■ **ECE/CS 4984: Wireless Networking and Mobile Systems** ■
Pre-lab and In-class Laboratory Exercise 4 (L4)

Part I – Objectives and Lab Materials

Objective:

The objectives of this lab are to:

- Introduce the client-server computational model
- Introduce writing a web service client
- Understand the role of middleware in this context (including SOAP and XML)

After completing the assignment, you should be able to:

- Characterize design requirements of example client-server applications
- Design a simple web service client using C#

Hardware to be used in this lab assignment:

- DELL notebook with 802.11b card
- iPAQ with 802.11b card and cradle
- Intel 802.11b Access point (GTA will setup one AP for the class)

Software to be used in this lab assignment:

- Notebook: Microsoft Visual Studio .NET, .NETcf

Overview:

Web services resemble the remote procedure call (RPC) ideology. They are a way for a client to call methods that are located on a service node somewhere else in the world. Web service idea differs from RPC in many ways. The first is that web services use standardized protocols to exchange information. HTTP, SOAP and XML are examples of this. HTTP is used as the transport mechanism to move data between 2 nodes. SOAP is used to enclose and give XML a context to be interpreted. Finally, XML describes and carries the data from one node to another. This methodology ensures that development time is faster, you don't have to create protocols from the ground up; safer, the pieces used are established and tested, and more robust, SOAP and XML can take care of describing data without the developer even knowing about it. All of this leads to a happier land of remote processing.

That all sounds good, but what do we need web services for? One reason is that mobile devices have little computational power compared to larger devices. Why process something locally and spend 5 minutes wasting battery power, when you could send the data off to a larger system that can do it in 5 seconds without any drain on your battery? Another reason is to obtain information quickly and easily for use in an application. Web services offer a standard way to exchange information between applications. Mobile applications like stock quotes, on-line purchasing and inventory control often need to gather or send information to larger systems. Web services offer an open and standard way to accomplish this. In this lab we will work with describing and implementing web services for use in just such as scenario.

Part II – Pre-lab Assignment

This portion of the assignment should be completed *prior* to the in-class lab session.

Reading Assignment:

- Understand basic web service concepts "lab4_webservices.doc" posted on the class web site
- "An XML Overview Towards Understanding SOAP" - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/xmloverchap2.asp
- If web services interest you (optional) - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/webservicesanchor.asp

Tasks: Ponder:

- Think about how web services could be used to your pizza pricing application in lab 3. What services could be offered by adding a server back-end to your pizzeria? (Document at least 2 in your report).

Part III – In-class Lab Assignment

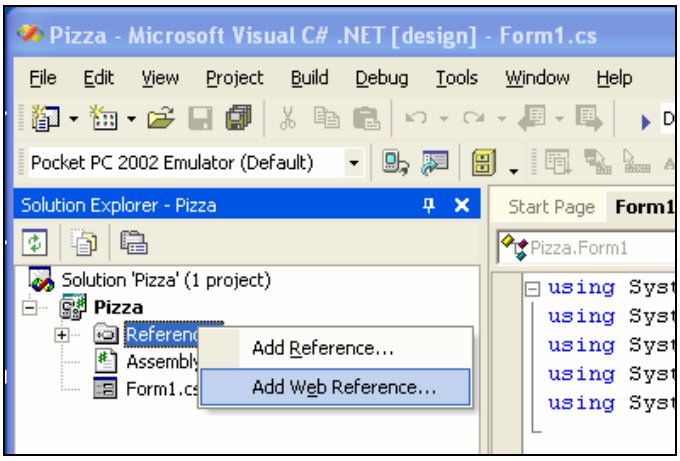
Part A: Order up!

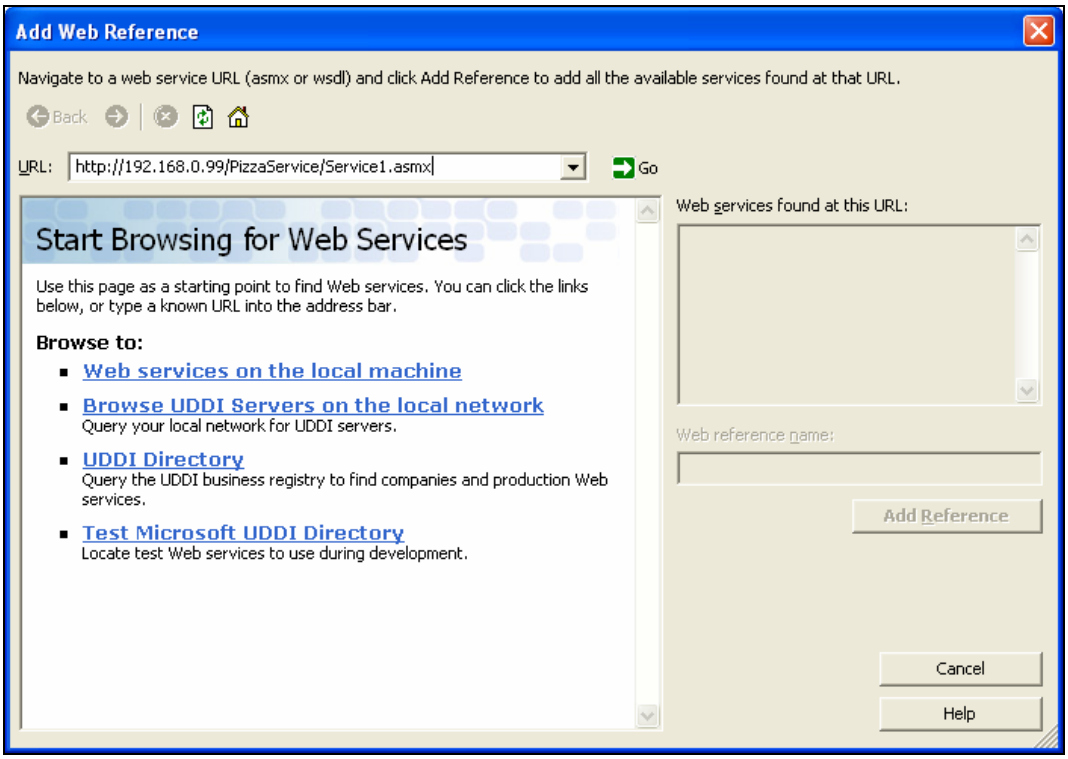
After implementing lab 3 and teaching your graduate student employees how to use it, you find there are still issues concerning the ordering process. Frustrated, you search for another idea. You remember hearing something about web services in your mobile systems class and decide to look into it. Alas, you find an answer. You decide to implement the ability to place an order using the iPAQ. Now the employees can simply hand the iPAQ to the table and the order can be placed when the patrons are ready.

In speaking to the professors of your course, they agree to help you. They agree to design the server back-end of the service and elect you to design the client-end that runs on the iPAQ. Fortunately, you already have a portion of the code written from lab 3. =D

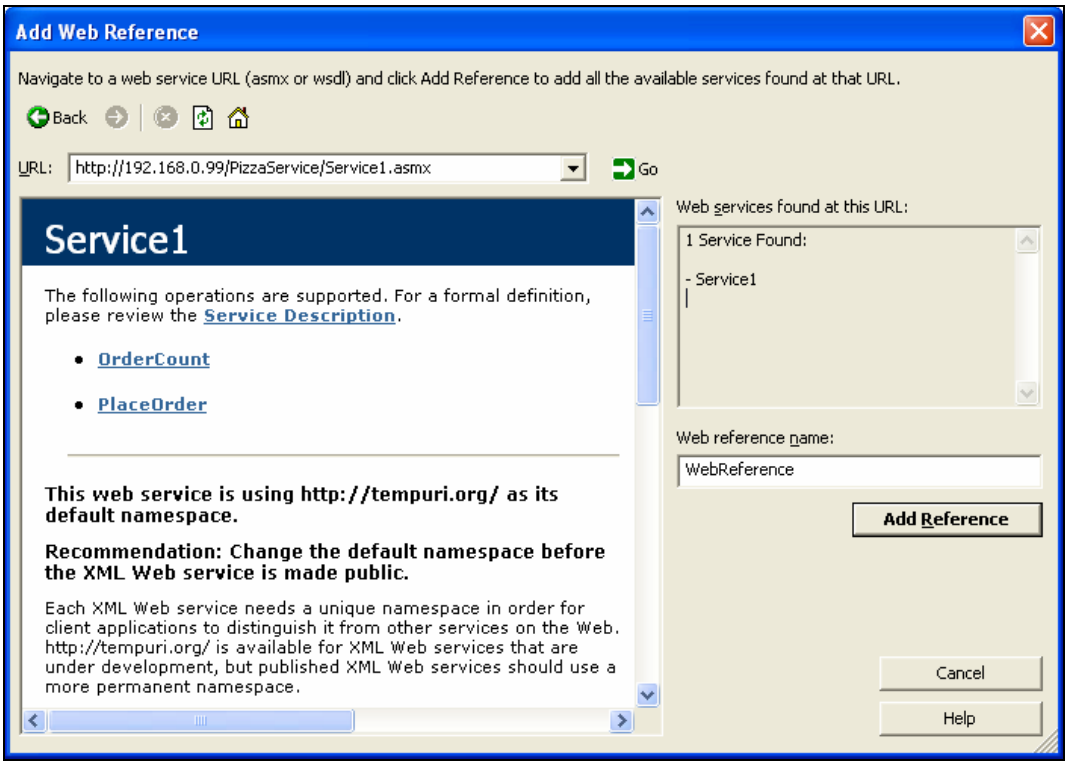
Part B: The Client

There are 2 pieces to a web service client that you need to make remote method calls. The first is a *web reference*. The web reference tells the application where to find the web service description; this is essentially a URL. When you create your client you have to add a web reference. This is done in the 'Solution Explorer' box of VS. Right click on the *References* tab and choose *Web Reference*.

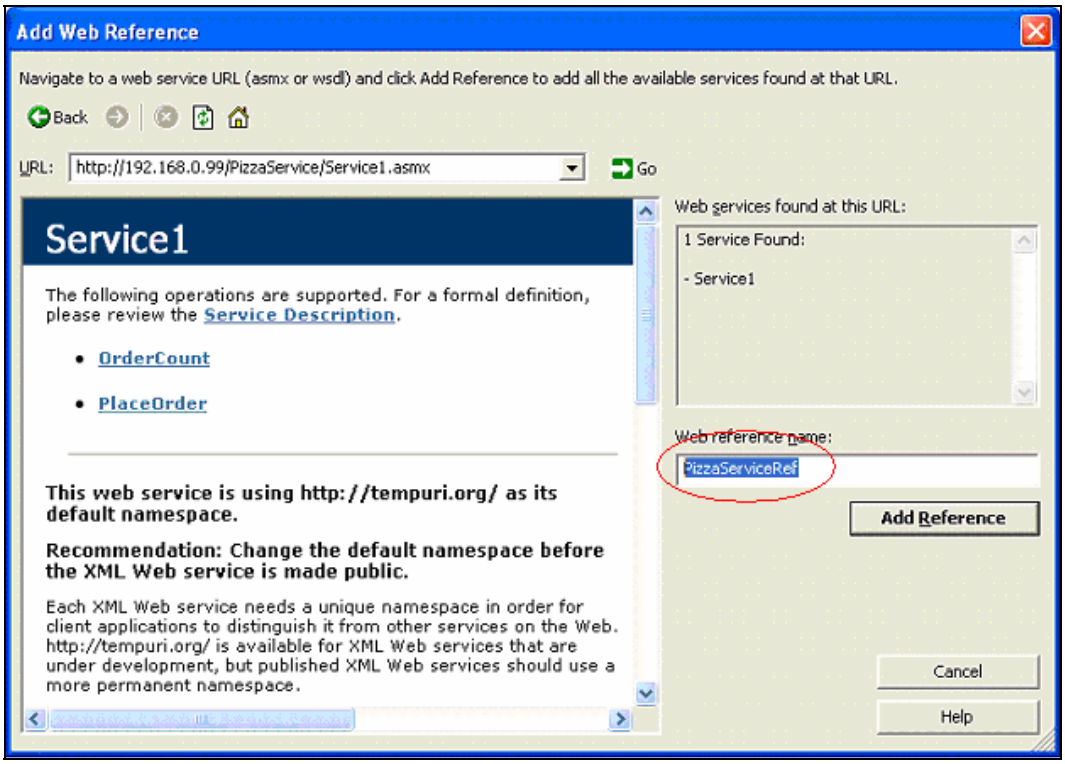




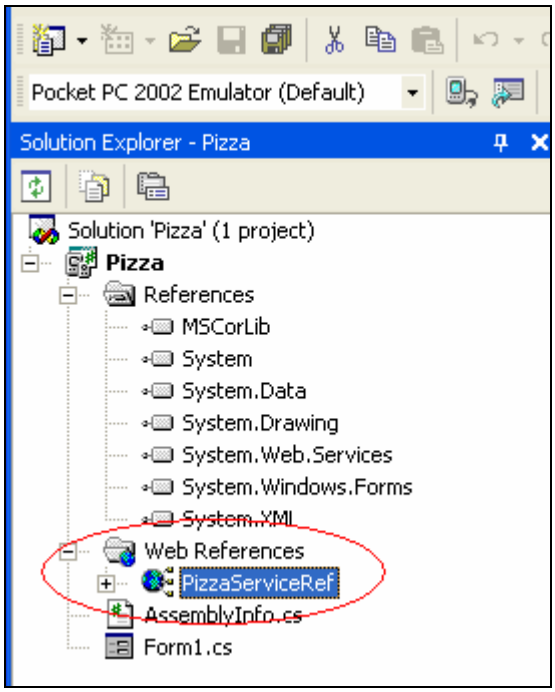
In the URL section, you will put the URL of the web service. The GTA will give you the URL for the web service. (For this example, we will use http://192.0.2.100/PizzaService/Service1.asmx as our web service URL). Click 'Go' and VS will locate the web service and download its description file.



In this case, our web service has two operations available: OrderCount and PlaceOrder. Change the name of the web reference and click 'Add Reference'.

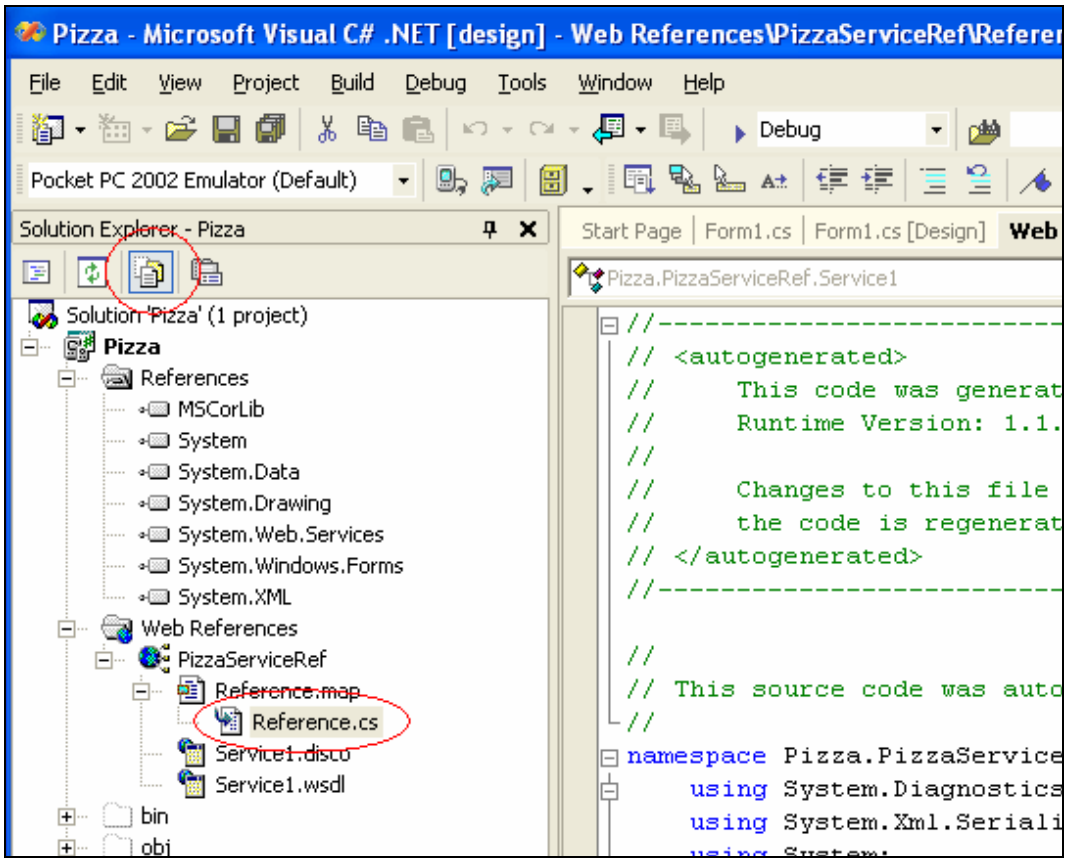


This will place a web reference into the project.



Let us go back a couple steps and talk more about web references first. In order for an application to make remote call, a *proxy service* must be present. The proxy service handles taking the parameters and *serializing* them, so they may be sent to the service. The proxy service also handles extracting the returned data from the web service and passing it back to the local calling method. Visual Studio helps us

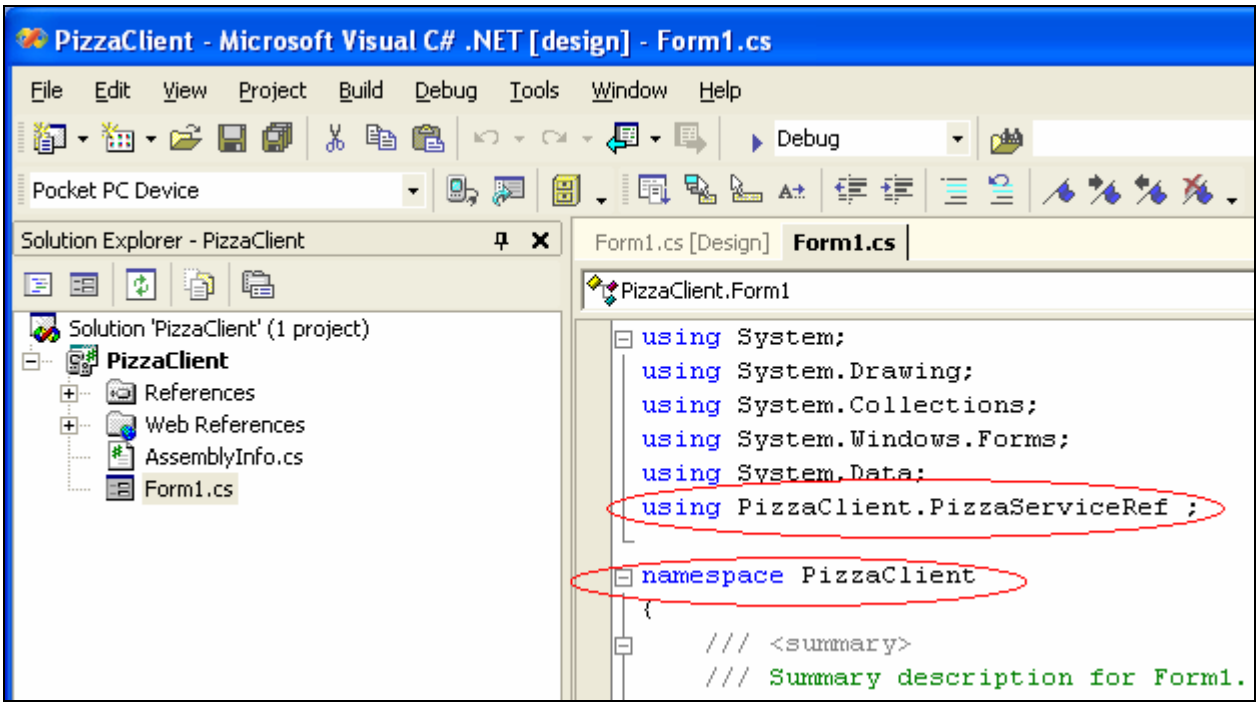
out here and creates the proxy services in the background for us. To view this code, first, click the 'Show all files' button in the solution explorer. Open the 'Reference.cs' file by double-clicking it. Scroll through this file and take a look at the auto-generated code.



You may return to your other code now.

Not only can we make calls to remote methods, but we can use objects defined on the remote service as well. This means we don't have to define classes that we might be passing around in two places. If you think about it, this is extremely handy and safe. You may not pass object methods across nodes. This makes sense, because you might have a different hardware platform on the other end which means that your local code might not run. Object data attributes may travel freely though.

With the web reference in place, you may make calls to any *web methods* defined. You will learn about writing web methods in the at home portion of this lab. The web reference is found in the object, *PizzaClient.PizzaServiceRef*. *PizzaClient* is the namespace that has been defined for your project; this can be found at the top of your client application. *PizzaServiceRef* is our web reference that we defined when we added it. This can be found in the 'Solution Explorer' box under the 'Web References" section. Instead of writing out the whole path, you can make use of the *using* keyword and make life easier. See the 'using' line below as an example; substituting in the appropriate class names.



For this lab, you will be using a *pizzaOrder* object as well as calling a method that are both defined on the remote service. The pizzaOrder class will be used to place an order using the web service. You will fill in all the information fields and all the *placeOrder ( )* method, sending it the pizzaOrder object as the sole parameter.

The pizzaOrder object is defined in the following way:

```
public class pizzaOrder
{
      public int size;        // 0-small 1-med  2-large
      public int crust; // 0-thin  1-orig 2-sicilian
      public int serverID;    // id of server
      public bool pepperoni;
      public bool sausage;
      public bool ham;
      public bool peppers;
      public bool onions;
      public bool pineapple;

}
```

All values are fairly self explanatory. The Booleans for toppings are set to true if the topping is to be included and false if not. The *serverID* attribute, is your group number.

The placeOrder ( ) method is defined as:

```
public string PlaceOrder ( pizzaOrder order )
```

If successful, it will return the order string written to file by the service. If unsuccessful, it will return "busy…".

Now that you have the tools for the job, you may begin. Design a user interface (UI) using pieces from lab 3 to construct an application that allows employees to order using the iPAQ as well as price. The GTA's notebook will offer the web service of taking orders. You will need to add a button offering the 'order' functionality as well as a small area for the return value to be displayed.

Part C: The Setup and Procedure

1. The GTA will have an access point setup using WEP. Setup the iPAQ and the laptop to attach to the access point. Give the notebook an IP address of 192.168.1.<100+your group number>. Give the iPAQ an address of 192.168.1.<200+your group number>. Each should have a netmask of 255.255.255.0. Setup WEP using the password as "ABCDEF4984". Check the connections by using the notebook to ping the GTA's notebook and your iPAQ.
2. Modify your lab 3 Pizza application to support the additional functionality needed by this lab. DO NOT overwrite lab 3; just start lab 4 off with a copy of it.
3. Use the return value of the service to debug your code.
4. Use ethereal to capture one of your order sessions. (Use filtering to make sure that you are seeing just your session.) Use the 'Follow TCP' stream attribute of ethereal to view the entire session and save it to a file. The trace will be analyzed later.