down to the bottom and you will see a commented section beginning with '[WebMethod]'. The [WebMethod] line specifies that this method will be a web service method. Only methods preceded by [WebMethod] will be available to the world. The example web method provided is, of course, a hello world method. Uncomment the 5 lines of the method and build the project. Run the project.
4. A simple test.
    a. When you run the project, a web browser will be opened and point you to a web page. (Note the URL of this page. You might need to look back at it.) This page is generated by the local web server (Internet Information Server, IIS). This page gives a link to the XML service description; feel free to take a look at this. The next section shows the available web methods. Every web method you defined in this project will be here. For now, we only have 'HelloWorld'. Click the link for HelloWorld.



b. This leads to a new page. This page has an invoke button and some definitions concerning the use of the method. If there were any parameters to be passed to the function, there would be space to input those parameters. Click the 'Invoke' button.



c. The returned XML document is the response that a client would receive if they called the web method, 'HelloWorld', with the parameters we specified, in this case, none.

d. This is a way of testing your web methods to see if there is an issue with the web method or the calling client method. Unfortunately, you can not specify object attributes.

e. You may close the browser now.

Part B: How does it work?

For web services to work, the web server must know what to do with the connection it is receiving. For our setup, using IIS, IIS knows how to interpret the .asmx file extension type. ASP.NET (Active Server Pages.NET) is used when dealing with .asmx files. When a file with the .asmx extension is requested, IIS knows what to do with the data being sent to it. More about this will be seen in the latter part of this lab.

Part C: Your Task.

For this section of the lab you will design the back-end that was provided for the in-class portion of the lab. It should match the method definition presented in the earlier section. It should return the same

values as well. It **does not** need to store the values of the orders place, simply return the string with a summary of the information provided by the client. The return string has the following structure

<clientID> <string for size> <string for crust> <topping1> <topping2>…

Each identifier is followed by **a single space**.

For testing, set up the iPAQ and notebook to communicate. You can use either ad hoc mode or involve the access point. Either way works. Assign each an address from the private space (192.168.*x.y*) and an appropriate network mask. Test the connectivity by pinging the iPAQ from the notebook.

Test your server with the client you developed in the earlier section of this lab. You will have to modify the web reference to point to your notebook's server instead of the GTAs notebook's service. This will involve changing the URL to point to the new service and rebuilding the client project. (See step 4a for an easy way of finding the service URL.)

You will also need to define the *pizzaOrder* object in your **service** code. It does not need to have any bracketed [ ] definitions added; just simply copy and paste it in.

Part III: The Report

(Note: if needed do some googling to answer the questions)

Part I: In-class report

1. Using the trace you made with ethereal:
    a. Neatly format the trace file. For the XML contents tab-ify the structure so it is easy to follow the beginning and end of the different sections.
    b. Divide the session into sessions and identify each session on whether it is data from the client to the server or the server to the client.
    c. For the SOAP contents, identify the values being passed **into** and **returned from** the web method call.
    d. Search the session for instances of 'http://tempuri.org', why is this link present and where does it point? (semi-trick question)

Part II: At-home report

1. Include your code for the *placeOrder* web method that you designed.
2. Assuming that the pizza orders were written to file or 'executed', what are 3 other functions that could be added to the pizzeria's web services to help the employees and the management?
3. Web services serve the same functionality as RPC, list the advantages and disadvantages of using web services instead of RPC (at least 2 for each).
4. What are 2 other application areas for web services besides ordering pizza? (Outside of a pizzeria environment).

Part III: Conclusions

What did you learn from this lab? What sections are unclear or you would like to know more about? What are some suggestions/comments/criticisms of this lab and the tools used?