Web Services Tools and Demo

Typical Tools for Design, Development and Deployment of Web Services

With demos of creating simple WS clients and WS servers

Version 15

WS tools and demos – the "How To's" of WS

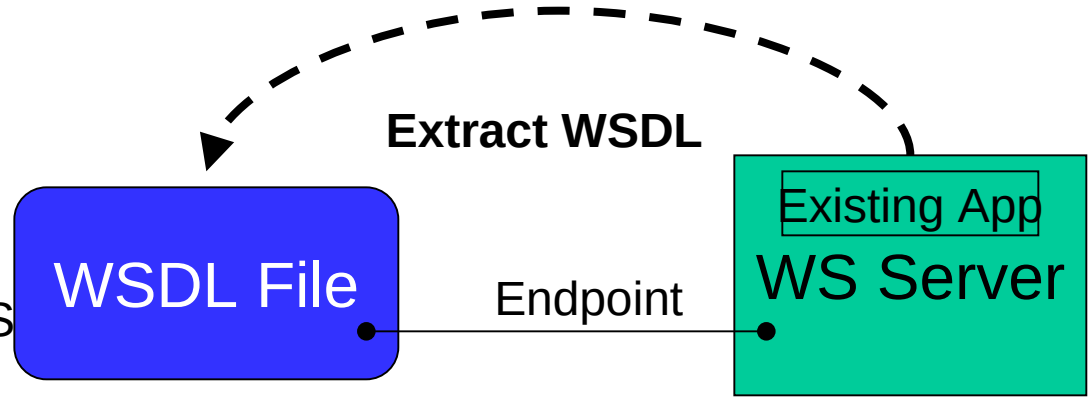
- Tools are needed for:
  - Creating/defining XML documents and schemas
  - Creating/modifying service definitions/WSDL files
  - Mapping WS artifacts to/from programming concepts
    - Both directions may be needed depending on the integration scenario
  - Managing and locating deployed Web Services
- Some typical and available tools:
  - XMLSPY – General Purpose for XML docs and WSDL files
  - Artix – C++/Java-based Web Services
    - Enterprise Web Services & middleware integration toolkit from IONA
  - VS.NET tool set - Microsoft
  - J2EE/App Servers – J2EE in general – e.g. EJB
  - Many vendors and many tools, … and more every day
    - IONA, IBM, Sun, BEA, Cape Clear, Altova (XMLSpy), etc.

Two Basic Approaches for building a Web Service

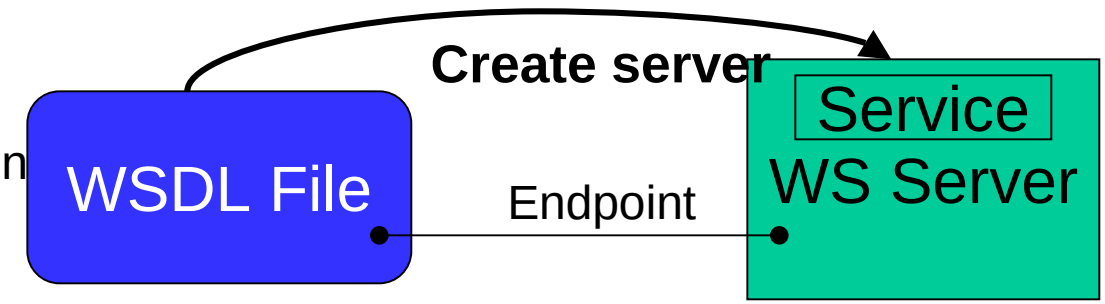
- From Scratch – a new application or system
  - Start with WSDL and/or data/message schemas and produce the application using generated application building artifacts:
    - "blank" client and server, client-side stubs & server-side skeletons
  - Write the business logic in the "blank" server
  - Fill out the "blank" client code
- From Existing Systems – either
  - Automatically derive the WSDL from the existing interfaces for the application or from data schema files
    - There are automatic WSDL generators for CORBA IDL, Java classes, J2EE EJBs, COBOL, other middleware (MQ, Tuxedo,…), message/XML Schema files, . . .
    - Often you have to enhance the generated WSDL
  - Do a top-down design of the WSDL, and implement the service using new code and the existing systems
    - You may get better WSDL. Remember that one of the main goals of a service is to keep its clients simple.

Diagrams to show these Development Approaches

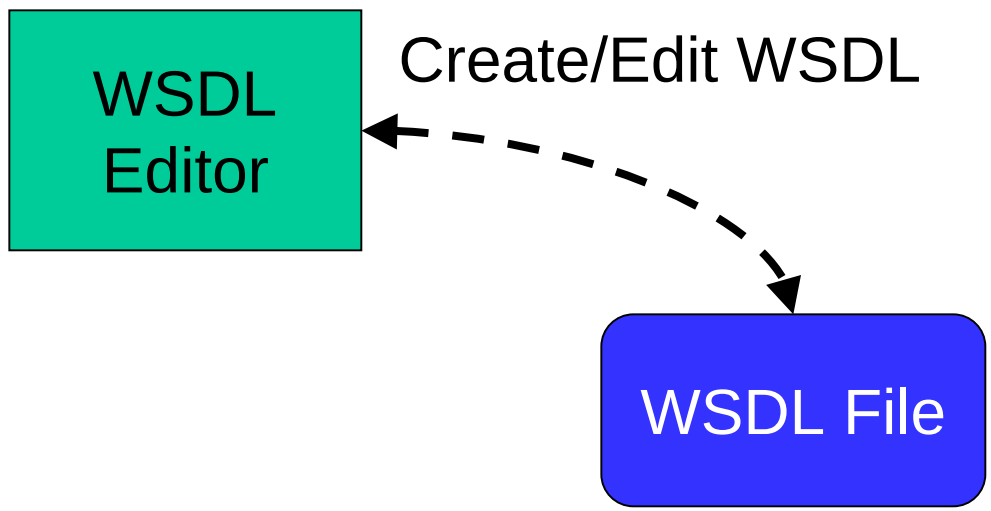
Existing App has an API
Extract WSDL from API
Build client and server skeleton
Client calls App API through WS



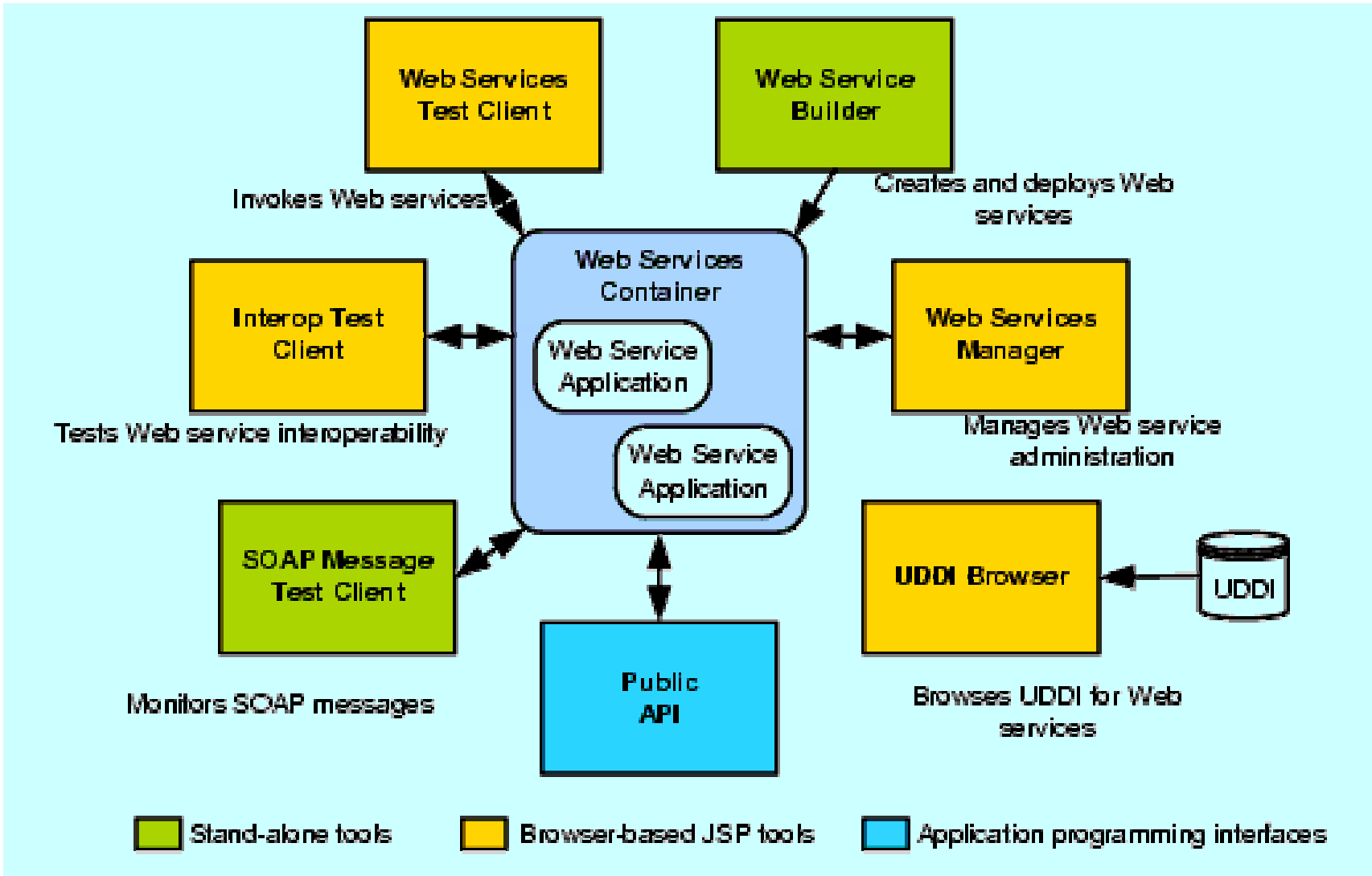
Design a WSDL (top down)
Create client and server skeleton
Implement server/service code
Client calls service through WS



Design/update the WSDL
Utilize in above scenarios during development/evolution of system



Typical WS Development/ Deployment Components



Development Demo Scenarios

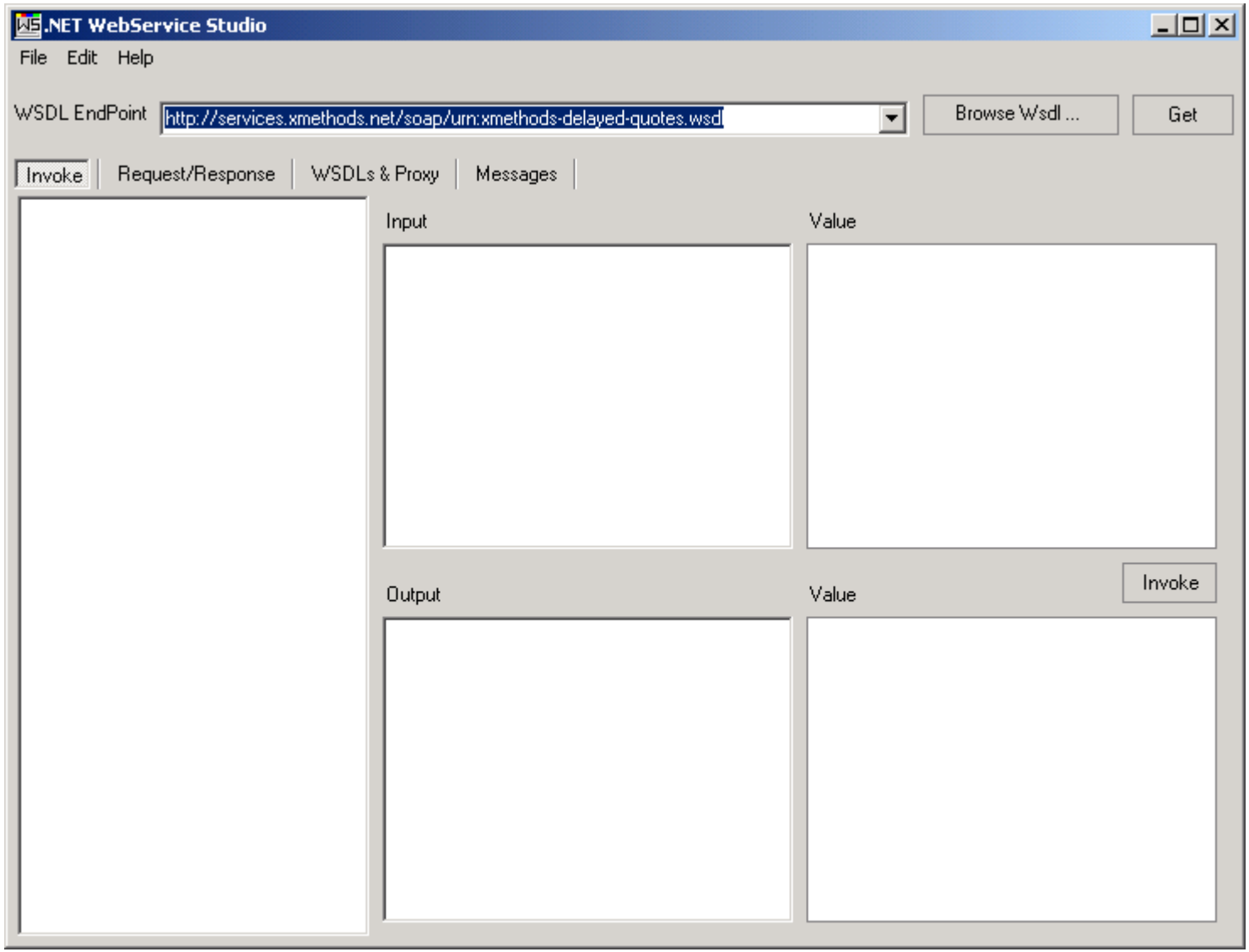
- Automatic client generation
  - Seeing WS clients in action …
- Hello World – VB.NET client creation
  - Local Artix server
- Stock Quote – VB.NET client creation
  - Our original example - again

- Hello World – VB.NET Web Service Creation

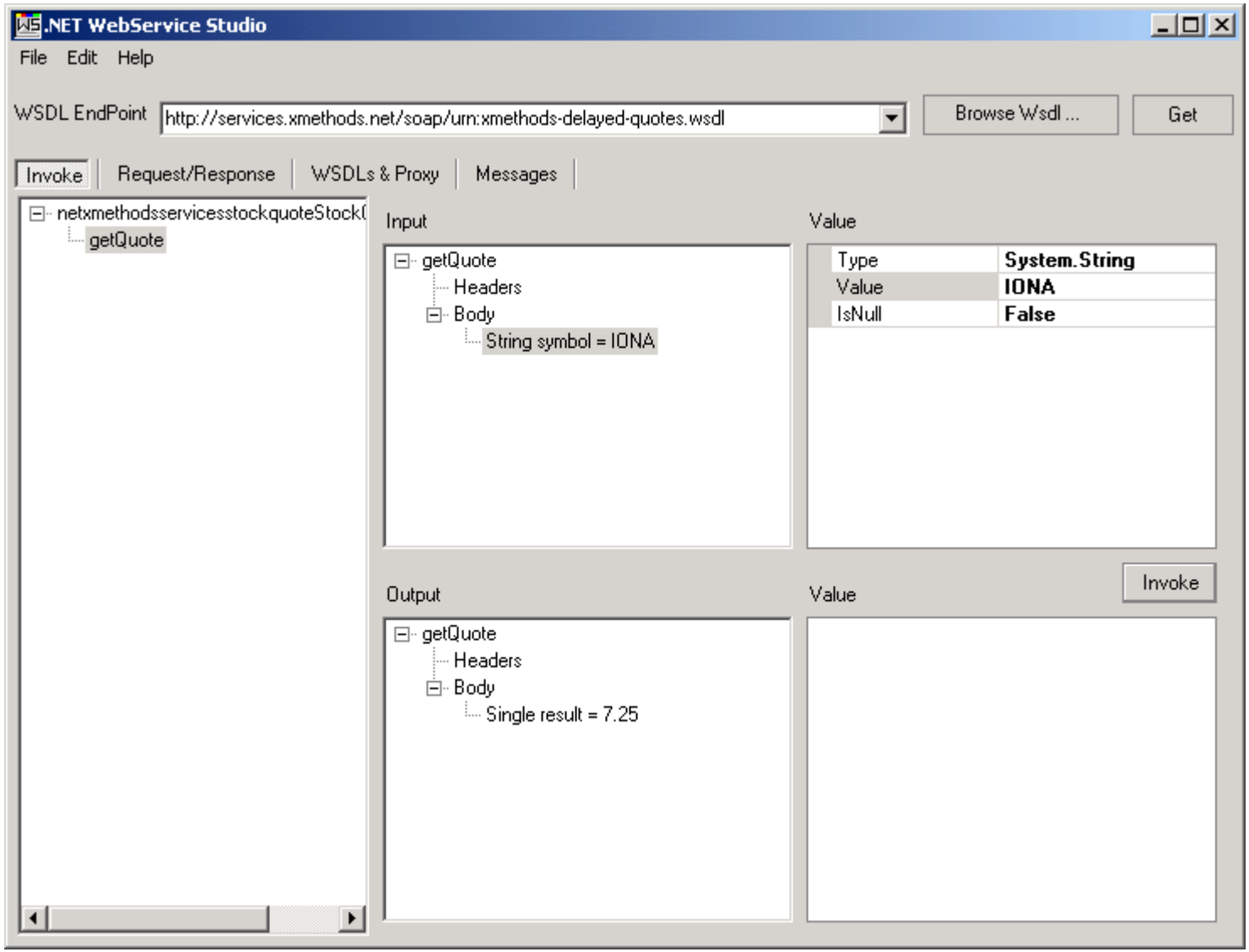
Automated WS Client Generation

- A number of tools available for this task
- Generate Java, VB, C#, Javascript, ASP, JSP and other
- We will use **Web Services Studio 2.0**
  - Freeware product – 114KB – simple install
  - Reads/analyzes a WSDL file, located anywhere on the net
  - Creates/displays automatically generated client/proxy code
  - Simplifies the invocation of WS methods with arbitrary inputs, and then viewing the response messages
  - Can view the actual SOAP messages involved
- Available from:
  - http://www.gotdotnet.com/ - locate WebServiceStudio 2.0 page
  - Requires .NET SDK environment
- Let's see it in action …

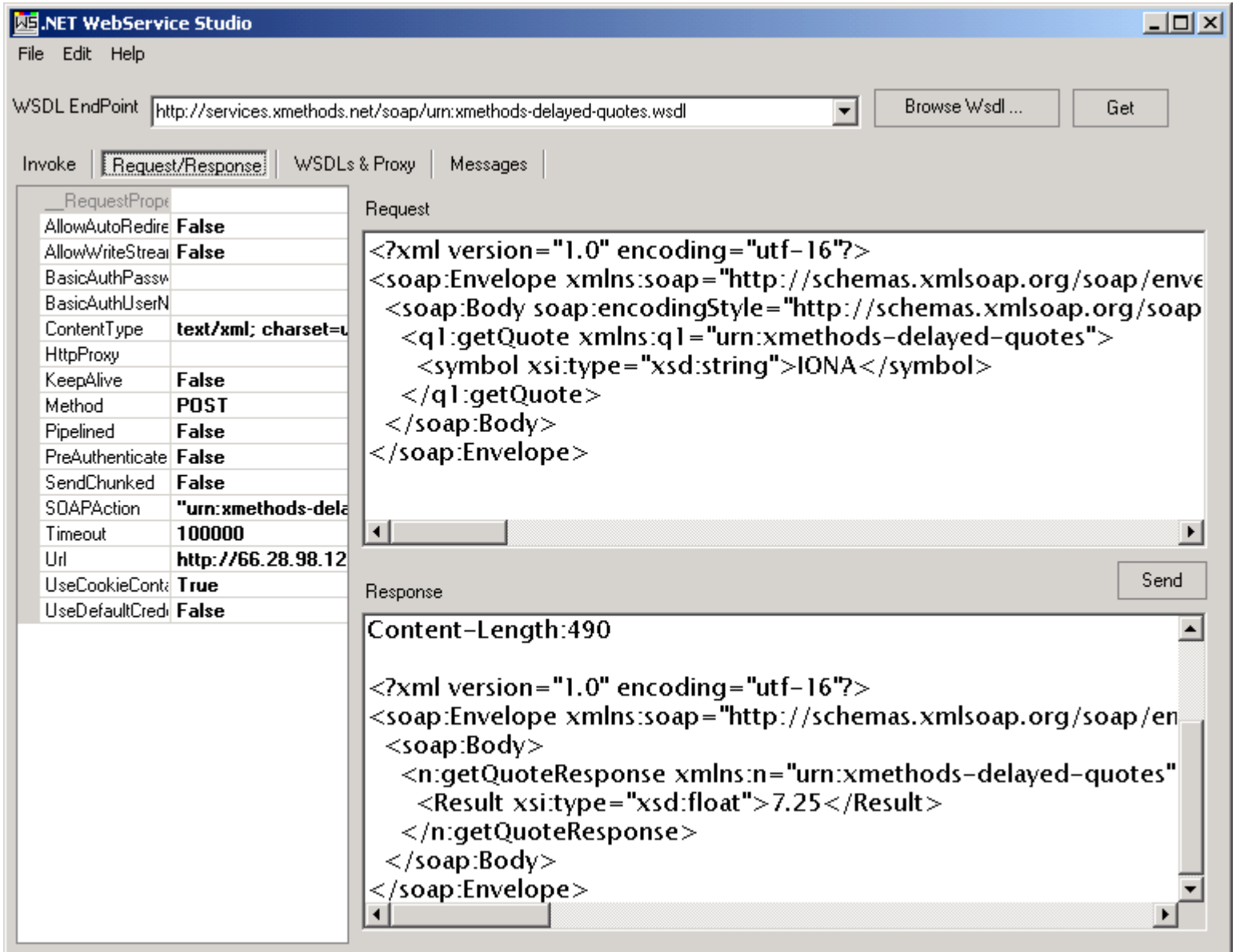
Screen Shot – WS Studio 2.0



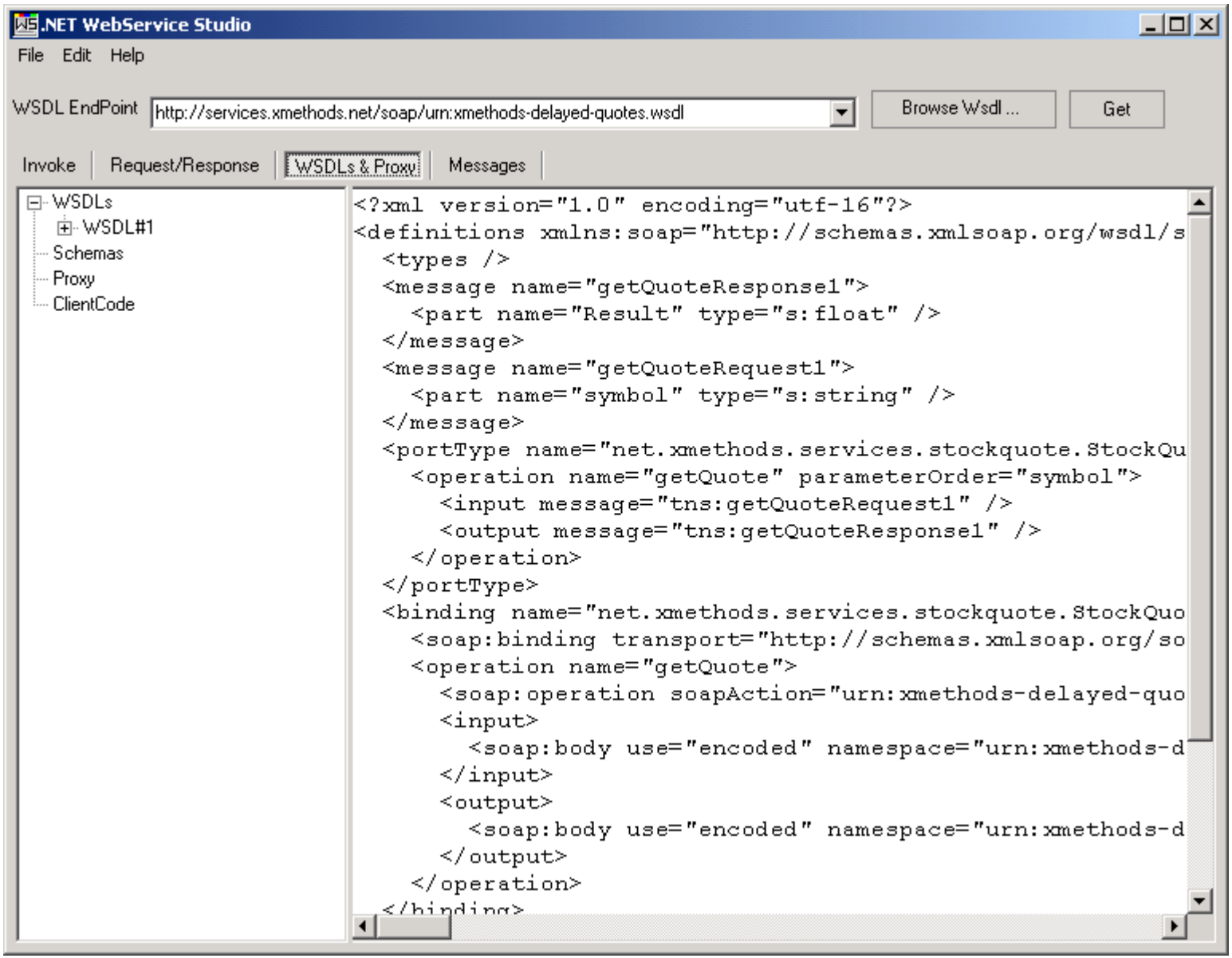
Web Service Client Created and Executed



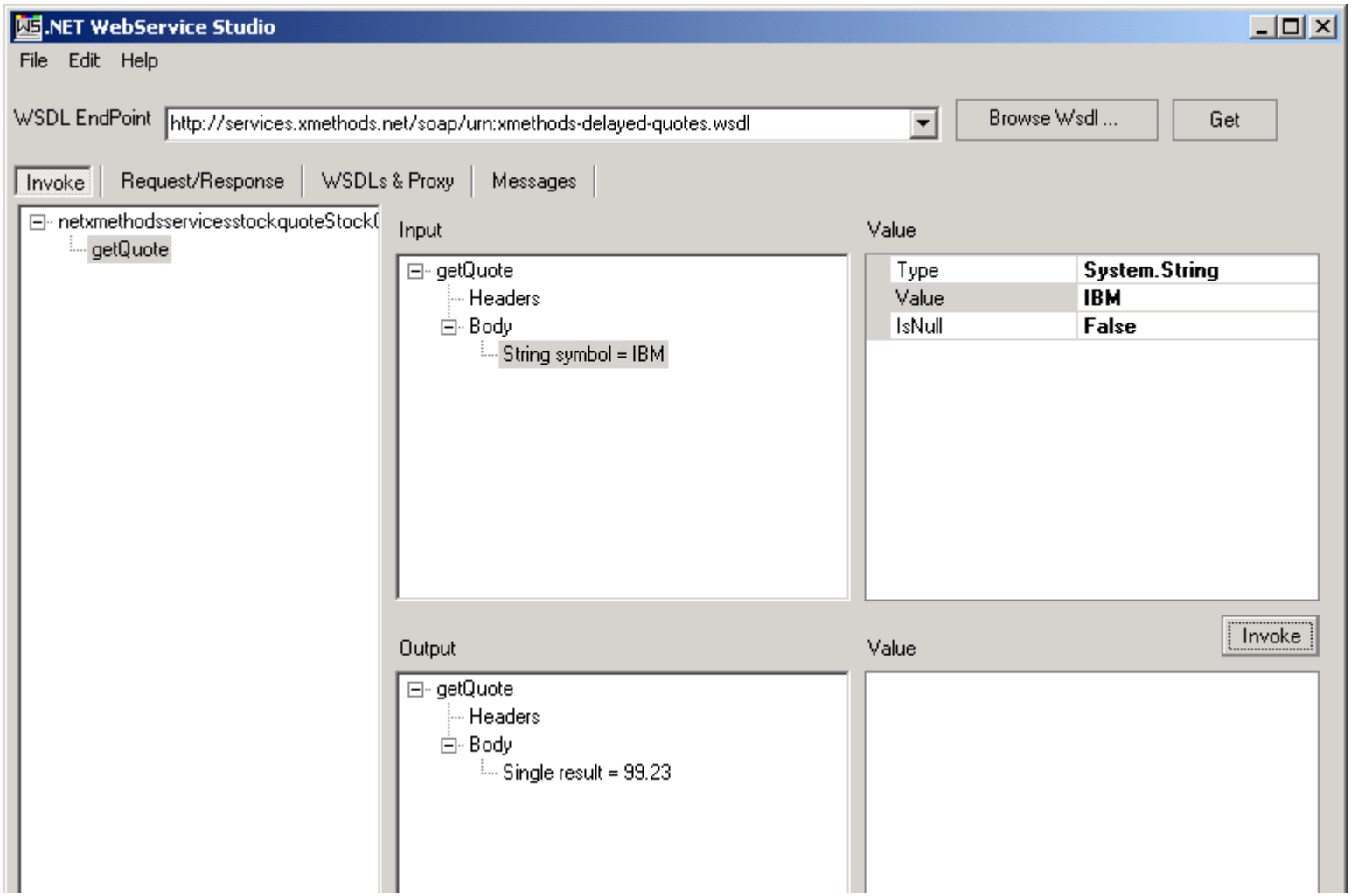
SOAP Messages to/from Service



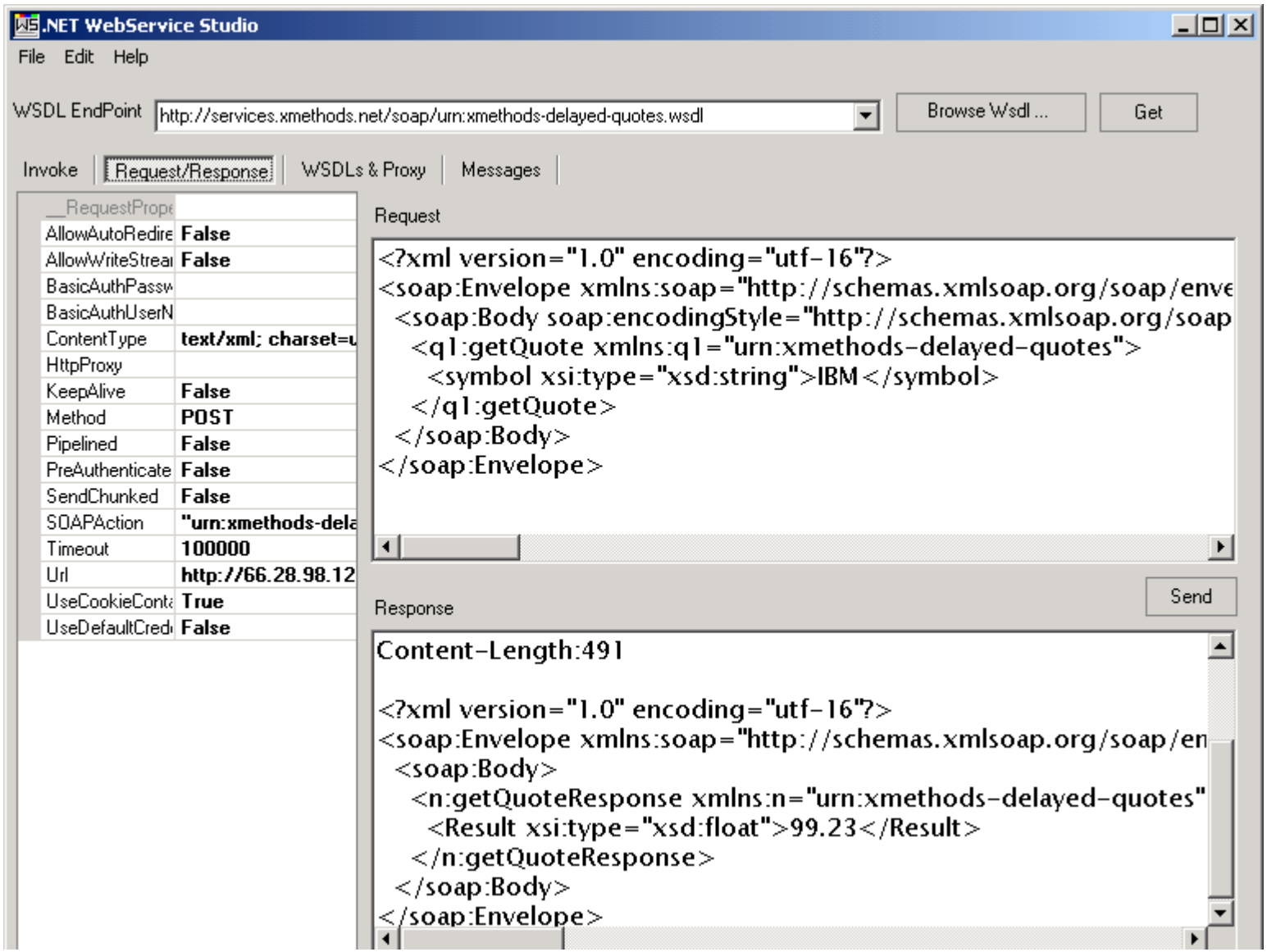
WSDL File display



A Different Stock Symbol … and result



A Different Stock Quote – the SOAP messages



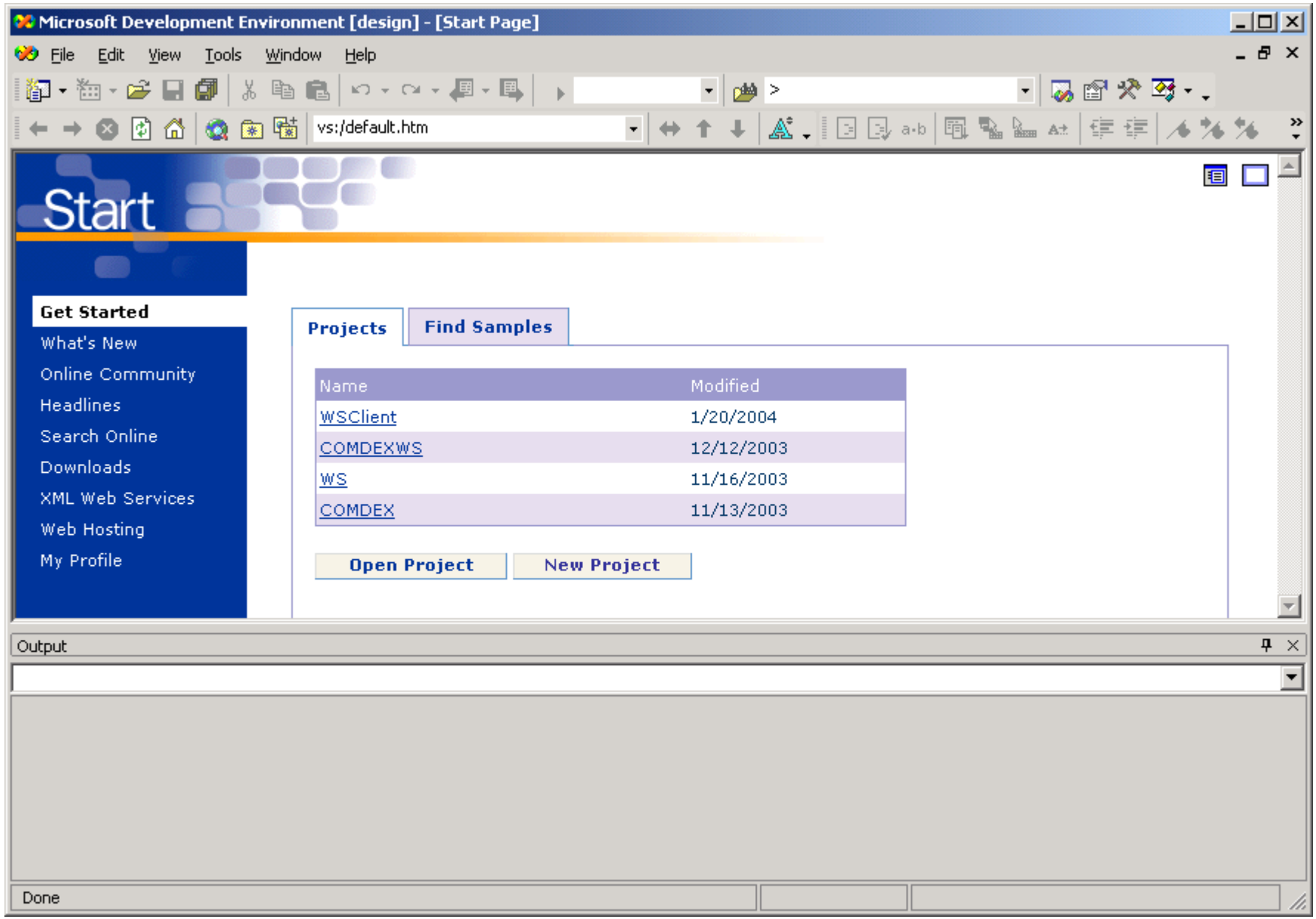
Creating Web Services Clients Automatically

Writing Client Code – Simple Examples

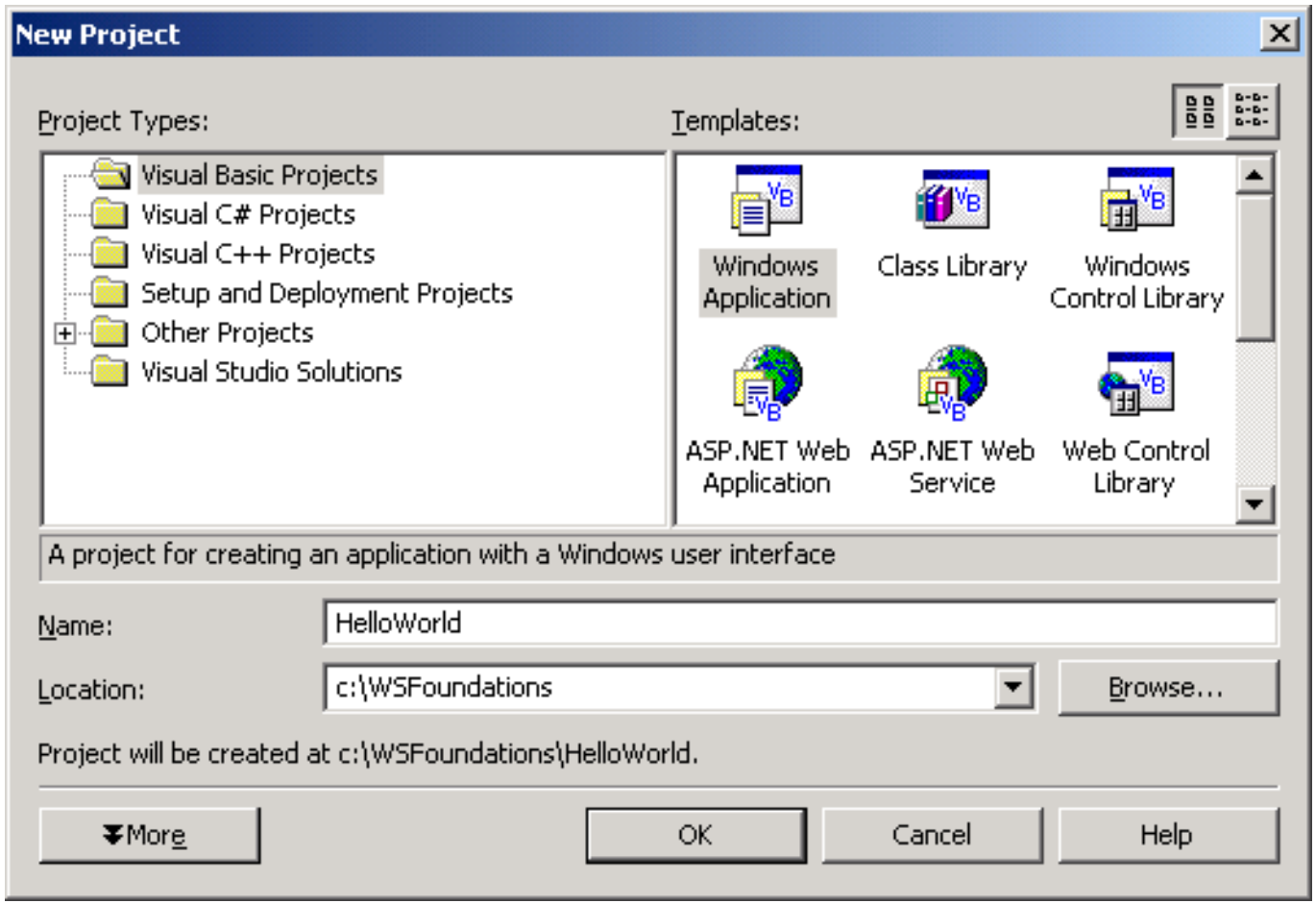
- Using Visual Basic within Visual Studio.NET
- Simple, unpolished examples
- Illustrate the underlying theme of ease of access to Web Services from most development environments
- Major building step
  - Add a "Web Reference" to the current project
  - Based solely on a WSDL file specification:
    - Local/remote file system
    - Remote URL

HelloWorld VB.NET Client

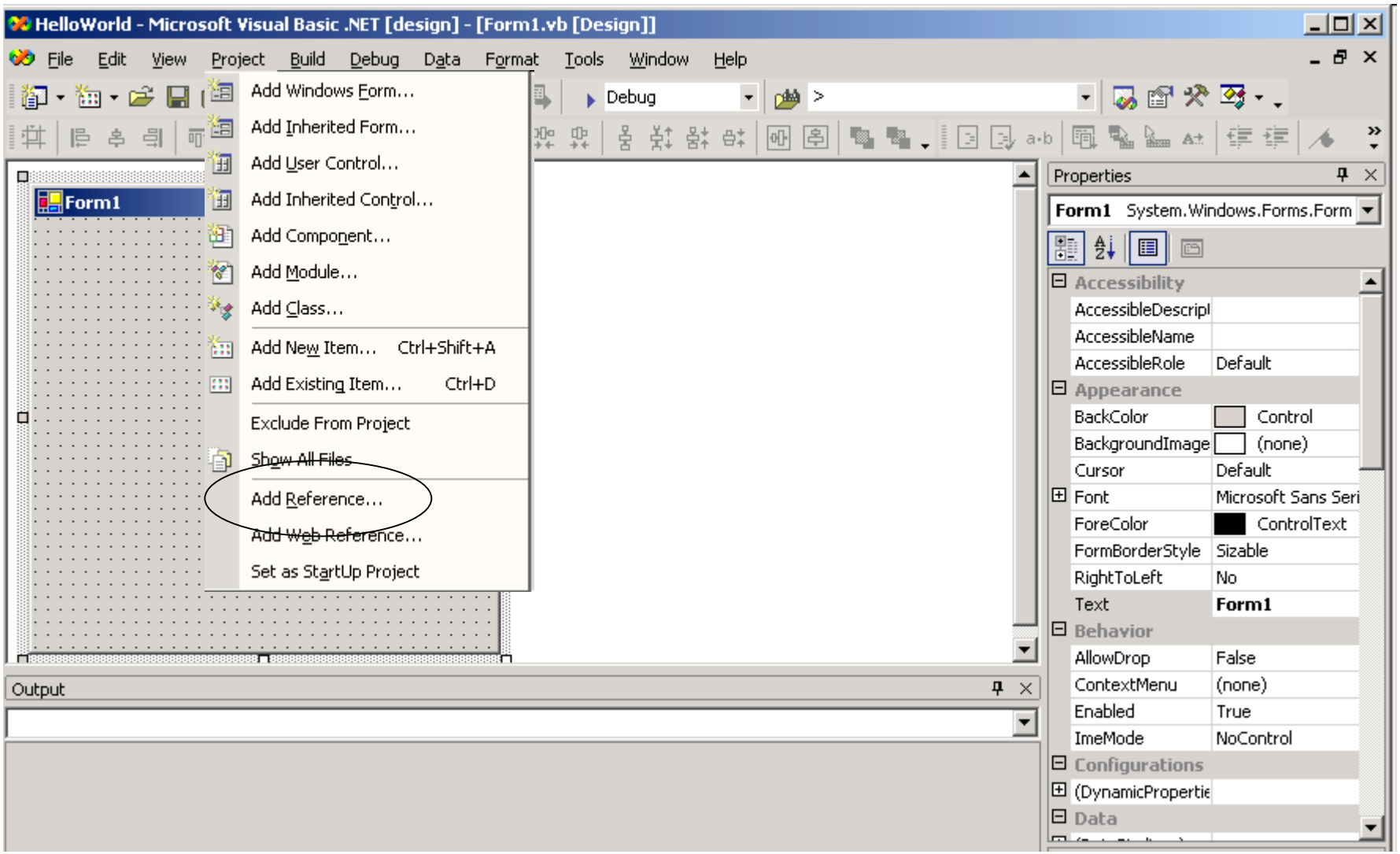
- Use a remotely stored WSDL file (HelloWorld.wsdl) on our demo system
  - Simple data exchange service
  - Only two operations – greetMe and sayHi
- Create a simple client (using VB.NET)
- Execute the client to invoke a remote service
  - Server written in C++ using IONA Artix product running on a remote server
- Let's do it!


Microsoft Development Environment [design] - [Start Page]
File Edit View Tools Window Help
vs:/default.htm
Start
Get Started
What's New
Online Community
Headlines
Search Online
Downloads
XML Web Services
Web Hosting
My Profile
Projects
Find Samples
Name
Modified
WSClient
1/20/2004
COMDEXWS
12/12/2003
WS
11/16/2003
COMDEX
11/13/2003
Open Project
New Project
Output
Done

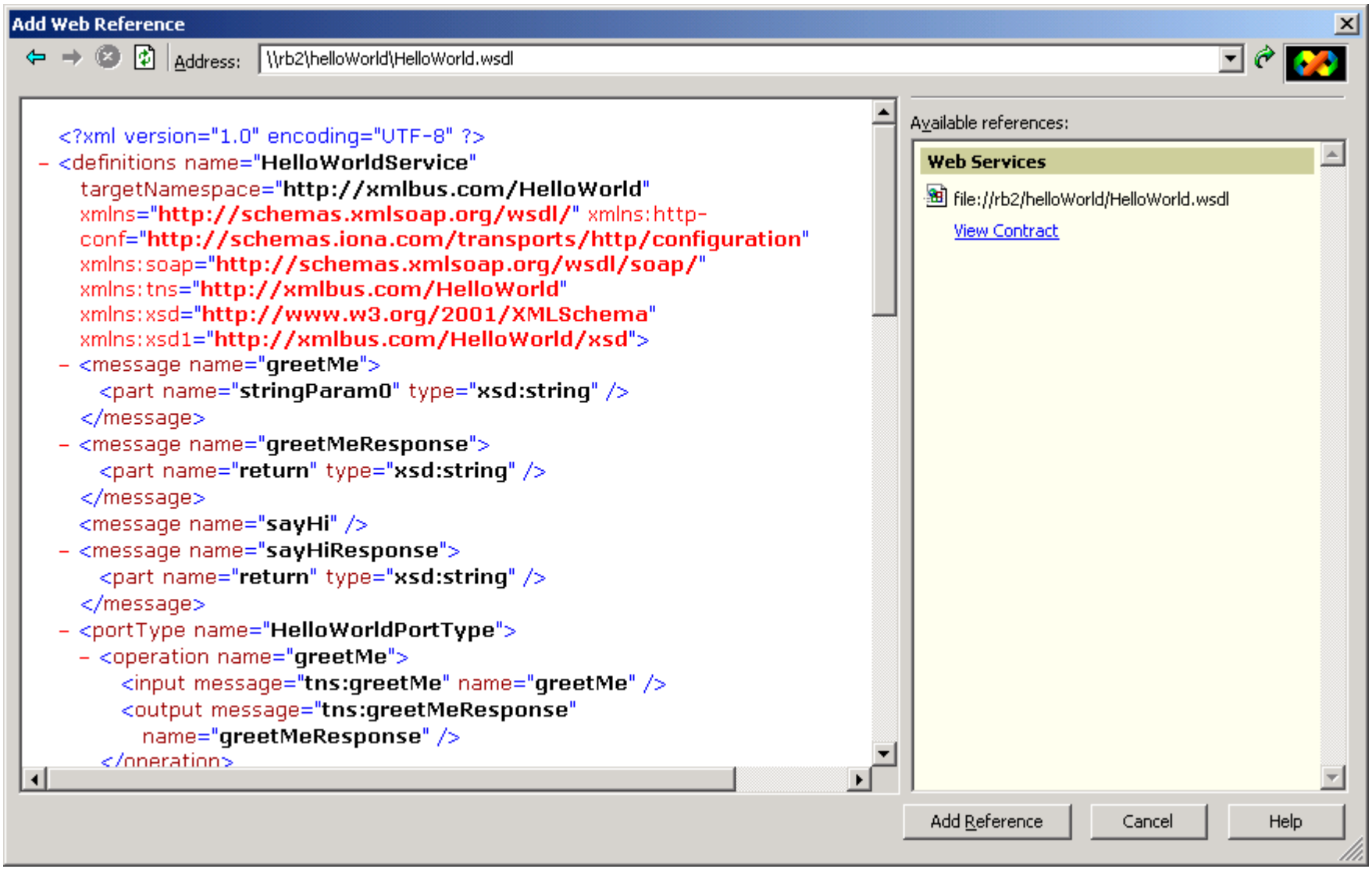
Windows Application for our Examples



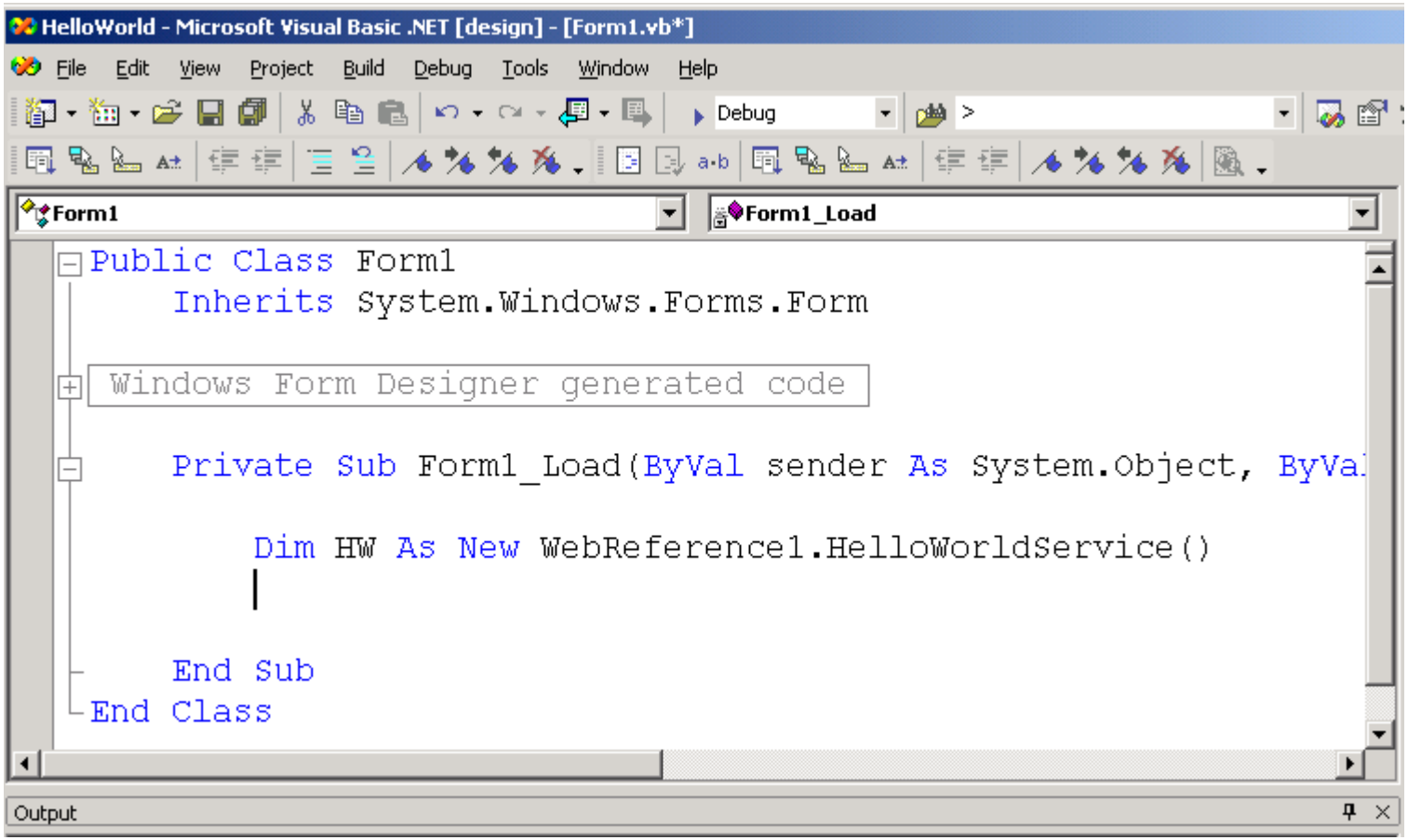
Key Step: Add a Web Reference



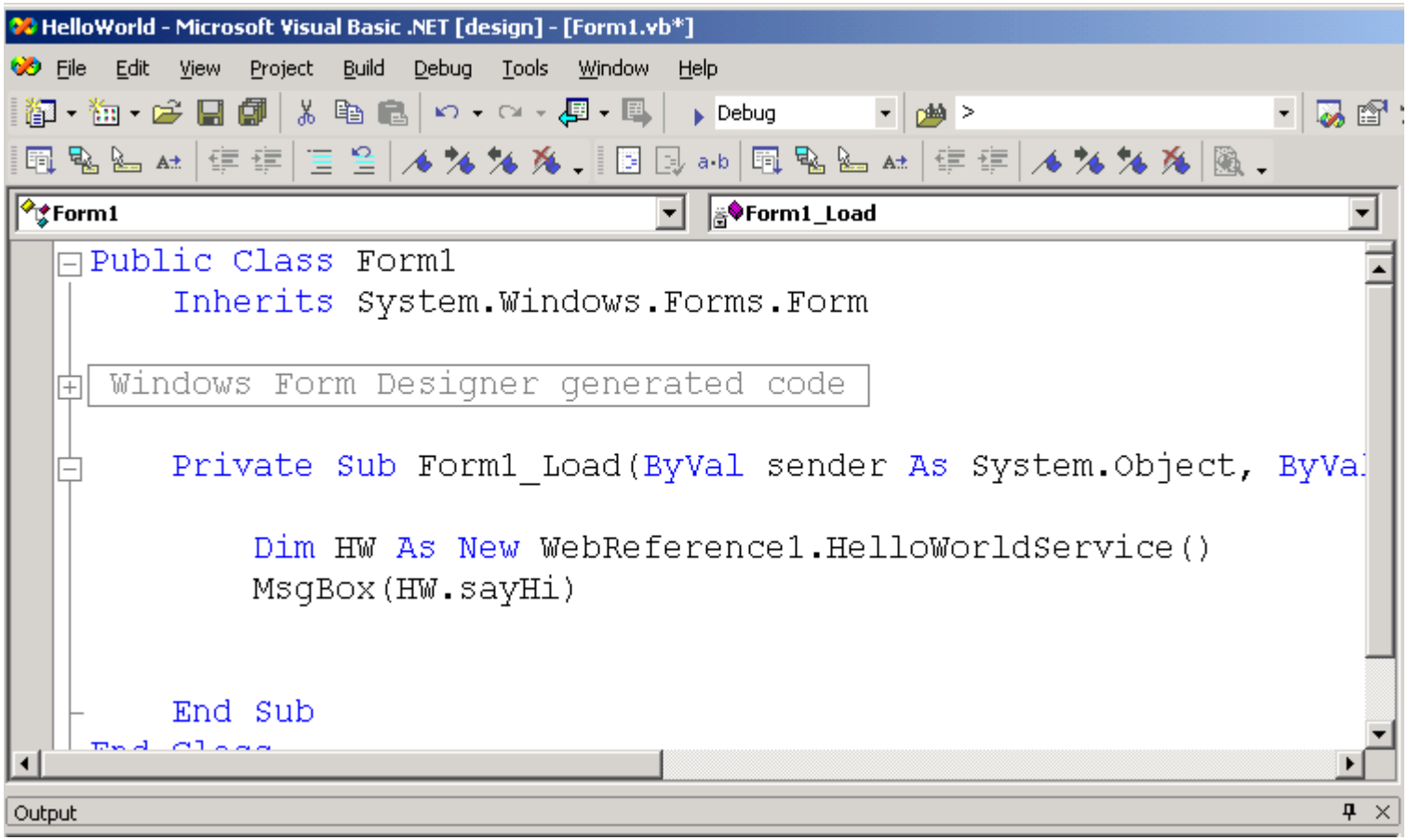
Add Web Reference Dialog



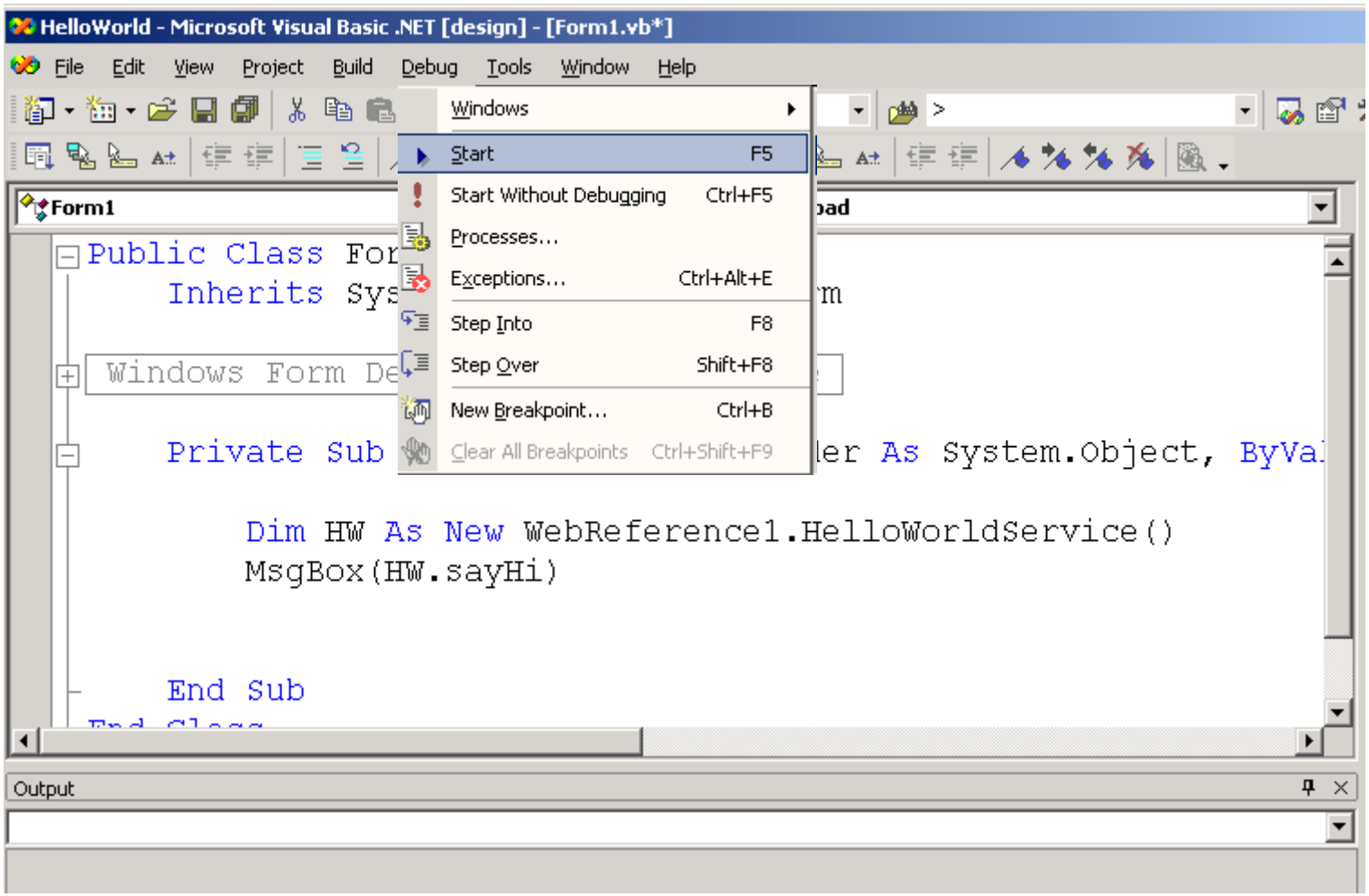
Create a Proxy object to WS server



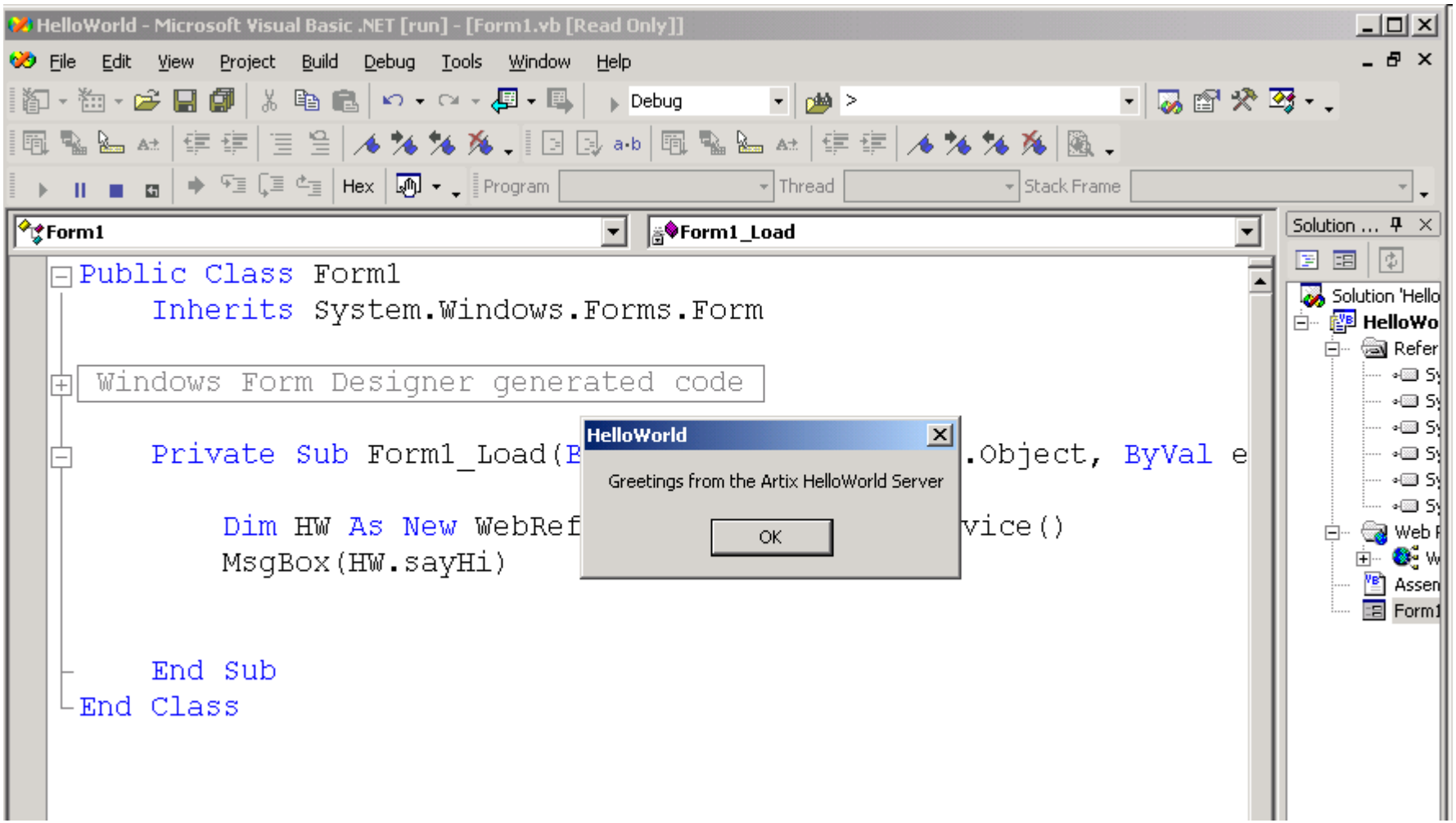
Invoke operation via Proxy Object



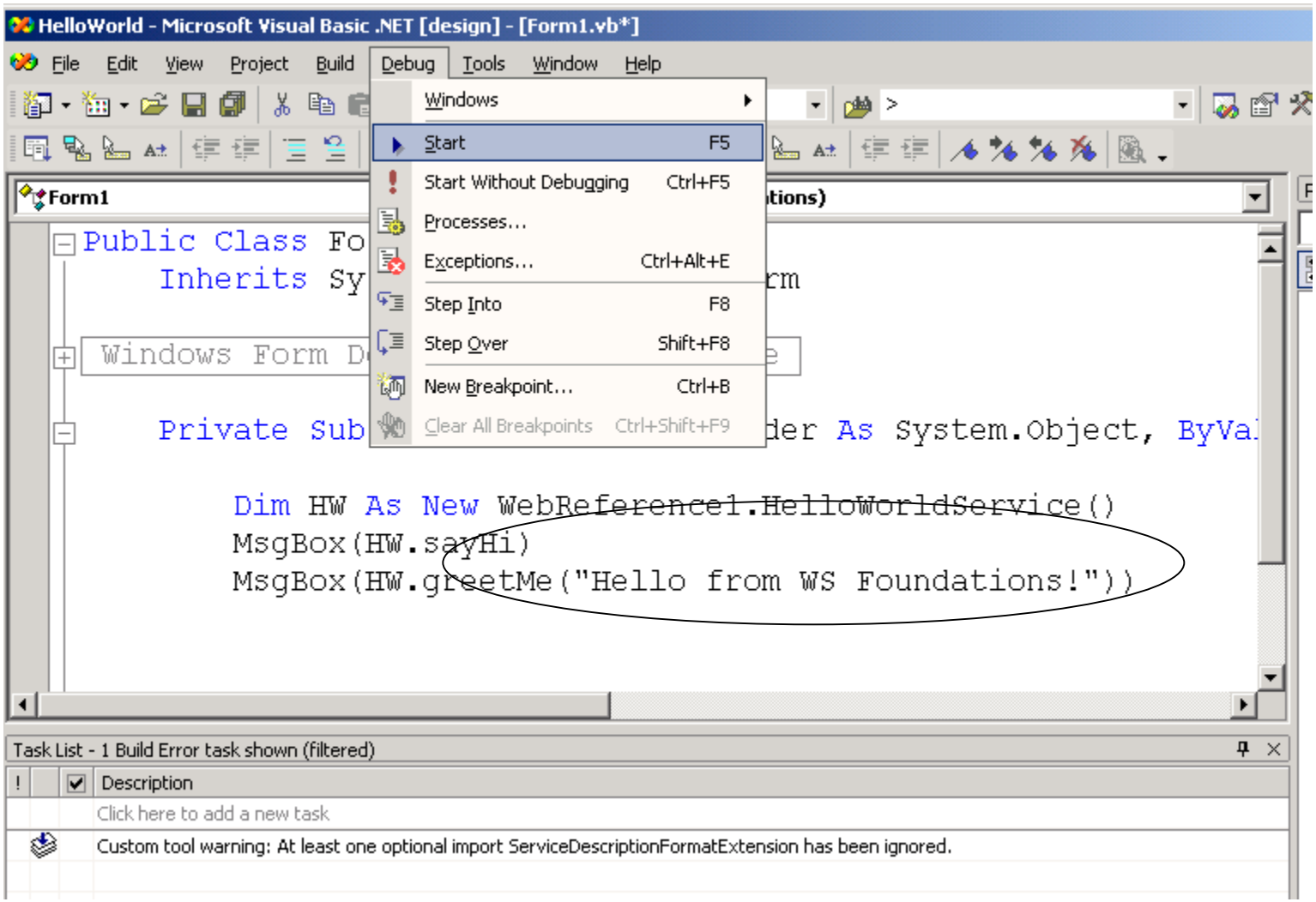
Invoke using the Debug menu



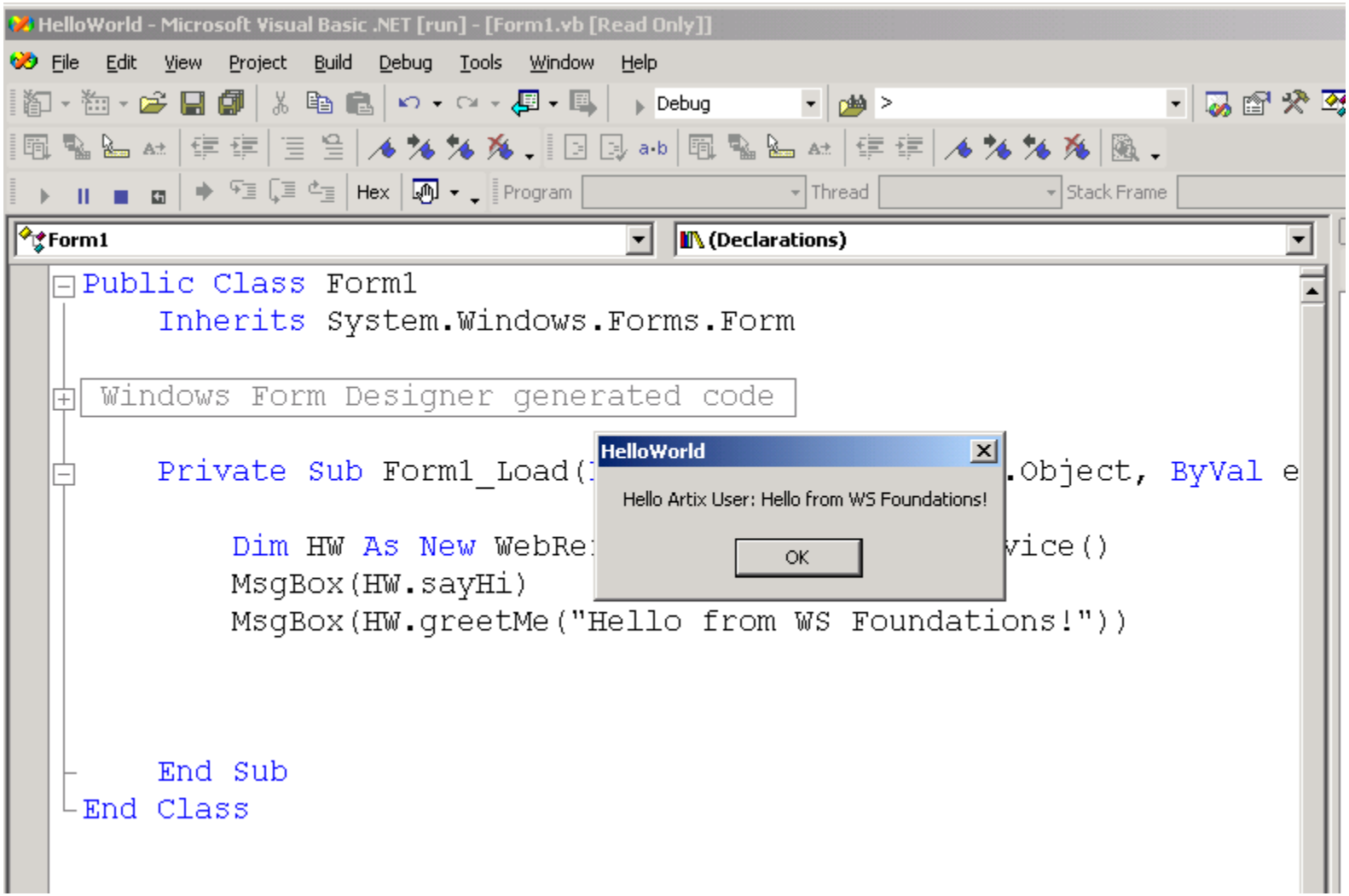
Web Service response



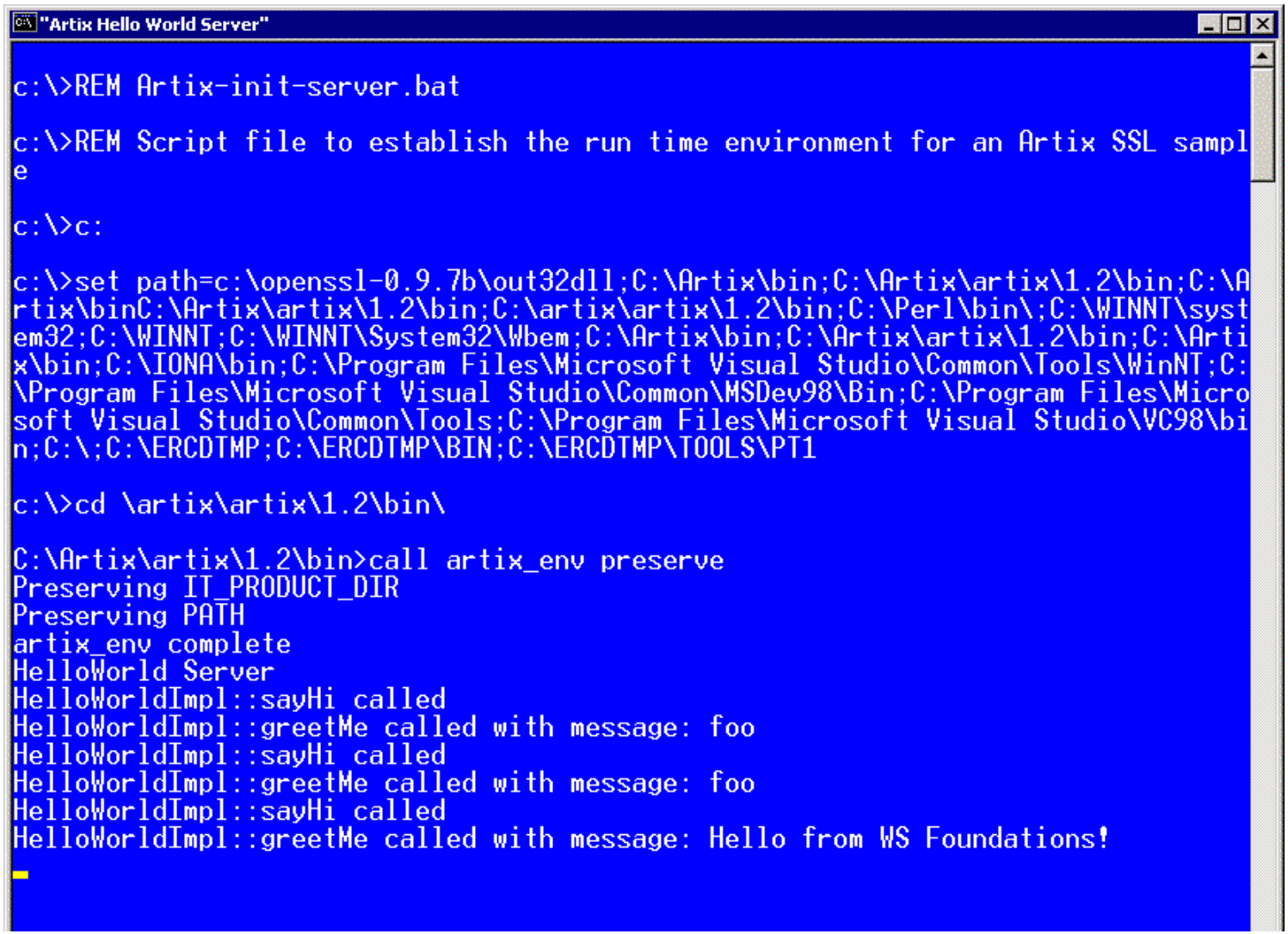
Invoke another operation



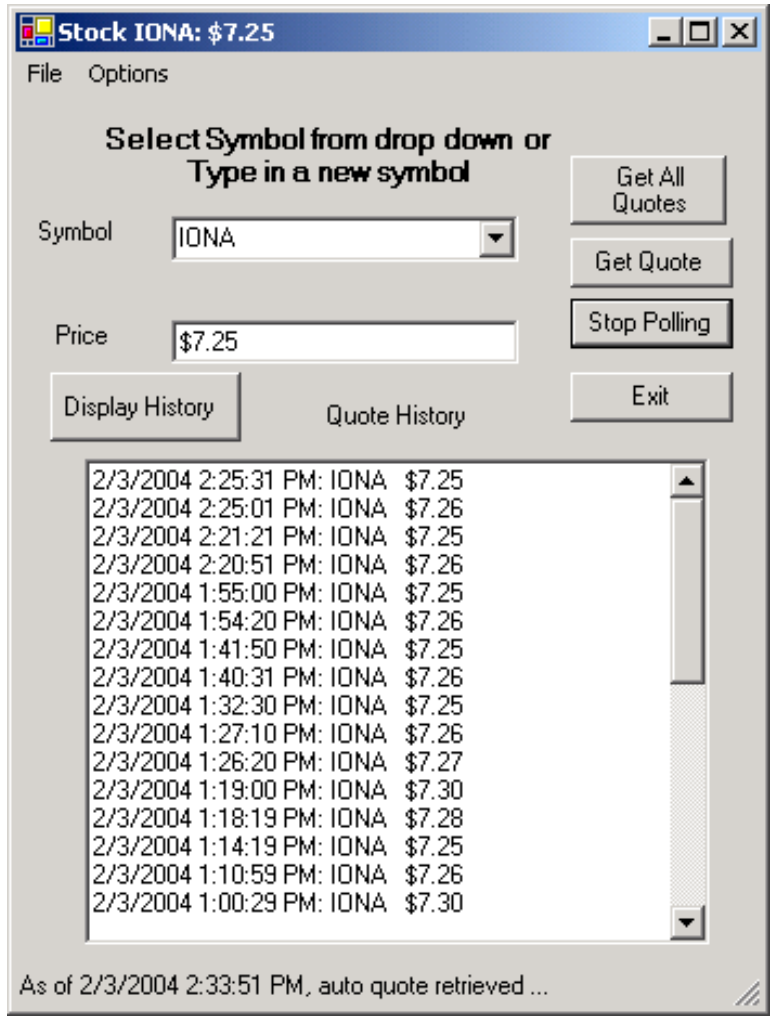
Service response



Web Service Server Log



A VB.NET Client to StockQuote Web Service



Writing Web Service Clients

Creating a Web Service from Language Constructs

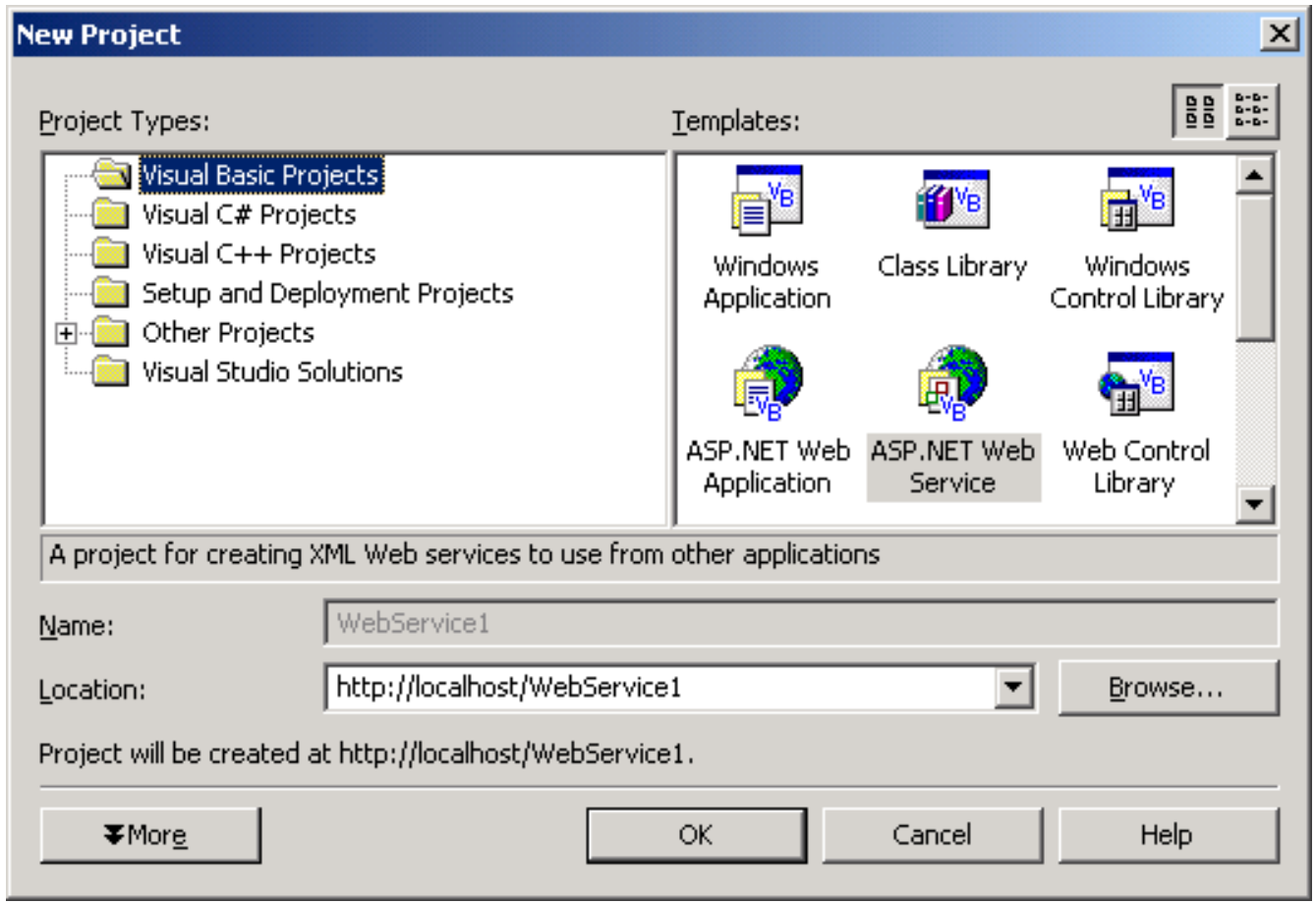
- Within most Web Service tools, it is easy to expose a function/method as a web service
- With VS.NET, we simply create an ASP.NET Web Service application, and add to any public function declaration(s) the **attribute:**

<WebMethod()>

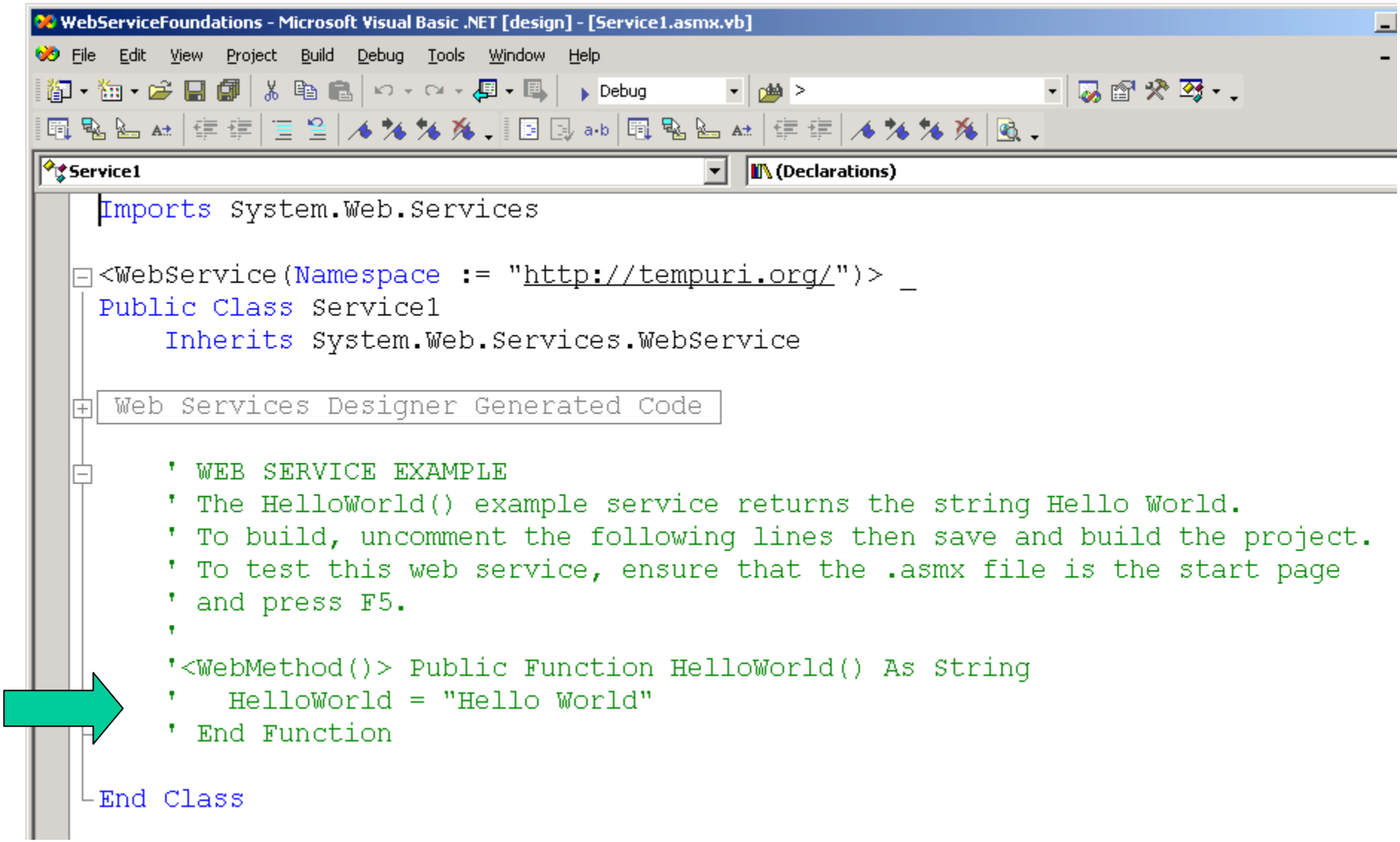
- This allows the .NET framework to supply the necessary WS aspects to expose the method as WS operation and invoke from a remote client

- Let's see simple illustrations of this using VB.NET …

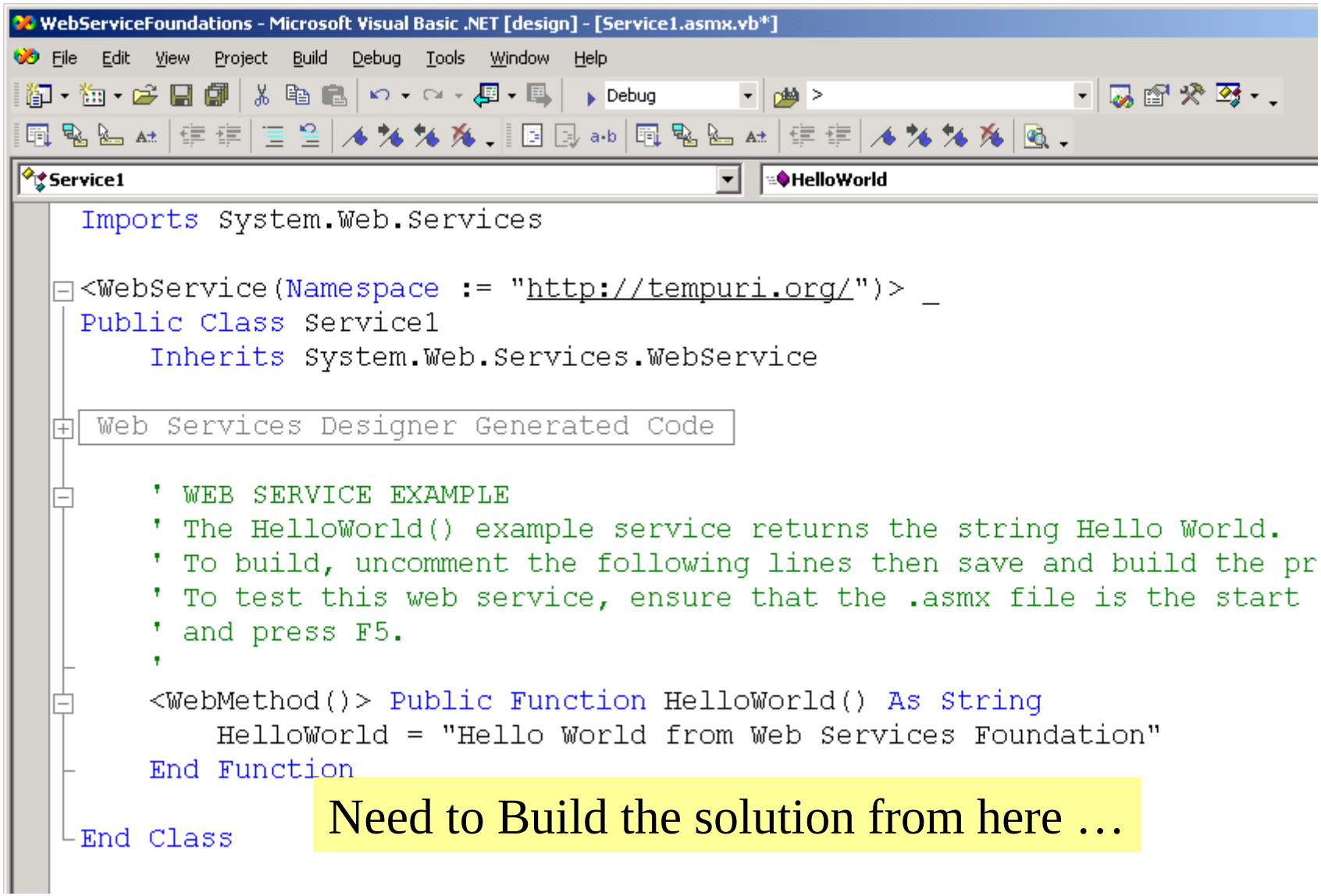
Create a Web Service Project …



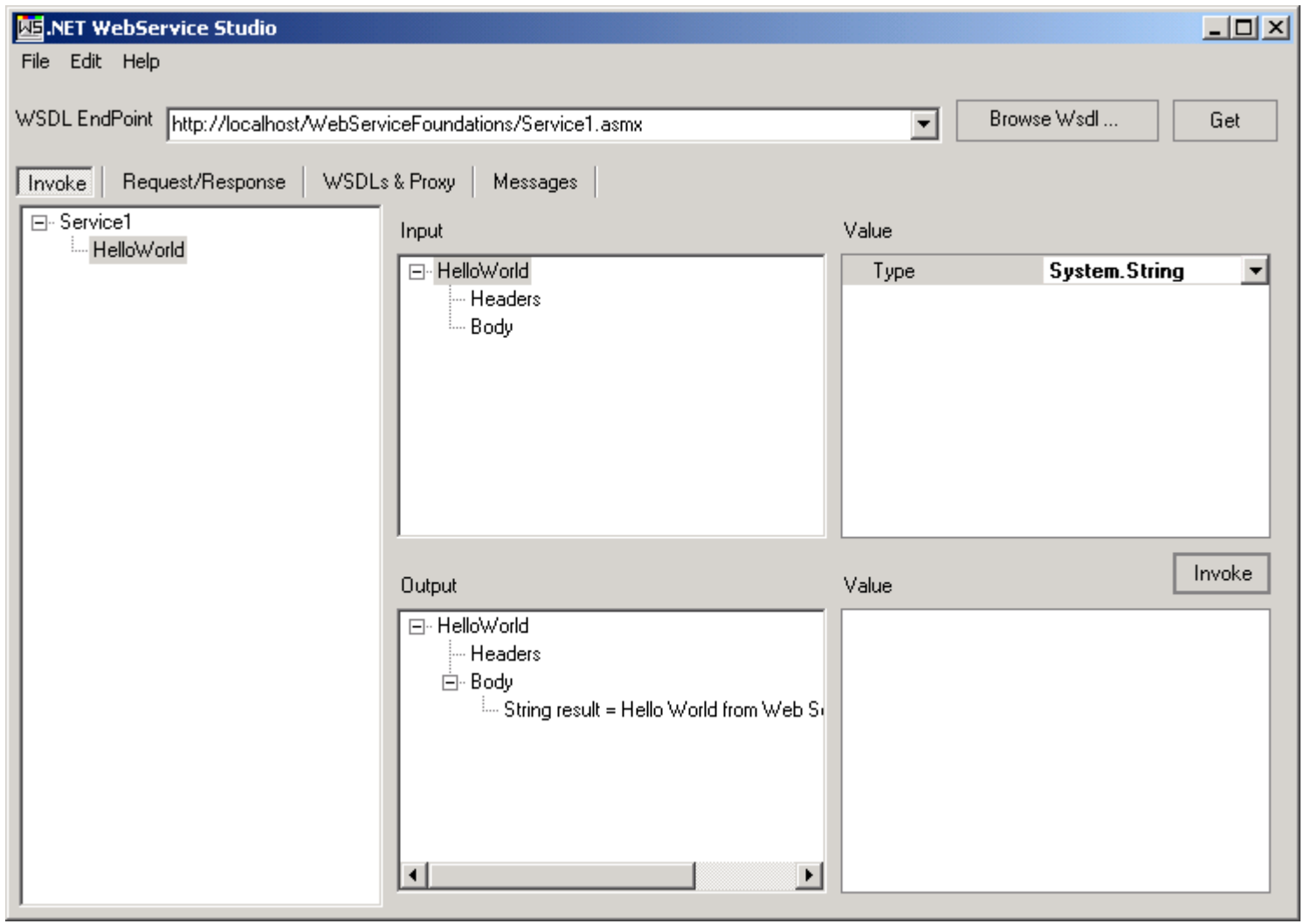
Generated Source File …



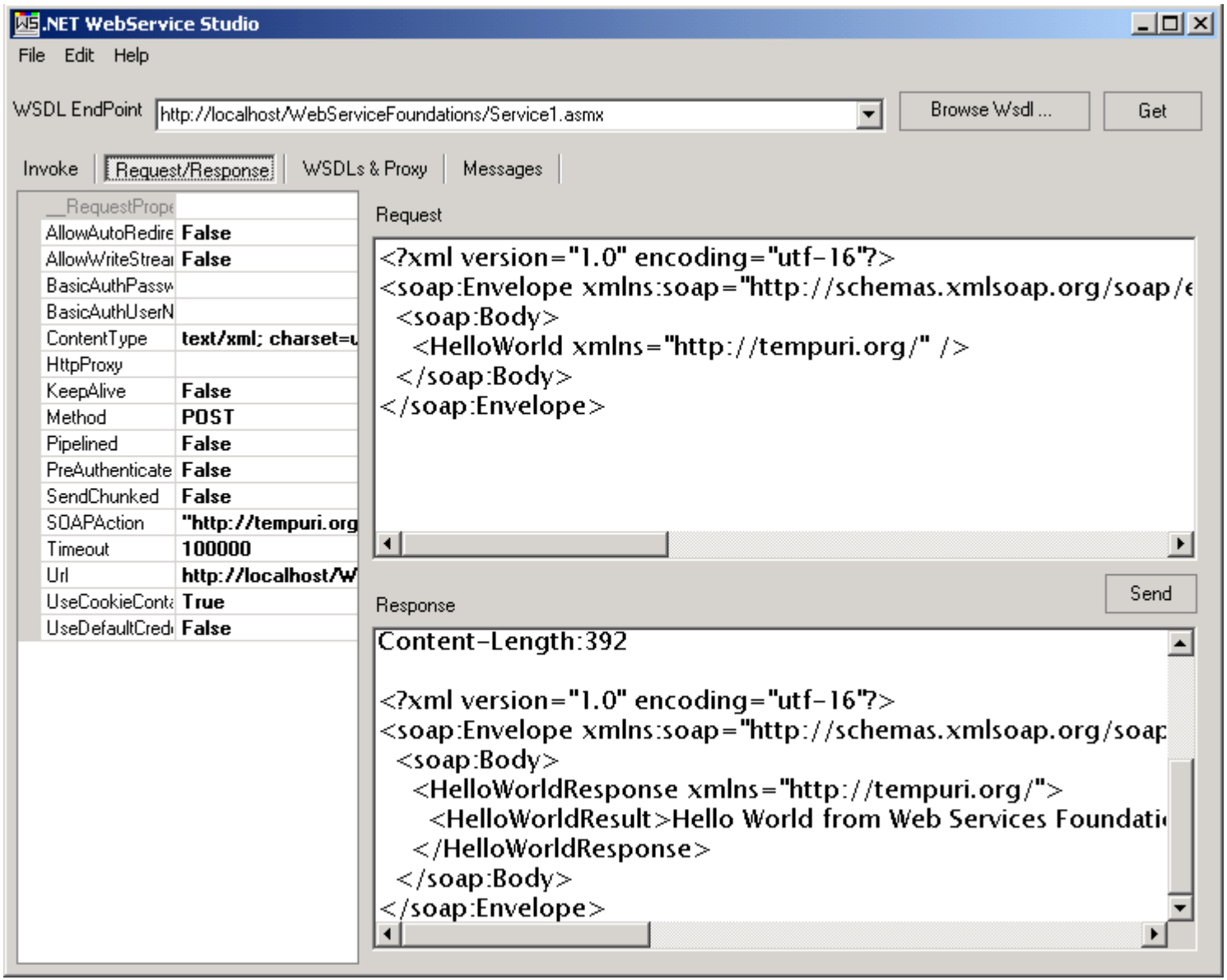
Add Code by removing comments



Using WS Studio Client – for SOAP



SOAP Messages to/from Server



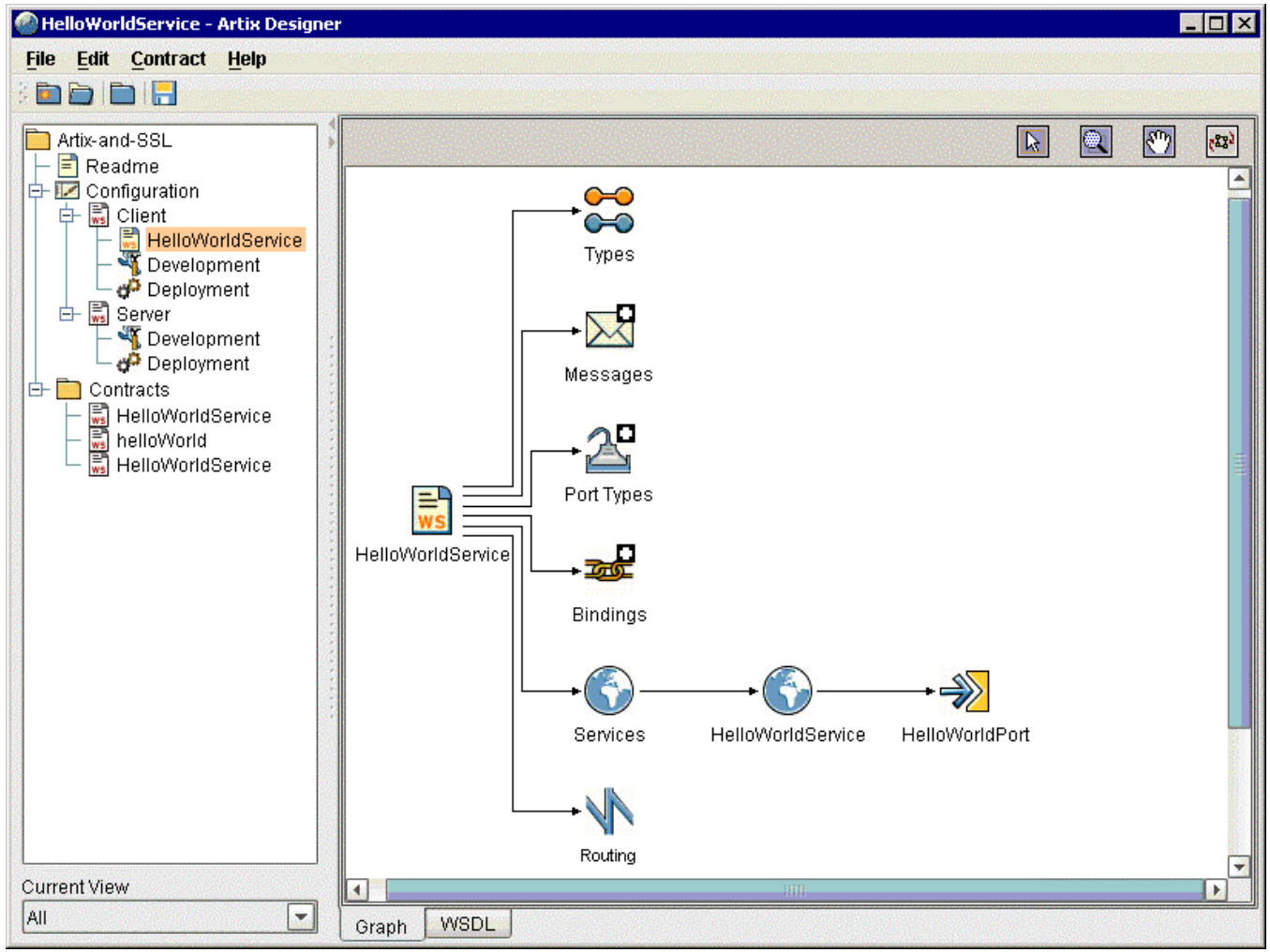
Server Code Generation

- Web Service ToolKits can also generate server side code skeletons directly from WSDL files
  - Java, C++, CORBA and other environments
  - Typically a collection of classes and interfaces
- Then the programmer must add the code to implement the operations (possibly implementing them using existing systems)
- Last step is to deploy the server side implementation into the web service container, to allow it to be invoked by remote clients
- No time to do here, but a later section describes how this was done for a Java-based environment (WSID)

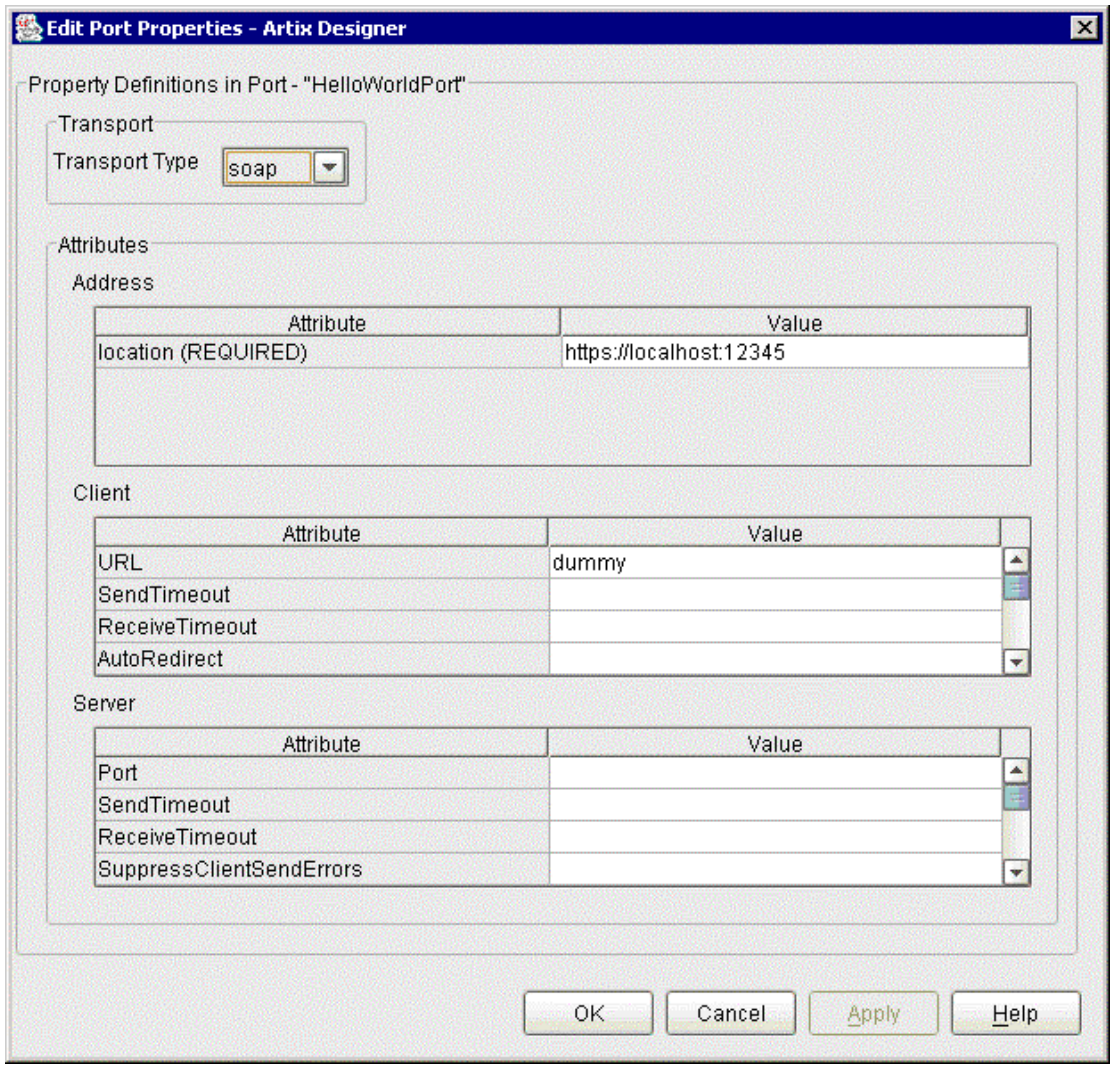
Tools to create WSDL files

- XMLSPY – somewhat generic XML support, but it can be made to work OK for WSDL
- WSDL support embedded in other tool sets
  - Microsoft Visual Studio
  - BEA's WebLogic, etc.
- Visually oriented (GUI) WSDL construction tools becoming available
  - See next few slides for a sample tool (IONA's Artix)

Top level WSDL structure



Detailed WSDL element wizard - Port



Summary of Development Demos

- These capabilities are available in most WS toolsets
- Simple to access remote service WSDL files to:
  - Create simplistic clients for immediate access/testing
  - Create simple client in programming language of choice
  - Create stubs for server components
- These tools mean that you don’t need to understand the details of WSDL
  - The tools handle the WSDL, and give you stub and skeleton functions that you can use to program your clients and services.

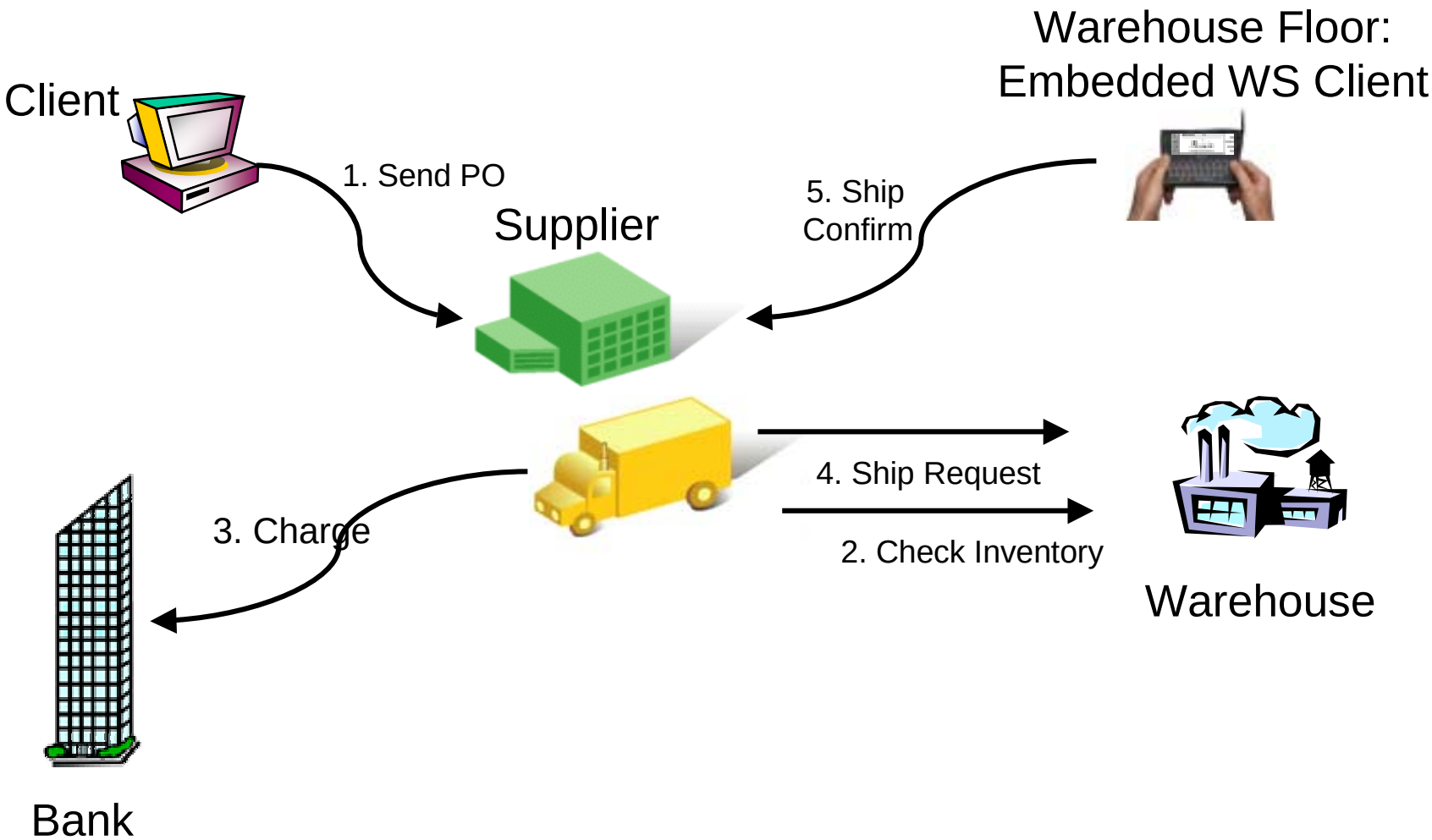
Web Services Execution Demo

Web Services Interoperability Demo (WSID)

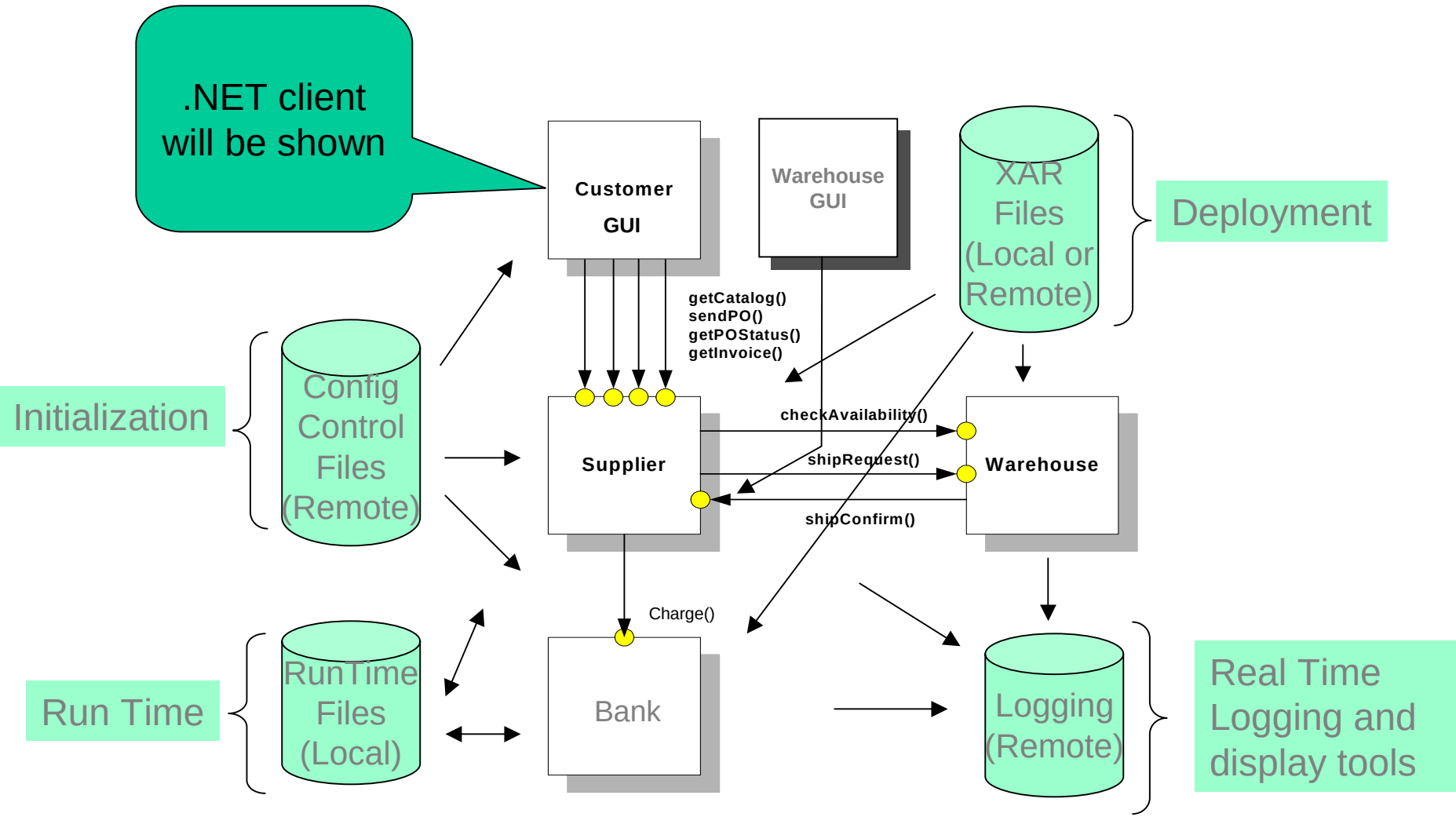
A Web Service Case Study

- Web Services Interoperability Demo (WSID)
- Group of industry vendors (IBM, Microsoft, IONA, The Mind Electric, Amberpoint, WebMethods …)
- Joint development of specs and implementations
- Development done globally, demonstrated at numerous trade shows (e.g., XML-ONE)
- Multiple platforms, OS, programming languages
  - .NET, several J2EE systems, straight Java, and some unknown!
- WSDL files as the key definition / specification for the project
- Tony Hong of XMETHODS.NET was the technical leader
  http://www.xmethods.net/wsid-03-2003/

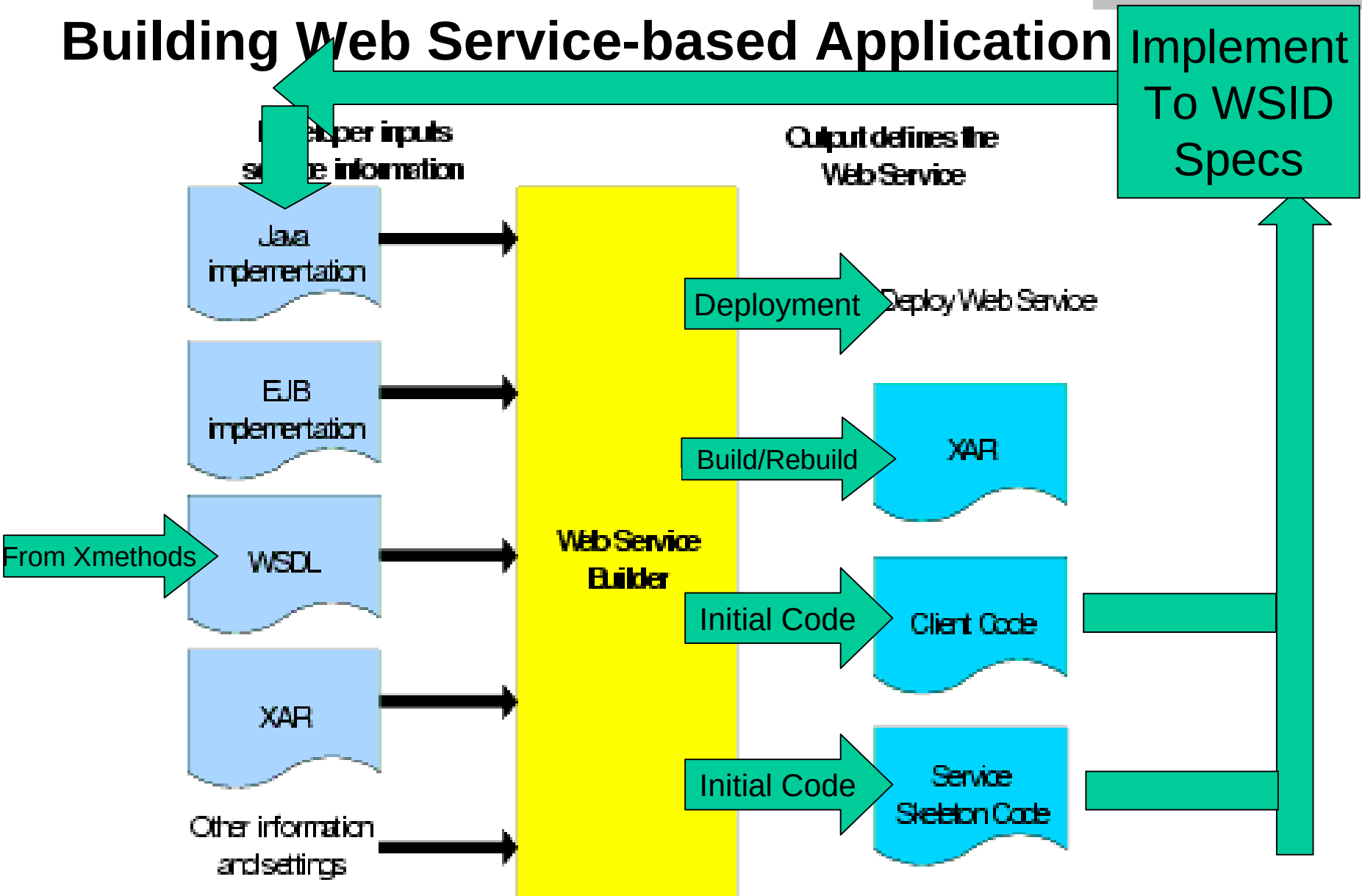
Web Services Interoperability Demo



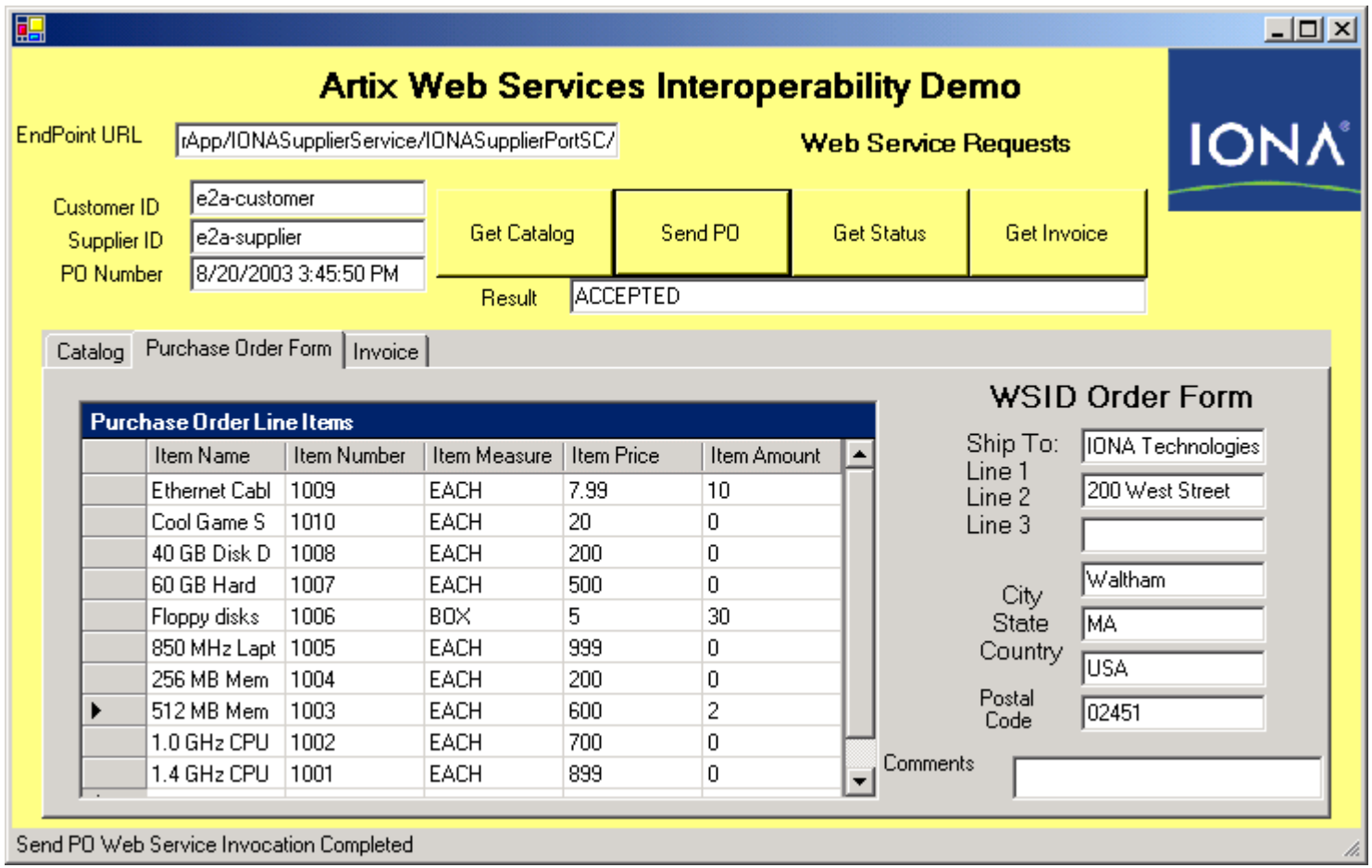
WSID Architecture



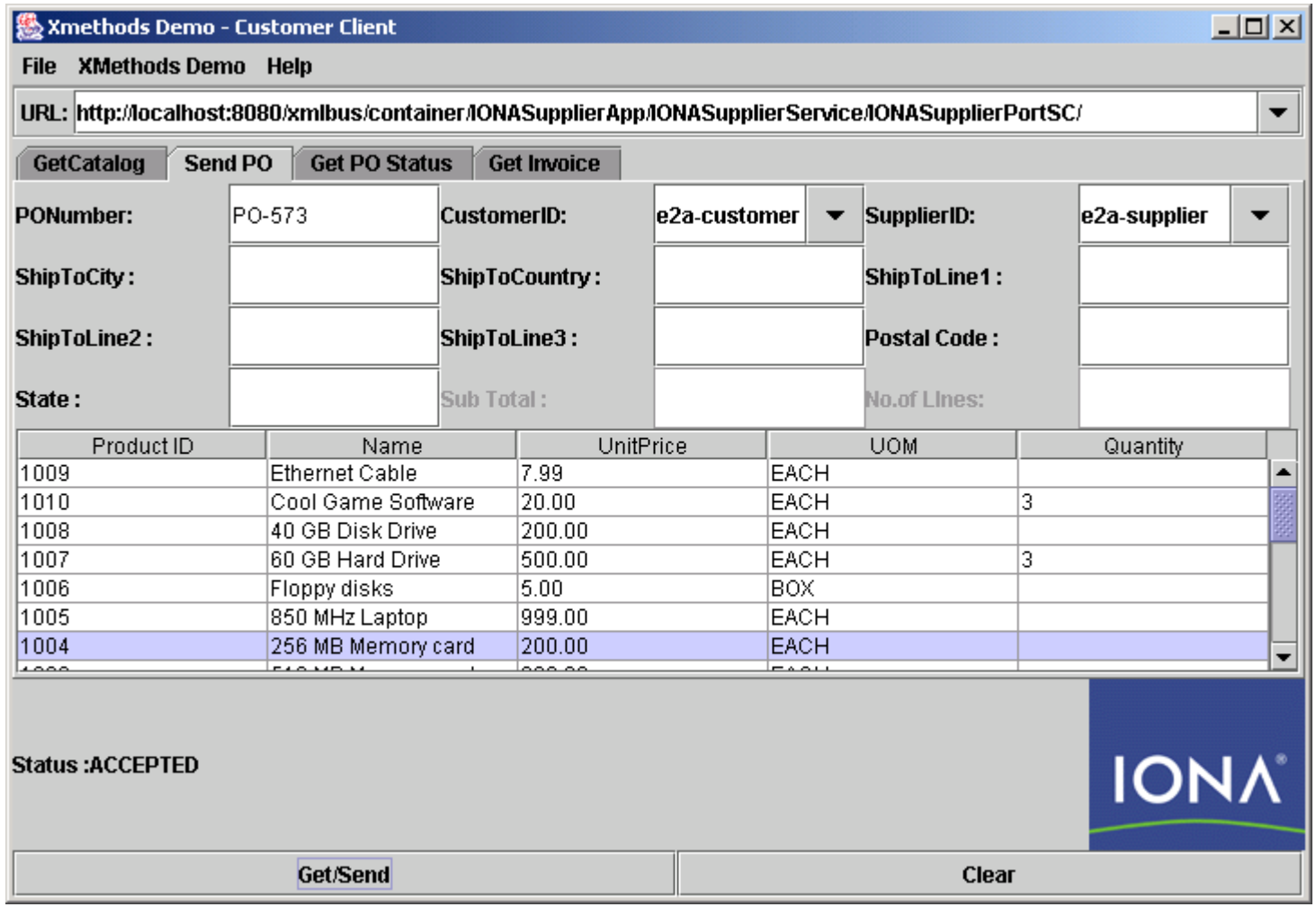
Building Web Service-based Application



.NET (VB) Client for WSID

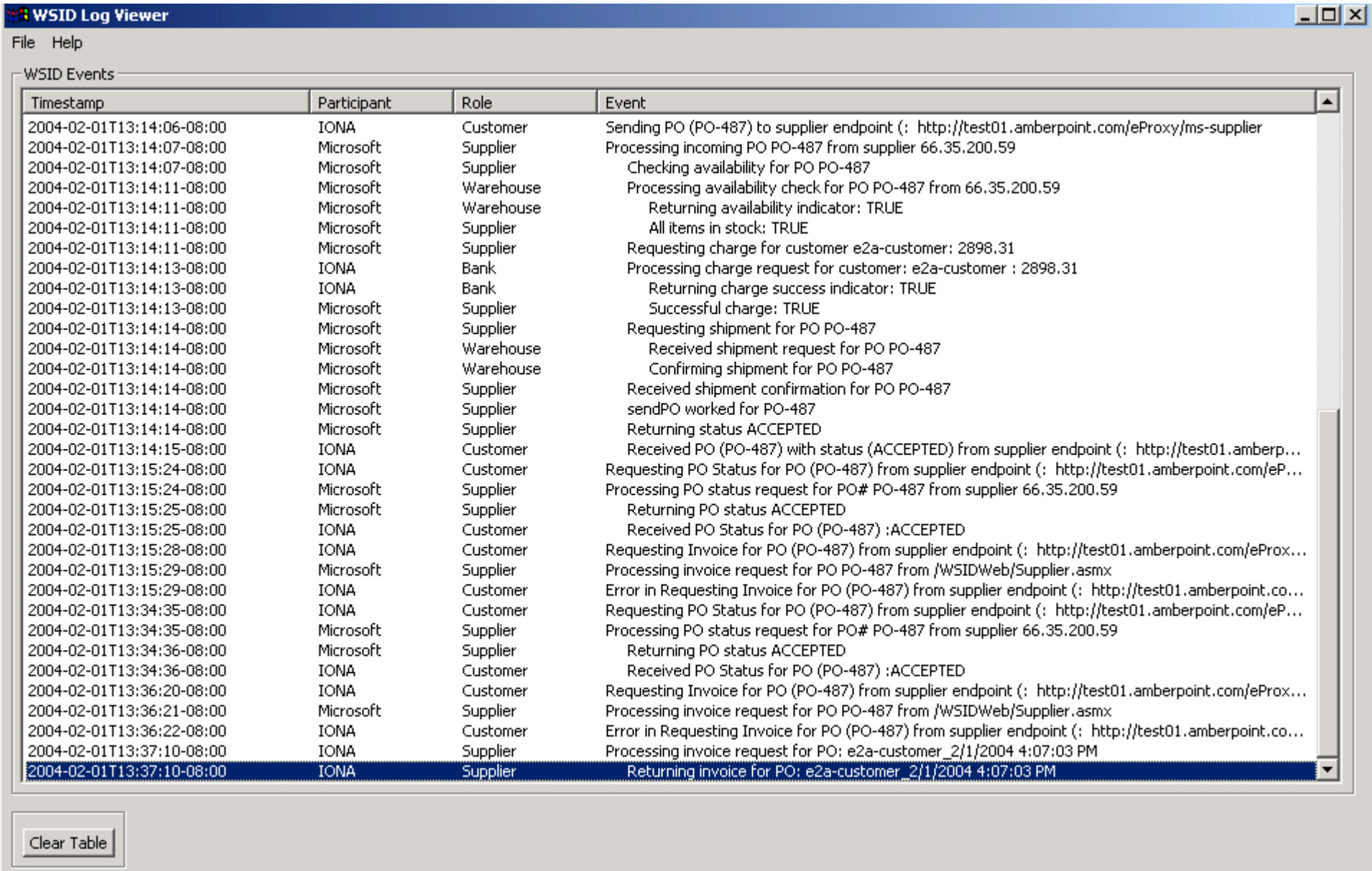


Java Swing Client for WSID

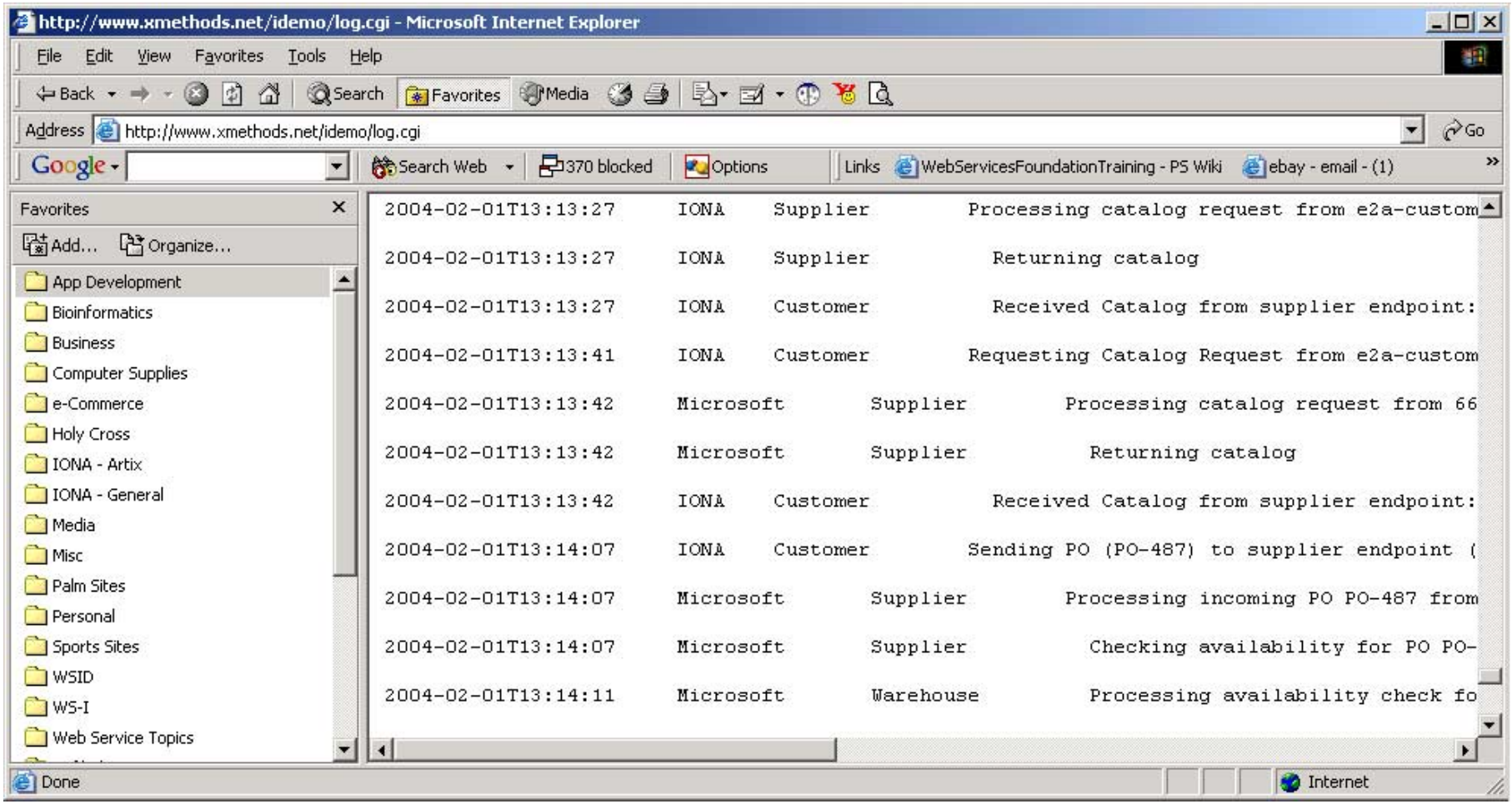


Let’s see WSID in action …

Logging screen shot



Browser view of logs



WSID Demo Summary

- What we saw in this demo …

- Multiple interacting web services
- Implementing a typical Supply Chain Example
- Multiple Language GUI Clients
- Web Service server also acting a Web Service client

- Each of these components has been implemented by a set of WS vendors
  - And it has been tested with arbitrary selections of these components, running on machines distributed across the US!

Chapter Summary

- Web Services are real – here and now!
  - You can design, develop and deploy with them now
  - Both for internal use and between enterprise vendors
  - Existing services can be discovered and accessed easily
- Many tools available with varying degrees of maturity and complexity – and more to come
  - Evolution of tool capabilities ultimately drives ease of use of WS
- The most interesting part of designing a system with Web Services is the high-level architecture of the system
  - What services should exist, and how should they interact?
  - What is the right structure for the information that should pass between clients and services?
  - These are the topics of the next chapter