DYNAMIC WEB SERVICE INVOCATION

THESIS

Presented to the Graduate Council
of Texas State University-San Marcos
in Partial Fulfillment
of the Requirements

for the Degree

Master of SCIENCE

by

Zhitong Zhao

San Marcos, Texas
December 2005

ACKNOWLEDGEMENTS

Many people have contributed, directly or indirectly, to the successful completion of this thesis. They will all be remembered in my heart. I would like to thank the following:

First, I would like to thank my advisor, Dr. Anne Hee Hiong Ngu for her excellent guidance from conducting the research to writing the thesis. I really appreciate the patience and respect that she has given me.

Second, I am extremely grateful to my other thesis committee members, Dr. Xiao Chen and Dr. Greg Hall, for being great sources of advice.

Finally, I would like to say "Thank you!" to my parents, Dr. Jianai Zhao, and Mrs. Jiaru Wang, for their constant help and encouragement during my thesis writing period, and Grace Chen, for comforting and encouraging me during this stressful time.

This manuscript was submitted on November 28th, 2005.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

DYNAMIC WEB SERVICE INVOCATION

by

ZHITONG ZHAO, B.S.

Texas State University – San Marcos, December 2005

SUPERVISING PROFESSOR: ANNE HEE HIONG NGU

With the relentless growth in Internet functionality, distributed computing systems have attracted more and more attention in the Information Technology world. This has resulted in the recent standardization effort of distributed computing architecture, which is known as Service Oriented Architecture (SOA). The Web Service is the centerpiece of this architecture. Some of the key challenges in implementing the SOA are maintainability, reliability, and security.

In this thesis, we propose to use the dynamic Web service invocation method to address maintainability and reliability issues without sacrificing the overall system performance. To achieve our goals, we proposed and implemented a Dynamic Web Service Invocation Framework (DWSIF). The dynamic invocation of Web services allows both service providers and service consumers to remain autonomous and maintain the loosely coupled relationship without scarifying the performance.

Through a series of experiments and objective evaluations, we have shown that dynamic web service invocation can serve its client better than static invocation, particularly in maintainability and reliability.

CHAPTER 1

INTRODUCTION

The relentless growth in Internet functionality and bandwidth have enabled a new wave of innovations that are transforming the way all types of organizations collaborate and interact with partners, both within and across organizational boundaries, forming a connected system. In the connected system, many independent pieces of software are able to connect with each other to accomplish a complicated task at a large scale, which can not be done by any one piece of the software in the system. Scientists and engineers have focused their efforts on automating software integration to meet the requirements for functionality, compatibility, and flexibility of packaged software. However, there are many challenges in building a connected system. Some of them, such as reliability, security, maintainability, and performance, are the typical challenges these systems are facing. These challenges must be carefully identified, evaluated, and solved before the integrated software packages are pushed into the market.

For example, the current research and development results showed that all the legacy B-to-B implementations, such as EDI (Webopedia, 2004) (Electronic Data Interchange), CORBA (OMG, 2005) (Common Object Request Broker Architecture), DCOM (Microsoft Inc., 2005) (Distributed Component Object Model), and Java RMI

(Sun Microsystems Inc, 2003), have weaknesses in maintainability, security, and cost of integration. Many software industrial leaders are looking for a new implementation system that can fulfill their B-to-B needs without these major weaknesses. With the development of the Internet and the XML technology, a Web service technology has currently emerged as a very promising replacement of the original B-to-B integration technology.

1.1 Current Stage of the Web service

What is Web service? Different people from different sectors of industry may have different answers. Some of them are

- Web services are loosely coupled software components delivered over standard Internet technologies (Cerami, 2002).
- A Web service is any piece of software that makes it available over the Internet and uses a standardized XML messaging system (Plummer, 2002).
- From a technical point of view, Web services can be seen as an application API that can be invoked by a Uniform Resource Locator (URL) (Bettag, 2005).
- The World Wide Web is more and more used for application-to-application communication. The programmatic interfaces made available are referred to as Web services (W3C, 2002).

According to these descriptions, we know that Web service leverages Internet infrastructure, and uses Web protocols, such as http, to communicate with its clients. Also, it provides a specific service requested by clients, such as locating an address,

looking for an issuances quote, etc. Like other services, a contract between the Web service provider and the service consumer must be set up, which both parties must follow in order to work together. Recently, Service Oriented Architecture (SOA) (Irani, 2002) was introduced. It sets the architecture standards for Web services and their clients. The connected system can be built based on the SOA. In the SOA, there are three main roles: a service provider, a service consumer, and a service broker (Systinet Inc., 2002). Unlike the OOA (Object Orientated Architecture), which is about reusable components encapsulated in one application, the SOA is about reusable applications encapsulated in one business domain, although they both can offer maintainability and reusability in some degree. Based on the SOA design, a subset of applications is able to connect with each other over the Internet, so that they are integrated into one super set system (Shan & Hua, 2005). The Web service is, therefore, a corner stone of SOA.

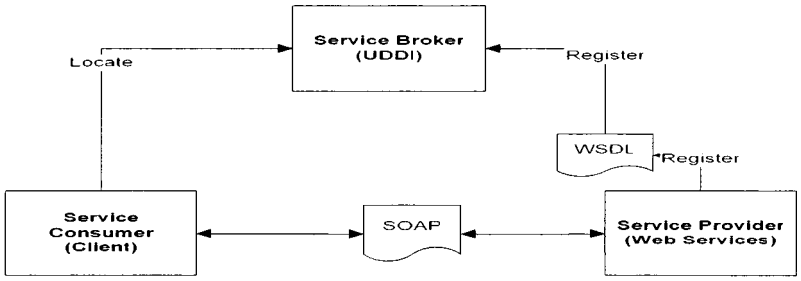

Figure 1-1 SOA Roles diagram

Under the operation of the SOA (See Figure 1-1), a service provider will register its Web service with a service broker. On the other hand, a service consumer will locate a service through this broker, and then subscribe to this service. The co-relations among these three components are listed below:

- Service Provider – It provides the Web service that service consumer is looking for subscribing. It publishes its services to a service broker.
- Service Broker – It stores a provider's Web service information such as the service location or the URI and the description information about this service. Based on these stored data, it can locate the Web service for clients.
- Service Consumer – It is service provider's client. After a broker locates a service, it will return a service reference to a service consumer, which can invoke this service directly.

Like other B-to-B application architectures, the current SOA also has to perform three key functions: discovery, description, and transportation. These key functions are implemented based on three main standards: UDDI, WSDL, and SOAP. They are briefly described below:

- UDDI – Universal Discovery, Description, and Integration, is a repository of the Web services. It provides a registry service for a service provider to publish its services. It also provides the API for locating and binding services for a customer to look for a specific service. Currently, IBM and Microsoft maintain two public domain UDDI registries.
- WSDL – Web service Definition Language, is an XML document that describes the Web service information, such as methods, parameters, and addresses. A client needs to build a proxy based on this XML document statically or dynamically if it wants to consume this service. The proxy can be viewed as a contract between a provider and its client. Once a client builds a proxy based on the WSDL, both parties need to honor the contract. Otherwise,

client invocation process will fail. The W3C maintains the WSDL specification.

- SOAP – Simple Object Access Protocol is a lightweight distributed transport protocol that allows the Web service and its client to exchange the information. In other words, a client sends a SOAP request to ask for a service, and then the Web service will receive and process this request, and send the response to this requestor through a SOAP response. The SOAP message is in XML format and transported using HTTP, or HTTPS Internet protocol. Microsoft is the original creator of the SOAP specification and implementation. Currently, the W3C maintains the SOAP specification (Werner, 2005).

Since the SOAP message and the WSDL document are in XML format, the Web service is self-describing and self-extendable. Also, since the Web service is using Internet protocol such as HTTP, it is a cross platform application. Most modern Web servers can host the Web service. Some examples are Apache, Weblogic, and IIS. J2EE and .Net are the two primary development environments for supporting the Web service.

1.2 The Challenges of the SOA

Since the Web service built on the SOA is going to replace the legacy B-to-B integration infrastructures such as DCOM, CORBA, etc., it has to achieve the following design goals:

- Performance

  Because the Web service has to handle exchanging a large amount of data between a service provider and its clients over the Internet, the performance challenge has to be addressed.

- Scalability

  Because the Web service has to be able to handle from a few to hundreds of thousands of requests from its clients in seconds, its scalability has to be addressed.

- Security

  The Web service operates on the Internet. The Web service and its client systems can exchange very sensitive data such as the financial and personal credit information. Security is, therefore, a major concern among service providers and clients. (Yang, 2004)

- Reliability

  Since Web service can serve clients around the world to complete a crucial business transaction, 24 x 7 up time is often required. Therefore reliability is extremely important.

- Interoperability

  Since the Web service and its client systems may be developed using different programming languages and running on different platforms, its compatibility and tolerance must be carefully tested and evaluated. Therefore, its interoperability is one of the primary goals from day one.

- Maintainability

  Software maintenance tasks are always expensive. The manual nature of operations was requiring some business to spend as much as 70 to 80 percent of their IT budget on maintaining their existing system (Morris & Chong, 2005). The Web service and their client systems may belong to different owners. This nature makes a connected system even harder to maintain. How to maintain their life cycles without breaking the related contract is attracting more and more attention among the scientists and engineers in the fields of the Web service, the Client system, and the Service broker.

Fortunately, some issues are already addressed, some are being addressed along the right direction, and some are at the advanced assessment step for their applications.

- Performance

  Because Web services are hosted by a Web server, their performance is significantly affected by the Web server performance. Almost all the major players, that include Microsoft, IBM, Oracle, BEA, Sun, and Apache Open Source, incorporate Web service into their latest version of the Web server products. And they are even making some enhancements to boost Web service performance, such as shortening the response time, supporting multithreading unsynchronized call, etc.

- Scalability

  Because the Web service is built on a multi-tier architecture, and runs on a Web server, it is scalable, just like other multi-tier Web applications that are running on a Web server.

- Security

  The Web service can rely on Web server security settings. It supports HTTPS. OASIS has currently maintained security specifications (OASIS, 2004). The industry leaders, such as Microsoft, Sun, and IBM have implemented these specifications on their latest Web server products. Microsoft just released Web service Enhancement 3.0 libraries to improve the Web service security (Brown, 2005).

- Reliability

  Since Web services run on Web servers, they can be treated as the mission critical Web applications and the reliability can be further assured by deploying them to clustered Web servers. The only issue left is what if the entire site is down?

- Interoperability

  Since the Web service and its client systems communicate through the standardized SOAP message - an XML document, the Web service must be able to process a strongly typed client request without knowing the client system's programming language, and vice versa. Maintaining the interoperability is an ongoing effort. In fact, most primitive types including integer, short, string, char, and float, in WSDL are highly interoperable. But,

if one language adds a new data type, it requires the other languages also to support this data type, or map it to its own data type.

- Maintainability

  Currently, as long as the Web service does not change its interface, clients can always invoke it from anywhere without changing any of its client codes. But if the method signature in a Web service is changed, such as adding or removing method parameters, its clients have to manually change their codes to adapt to these changes in the service interface. Otherwise, the client applications will fail.

1.3 Motivation

The SOA shows us a new and effective approach for building connected system, so that an integrated software package constructed by a group of software components that connect with each other can accomplish a complicated task at a large scale, that can not achieve by any one single piec of the software. The Web service is the corner stone of the SOA implementation.

Since it is still a relative new technology, quite a few aspects must be improved in the near future. A lot of efforts have been made to overcome the difficulties mentioned above directly or indirectly with different technical orientations. Microsoft has worked on Windows Communication Foundation (formerly code-named "Indigo") for a while (Swartz, 2005). It is a set of .Net technologies that provide Web service with secure, reliable, and transacted messaging along with interoperability. Computer Associate released a Web service monitoring tool, “Web Services

Distributed Management" (CA Inc., 2005), which can monitor Web service message exchange in real-time and log all critical service level metrics for subsequent analysis and business process automation.

There are still a number of key challenges. Since the relationship between a Web service and its client system is usually loosely coupled, this means that they are interacting with each other only at client invocation moment. During this crucial stage, the service provider has to make sure that the service has high reliability and low maintainability. Otherwise, Web service can not fully demonstrate its true value.

1.4 The Proposed Solution

In this thesis, we propose a mechanism that will address the reliability and maintainability issues in Web service without sacrificing the overall performance. We extend the Service Broker capability in the current SOA (See Figure 1-1), and add a dynamic invoker for Web service consumers.

Our service broker acts as a Web service registry. It is needed for invoking a Web service, which changes its interface definition frequently. It records the additional information, which is not supported by the current UDDI or WSDL. An innovative dynamic invoker will be designed to consume the WSDL and the additional information from our extended service broker registry. Whenever there is a change in the Web service such as its interface, our dynamic invoker will build a new client proxy to reflect this change at run time; this is similar to a contract fixer. During the client invocation process, our dynamic invoker will absorb the changes, and still provides the client previous interface. This dynamic invoker also caches the

proxy and saves these service changes as a profiler in the client side. This reduces the requirement for recreating the client proxy at each invocation. A new proxy is only required when the Web service undergoes another change.

In our solution, the service broker is crucial in the entire architecture. It will be safe and reliable that the service consumer will know any changes in the Web service interface to which the service provider makes, and will know what the backup of the Web service is, etc. Therefore, a change control procedure related to these changes will be implemented between each service provider and its consumers. As a result, a service provider is free to make any necessary changes to fulfill the rapid business needs without any impacting events on its client system. Similarly, the client will never have to worry about its provider changing or relocating the service. A piece of identical client code will run regardless of how the service interfaces have changed. Under the proposed solution, the Web service will be safer, faster, more reliable, and cost lower for building the connected system.

1.5 Thesis Structure

The thesis is organized as follows: In Chapter 2, we will discuss the current SOA, its life cycle, and the challenging issues. In Chapter 3, we will discuss the process of the dynamic invocation by using .Net and J2EE platform, and the related issues, and, our dynamic Web service Invocation process, which is the solution for some of the key issues that the Web service is facing. In Chapter 4, we will present the "Dynamic Web Service Invocation Framework prototype" in detail. In Chapter 5, a few important concluding remarks of the thesis and a brief discussion about

possible future works will be presented. The necessary data and prototype system source code is presented in the Appendix.

CHAPTER 2

THE RELATED WORKS

2.1 Background

In order to build an effective connected system, scientists and engineers have focused their efforts on understanding the interaction behaviors among the individual component. In the field of the Web service, the interaction between the Web service and the surrounding software environment was well studied and a number of new techniques were proposed. In the distributed computing system, one of the basic processes - a client application interacts with the Web service – is recognized as the process through an invocation (Fang, 2005). Either a static technique or a dynamic technique can be applied to a web service invocation. We will discuss the existing static and dynamic techniques for invoking services.

2.1.1 Static Invocation

In static invocation, the clients generate proxy stubs during development time, using the WSDL of the Web service (Oracle Inc, 2005). When it is compiled to the executable application, the proxy will be generated also (see Figure 2-1-a). During its execution time, the client invokes a method of this Web service through this proxy (see Figure 2-1-b).

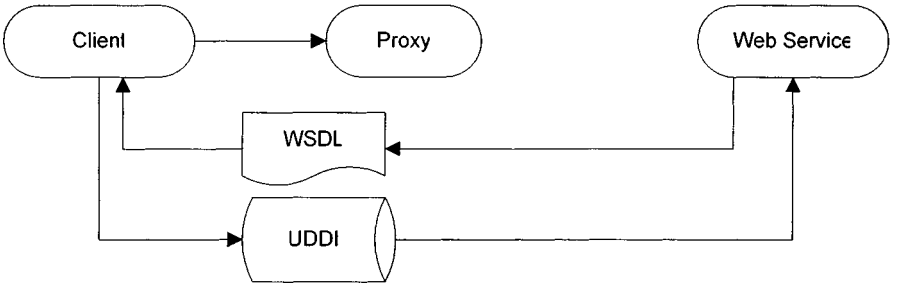

Figure 2-1-a Static Invocation at design time

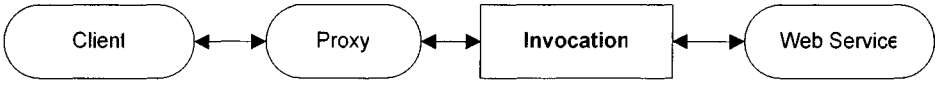

Figure 2-1--b Static Invocation at run time

As we can see from the two figures, by using a static invocation, a client program locates and validates its Web service, and builds the service proxy during its compilation time. During the run time, since the proxy is already in executable form, the client invocation process does not require much processing power.

Because the service proxy was built at design time, there are some limitations. The proxy is not reusable for the same service in different locations, or protocols, etc. Also, the proxy will break, if its service interface is changed.

2.1.2 Dynamic Invocation

Unlike the static invocation, during the dynamic invocation, a client application creates its service proxy at run time by parsing the Web service WSDL document (Oracle Inc, 2005). Then it invokes a method of this web service through this proxy. This process flow is schematically shown in Figure 2-2-b. During its design time, the client does not need to know the Web service at all. This principle is summarized in Figure 2-2-a.

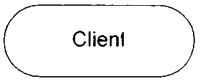

Figure 2-2-a Dynamic Invocation at design time

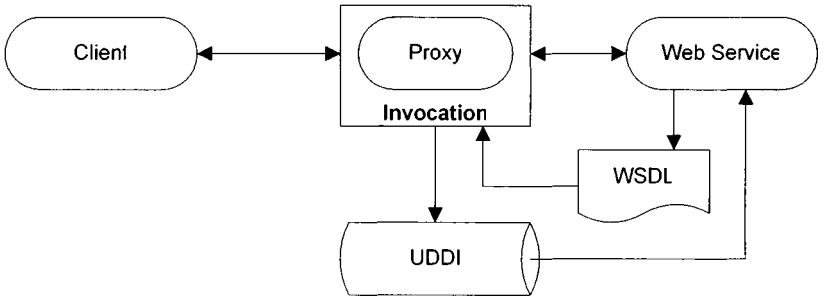

Figure 2-2-b Dynamic Invocation at run time

By using a dynamic invocation, the design task of a client application is simpler than that by using a static invocation. But the client application does not know if the Web service is valid or not. It leaves the locating and validating tasks to its run time invocation process. During the period of its run time, before it invokes its web service, the client application will locate and validate the web service, then build the proxy, and finally invoke the web service through this proxy.

However, using dynamic invocation technique will experience a performance penalty in the process, but it is far more flexible then the static one during its invocation process. A client can use the same code to access a group of Web services with the same interfaces. This will dramatically reduce the cost of testing and evaluating a group of Web services.

Because of its flexibility, a few research groups are trying to utilize this feature to solve some challenges in the Web service.

2.2 Current Research Prototypes

Three research prototypes on the Web service invocation already proved some advantages of dynamically invoking the Web service. We will introduce these research prototypes in the following sections.

2.2.1 Web Service Invocation Framework (WSIF)

The WSIF was originated in IBM, and it has been a part of the IBM Apache Open Source project. Bulit using Java technology, and based on the WSDL4J API, the WSIF is a programmatic API for invoking a Web service (IBM Corp. 2005).

The framework allows stub-less and completely dynamic invocation of a Web service at each invoking instance, based on the WSDL document of the service. It also allows updated implementations of a binding to be plugged into the WSIF at runtime, and the calling service to defer choosing a binding until runtime. Based on this consideration, the service consumer does not care about the binding type, which only its service provider requires only. Therefore, the related design is simplified.

A client can follow a general procedure to invoke a service dynamically in the WSIF. It is briefly described below.

- Step 1 - The WSDL document is read into a javax.wsdl.Definition object.
- Step 2 - By using this object, a WSIFDynamicPortFactory object is created. All type mapping between the XML types declared in the

WSDL and the Java classes used in the client are defined in this factory object.

- Step 3 - A WSIFPort object is created from this factory object.
- Step 4 - Both the input and output messages and parts from WSDL are wrapped in WSIFMessage and WSIFPart objects.
- Step 5 - After WSIFPort.executeRequestResponseOperation () is called to invoke the Web service, the result will be returned from output message of WSIFPort.

Figure 2-3 shows a sample client Java program that use WSIF to invoke a web service dynamically.

```
Definition def = WSDLReader.readWSDL(null, new
                    InputSource("http://www.PeopleInfo.com/addressinfo.jws?wsdl");

WSDLDynamicPortFactory dpf = new WSDLDynamicPortFactory(def);

dpf.mapType(new QName("http:// www.PeopleInfo.com/schemas/AddressRemoteInterface",
"Address"), Address.class)

WSIFPort port = dpf.getPort();

WSIFMessage imsg = port.createInputMessage();
WSIFMessage omsg = port.createOutputMessage();
WSIFMessage fault = port.createFaultMessage();

WSIFPart ipart = new WSIFJavaPart(java.lang.String.class, "James Cook");

Imsg.setPart("person", inputPart);


port.excuteRequestResponseOperation("getAddress", imsg, omsg, fault);


WSIFPart opart = omsg.getPart("result");
Address laddress = (Address)opart.getJavaValue();
System.out.println("Street : " + laddress.street);
```

Figure 2-3 using the WSIF to invoke a Web service

As we can see, the WSIF demonstrates a few advantages, when a service consumer invokes a Web service dynamically rather than statically. One of them is that by using a dynamic invoker, a client does not need to recompile and redeploy its software applications each time, when its subscribed Web service changes its protocols, or locations. Another one is that when a client tries to test or evaluate a few Web services, which have identical interfaces, the client can reuse one piece of code (See Figure 2-3) for all these Web services.

But the WSIF also has a few disadvantages based on the analysis of its implementation. First, since it generates a proxy at each time when a client invokes a Web service, there will be a performance penalty. Second, in general, the Web service interface is changed frequently because of the dynamic business environment. But the WSIF can not handle the interface changes. For example, if a Web service method adds or drops its parameters, it will require the client to change its code to adapt this change. Third, if a client needs to use an object that the Web service has, the class of the object also has to be defined in the WSIF. These disadvantages make the WSIF weakly usable under the current SOA environment.

2.2.2 Web Service Adapter

Dr. Mark M. Davydov just recently unleashed a Web Service Adapter design in his article "Ease Web services Invocation with Dynamic Decoupling" (Davydov, 2005). This use of the Web Service Adapter is closer to our design. However, it is still a conceptual model with no clear implementation.

This Web Service Adapter provides a set of classes that make up a reusable adapter layer to the Web services consumer application. This adapter layer encapsulates code that uses the stub and its generated classes. The public API of the adapter does not expose any of the stub classes; it instead maps them to classes that the Web service consumer application understands. As a result, no matter what changes a provider makes to its service, the client does not have to do

any changes if the adapter can handle these changes. And, its code will still run functionally in this situation.

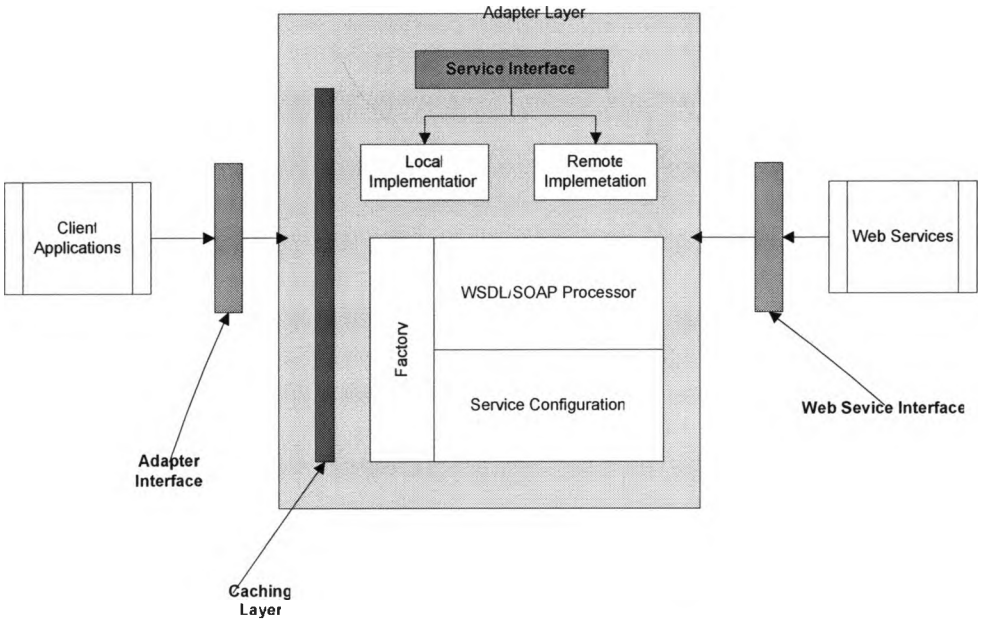

Figure 2-4-a Adapter Conceptual Model

The Client applications and Web services access the adapters through interfaces shown in Figure 2-4-a. The typical adapter operation sequence is shown in Figure 2-4-b. From these two diagrams, we can find some important process step features in this application.

The Adapter should work like Figure 2-4-b.

For a service provider:

Step 1 - Provider describes the adapter service interface to publish its new or updated Web services.

For a service consumer:

Step 1 - Consumer calls the adapter interface to invoke a Web service.

Step 2 - Adapter service interface calls the adapter factory. Depending on the information from its service configurator, the factory encapsulates the

knowledge of which implementation, local or remote. It has to be used to access.

Step 3 - If the factory decides to use local implementation, the factory will let the consumer make a local invocation to the Web service through the adapter service interface. Otherwise, the factory will make a remote invocation to the Web service through the adapter service interface. During the factory's invocation process, the factory processor will dynamically build a service proxy for the consumer to invoke the service.

Step 4 - The consumer will get the results from the service provider through the adapter service interface.

Reviewing the Davydov's design, we can find that the author does give a special consideration for the performance impacts because of the nature of dynamic invocation. Therefore, a caching layer is considered (See Figure 2-4-a).

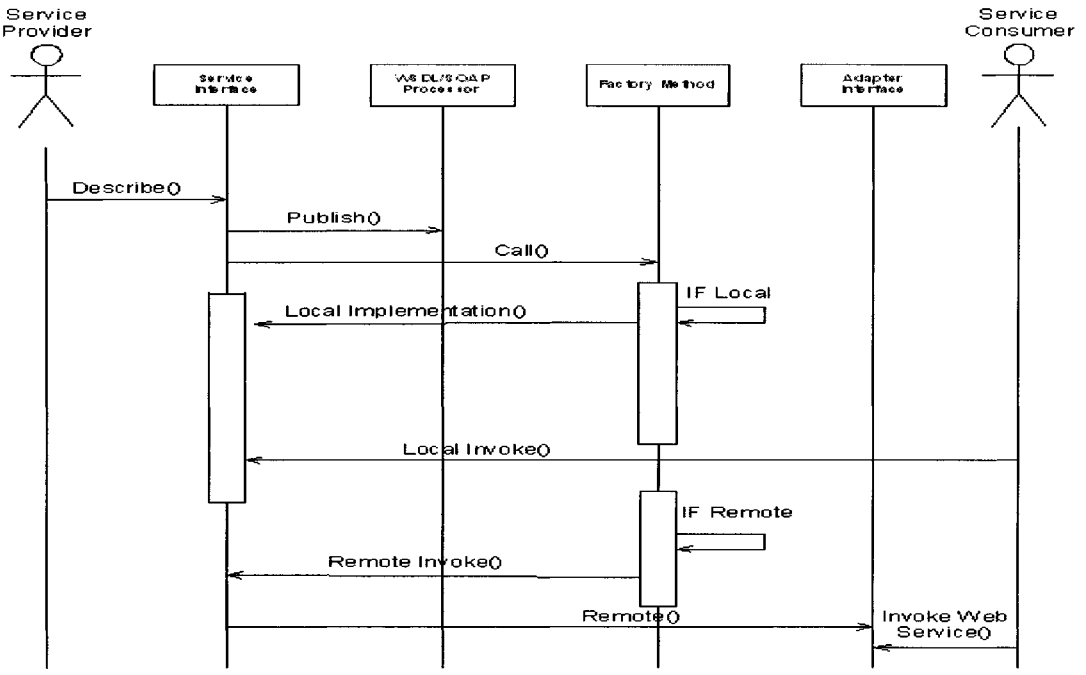

Figure 2-4-b Adapter Sequence Diagram (Davydov, 2005)

By using the adapter approach, the Web services invocation code becomes highly reusable and configurable. Because Web services invocations are all channeled through a common adapter in which service descriptors are deployed, one could decide dynamically what service is invoked -- doing this in the deployed code or even at run-time.

Since this design is at an early stage, there are a few unclear and premature specifications. Firstly, when a consumer invokes a Web service through this adapter interface, the Factory will judge whether the process is a local invocation or a remote invocation based on the information the Factory received from the Service configurator. But where does the information come from? And, how reliable is it? Secondly, how does one implement the caching wrapper, what should be cached, and what should not? When does the caching

information need to be refreshed? If the adapter caches frequently changed variables, such as returned function results, it could waste a lot of memory space for invoking a high volume Web service, since the results could be frequently changed. Thirdly, the adapter only considers handling the Web service changes and gives the service consumer an unchanged interface, so that the consumer code can remain the same. But what if the consumer decides to make the changes to adapt to the Web service changes permanently? The adapter may break the client application, according to this design.

2.2.3 The DynWsLib

The DynWsLib is a .Net library to dynamically call Web services at runtime. It is developed and maintained by Thinktecture's Christian Weyer (Weyer, 2004). Like the other two solutions mentioned above, it creates proxy based on the service's WSDL at run time, so that a client application can use it to invoke a method of the Web service.

Since it is developed for testing or evaluating Web services, it supports more invocation features, such as asynchronous invocation, user credentials, communication timeout, and multiple bindings. Also, it does cache the Web service proxy for improving the performance of the repeated invocations. This library is easily incorporated into a testing application. Figure 2-5 shows a simple Web service testing tool, which uses the DynWsLib to call a Web service.

However, DynWsLib does not address the maintainability and reliability issues that we discussed in the Chapter 1. Due to the use of dynamic invocation

and its many invocation features, DynWsLib allows a client to test a Web service comprehensively by using an identical testing environment and a testing code. But, since it lacks the ability of tracking Web service changes, using a cached proxy to invoke a changed Web service could break the invocation, even the Web service is completely functional. One has to clear the cached proxy and let DynWsLib regenerate one at run time to reflect the changes.

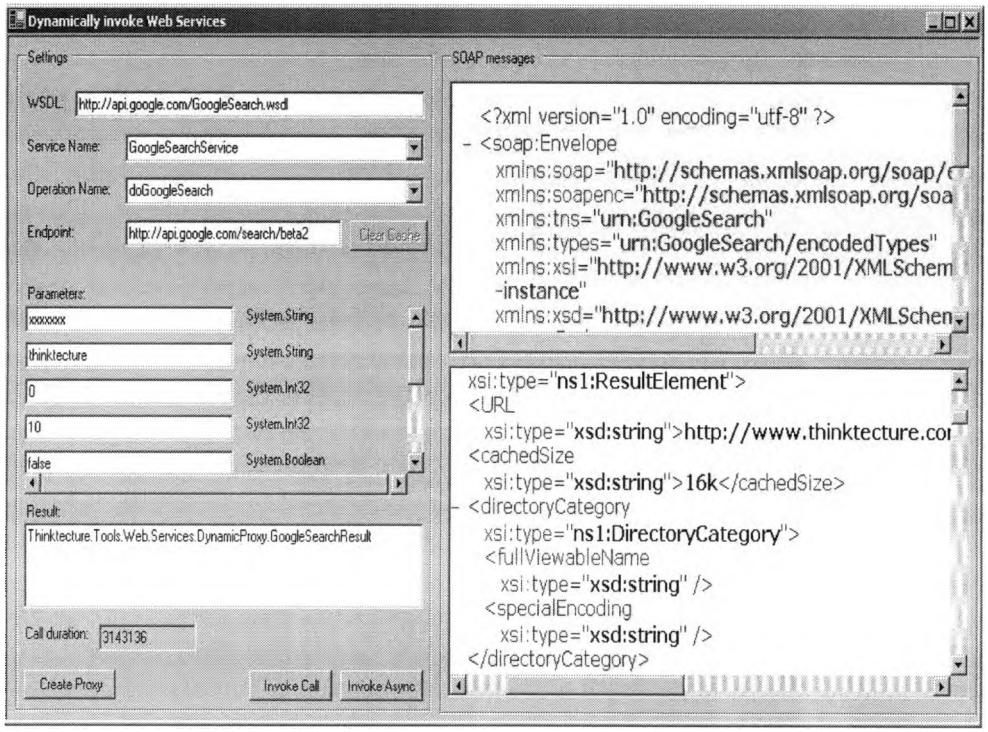

Figure 2-5 A sample testing application powered by DynWsLib

2.3 Summary

All the above prototypes recognize the importance of dynamic Web service invocation. And they clearly show the advantages of dynamic invocation versus static

invocation. But the weaknesses of these two prototypes still prevent people from fully trusting the dynamic invocation to be the best way to invoke a Web service under the SOA. Our solution, the Dynamic Web service Invocation Framework (DWSIF), not only absorbs the advantages that these prototypes have, but also addresses their weaknesses.

CHAPTER 3

THE DYNAMIC WEB SERVICE INVOCATION FRAMEWORK

The Dynamic Web Service Invocation Framework (DWSIF) is an infrastructure between a Web service and its clients. It tracks and records all Web service changes and gives a service consumer the ability to invoke a target service dynamically.

3.1 The Goals of DWSIF

DWSIF is intended to take the advantages of dynamic Web service invocation and to expand them even further. Meanwhile it addresses some of the challenges that are caused by this type of invocation. Among these challenges that SOA is facing today, our framework will address maintainability, reliability, and performance.

3.1.1 Maintainability

Maintainability is always a challenge for the distributed systems. In the existing distributed systems, such as DCOM, whenever a remote server component is recompiled and redeployed, its client applications have to re-reference the new version of the component that has been recompiled and redeployed. Since there is no absolutely bug free software, the related

software often makes changes after its deployment. Sometimes, when the environment which hosts a server application is changed, the client application needs to be modified to adapt to the changes. The business process and technology have been changing rapidly. The software enhancement requests will never terminate, until the software is replaced. Bug fixing function and software enhancements are easier done in the stand-alone software than in a connected system. Normally, in a connected system, one application change will require their relative applications to make changes also. Specifically, if many organizations are using one application, the coordination and the risk management will be needed and can be very involved among them. The cost of maintanence and the risk of failure in the connected system could be double or even triple that of a stand-alone application.

The Web service has already improved its maintainability significantly. Currently, if the method of a service only changes its implementation and keeps its interface unchanged, the client program can remain the same. There is no need for a client program to replace an existing one with a new one, and recompile the entire application again, since the mechanism is similar to that in the legacy component based system. But, when a provider changes service interfaces, this event will break the clients' applications. The DWSIF will address this issue and improve the maintainability in a connected system. By using the DWSIF, the maintenance cost of a system, which uses the dynamic invocation to invoke Web service, is

lower than that of a system which uses the static invocation to do the same thing. In Chapter 4, we will use a DWSIF prototype to show the result.

3.1.2 Reliability

In the connected system, the service consumers require that the services are highly reliable. Software reliability is the probability that software will provide failure-free operation in a fixed environment for a fixed interval of time (UTCS, 2005). High reliability of a Web service means that the failure rate of the service is low, when its clients invoke it under a fixed environment for a fixed interval of time.

One major factor that affects the reliability of a connected system is the dependency among its many components. In the SOA, the service consumers and their service providers most likely are from different organizations, and the Web services are hosted by a server platform such as a Web server. Therefore, the dependency of a connected system is high, and the reliability is low. For example, service providers could be doing system maintenance tasks, such as switching to a different database server or web server, or the web server software can malfunction. All these actions will make the services unavailable to their customers.

However, if the client applications can make adjustments to accommodate these sudden changes with additional help at the beginning of their invocations, the client still can get the results, which they normally get. As discussed in Chapter 2, the dynamic invocation gives clients the flexibility

to make necessary adjustments during their invocation processes. The DWSIF will provide additional assistance to Web service clients, so that the reliability of the connected system will remain high, even when the dependency is high. In Chapter 4, we will measure the reliability in the system with and without DWSIF.

3.1.3 Performance

As we evaluate a service or a system, one of the key factors we must measure is its performance. Poor performance can make software almost unuseable. No one wants to make a single request from an application to a Web service, and then, the application will wait for a long time to get a response back from this Web service. But under the connected system, the Web service, its client systems, and the service broker are all remote from each other. The performance is one of the major challenges in a connected system.

Not only does the DWSIF keep and improve the maintainability and reliability of the dynamic invocation, it also minimizes the performance sacrifice. By using the DWSIF to invoke a method of the service, the client will not notice a performance difference between static invocation and dynamic invocation, after the service has been invoked once. In Chapter 4, we will demonstrate the performance of both static invocation and dynamic invocation powered by the DWSIF.

3.2 The DWSIF Architecture

The role of the DWSIF is very important in the entire structure. Within the SOA framework, the DWSIF can be viewed as a broker or registry. A client uses it to locate and invoke a Web service. Service providers use it to register their Web services, and to maintain additional information, which is outside of the current WSDL specification. It also stores and tracks the clients and services relationship.

The DWSIF encapsulates all the changes the service providers make, masks the changes, and gives client applications an unchanged interface through its configuration engine, so that the client applications do not have to make changes each time when their service providers change a Web service interface. It also gives the client the freedom to change its application to adapt to the service changes without using the DWSIF configuration engine during the dynamic invocation process. The client application invokes a method of a Web service through the DWSIF. Due to the possible constant changing conditions of the Web service, the DWSIF has to use the dynamic invocation to invoke a requesting service for clients to avoid changing itself, whenever a service interface is changed. The DWSIF uses a caching mechanism to compensate the performance issue caused by dynamic invocation.

There are five main components in the DWSIF Architecture: Dynamic Invoker, Caching, Registry, Registration, and Interface. The structural and functional features are briefly discussed below and the related structure is schematically shown in Figure 3-1.

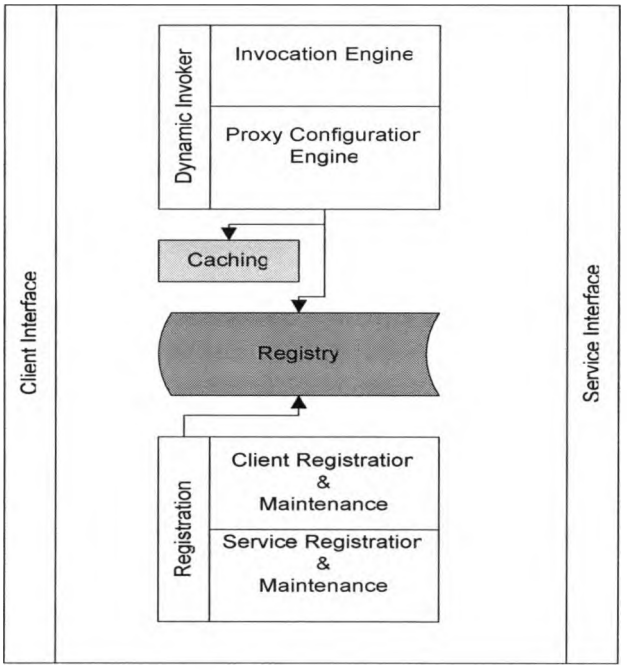

Figure 3-1 The DWSIF architecture and its components

3.2.1 Dynamic Invoker

A client application invokes a series of Web services through the Dynamic Invoker component. This component has two parts. Proxy Configuration Engine is responsible for validating and supplying a proxy of requesting service to the Invocation Engine. When a client invokes a new or updated Web service, the Proxy Configuration Engine will generate a proxy for that remote service. Or, the proxy can be obtained from the existing client local cache. Invocation Engine is responsible for invoking a requested Web method by using the proxy that is supplied by the Proxy Configuration Engine. Another responsibility of the Invocation Engine is to map and adapt the interface changes of a Web service to give the client an unchanged interface.

3.2.2 Caching

The Caching is a special place on the client side for storing the previously built proxy. After the first time a client invokes a new or changed service, the Proxy Configuration Engine will put the newly generated proxy into the client local caching directory for subsequent invocation use. If a client uses a cached proxy to invoke the Web service, its performance should match the static invocation performance. It also stores the service profiles, which contain information about service method. This enables the Invocation Engine maps and adapts the changes of the requesting service, so that the engine does not have to query the Registry to get the method information for every single invocation.

3.2.3 Registry

It is a relational database that stores all the WSDL information, additional information, such as default value for each parameter of the method, and client information, such as the client caching location. It also maintains the relationships between clients and their service providers. It is maintained by Registration process (See Figure 3-1). Figure 3-2 shows the relationships among each Registry database entity.

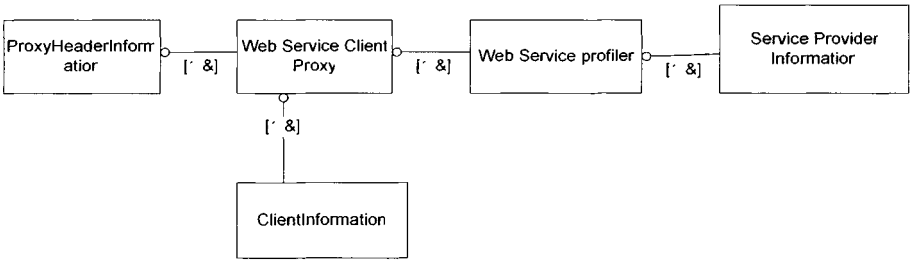

Figure 3-2 The Relationships among the Registry database entities

3.2.4 Registration

It is a process that the client and the service provider register and update their information into the Registry. It includes Client Registration and Maintenance and Service Registration and Maintenance. The Service Registration and Maintenance also creates profiles for each Web service methods' information.

3.2.5 Interface

The Client Interface and the Service Interface are the interfaces through which client systems and services provider access the DWSIF. Figure 3-3 is a sample DWSIF Library UML class diagram.

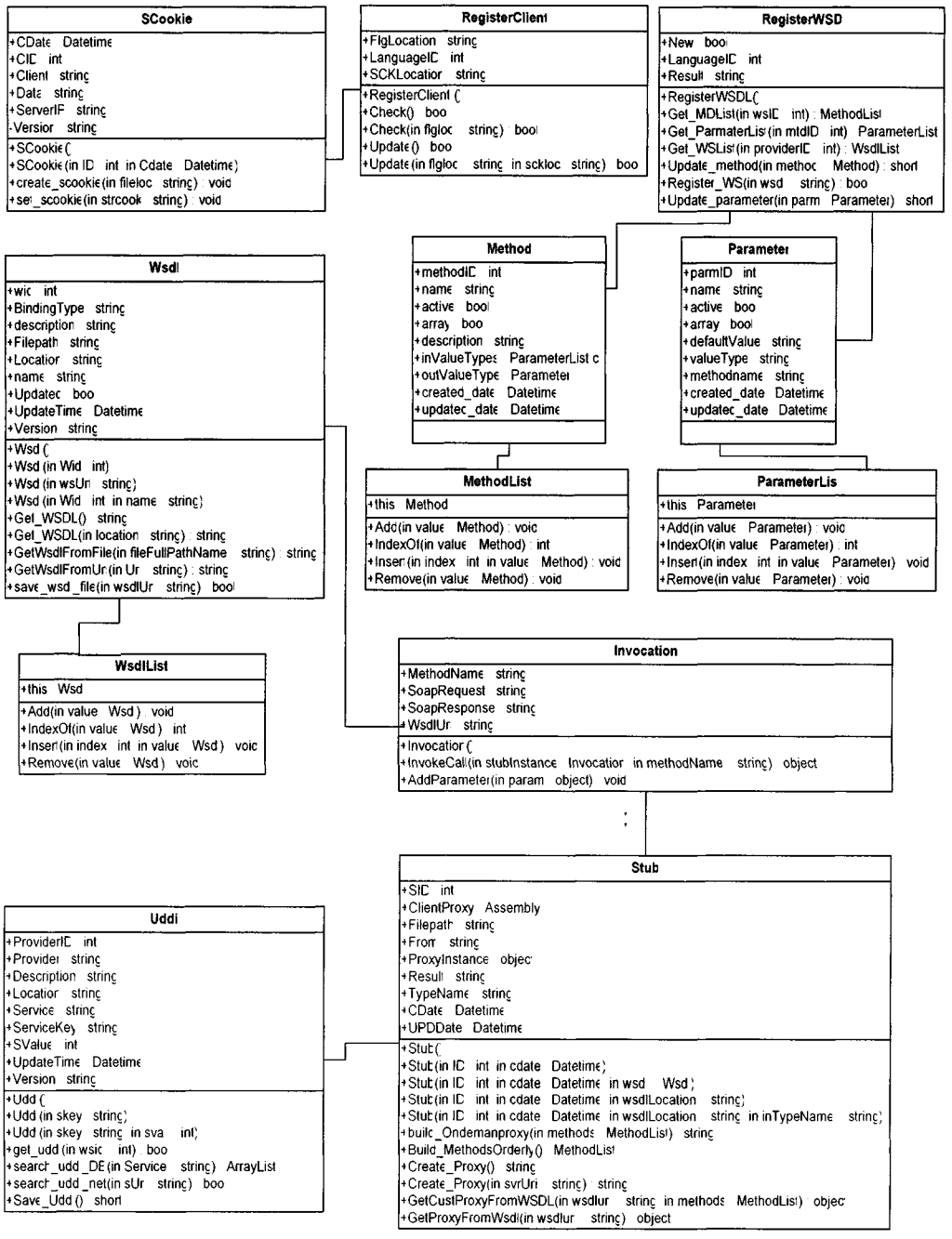

Figure 3-3 The DWSIF Library UML Class Diagram

3.3 The DWSIF work flow

The DWSIF work flow is schematically shown as the following UML sequence diagram in Figure 3-4.

For a Service Provider that needs to register a new or updated Web service, the process includes three steps:

Step 1 – The provider accesses the DWSIF by subscribing the Service Interface first.

Step 2 – The provider publishes the Web service at the Registry through Registration.

Step 3 – At the end of the registration process, the DWSIF sends a flag file. In this way, the client will know that the Web service has been updated since the last invocation, and will delete the existing cached proxy at client location.

For a Client that needs to invoke the Web service, the process includes six steps:

Step 1 – The client accesses the DWSIF through Client Interface first.

Step 2 – The client invokes the Web service through the Invocation Engine of DWSIF.

Step 3 – The invocation process requests to get the matching proxy from the Proxy Configuration.

Step 4 – If it is a new proxy, the Proxy Configuration will validate, confirm with the Registry, and then generate a proxy for the invocation and cache it into the specified client location. If it is an existing proxy, the Proxy Configuration will get it from the client caching location, and pass it to the invocation process.

Step 5 – The invocation process uses the proxy to invoke a method of the Web service.

Step 6 – The invocation process returns the result back to the client.

For a client that needs to update its information, the process includes three steps:

Step 1 – The client accesses the DWSIF by describing the Client Interface first.

Step 2 – The client registers its information, such as local caching location, or the status for adopting the Web service changes, at Registry through Registration process.

Step 3 - At the end of the registration process, the DWSIF updates the flag file in the client location. If a client adopts the Web service changes, the flag file will be deleted. At the next invocation process, the Invocation Engine of the DWSIF will not need to map and adapt the interface changes of the Web service anymore.

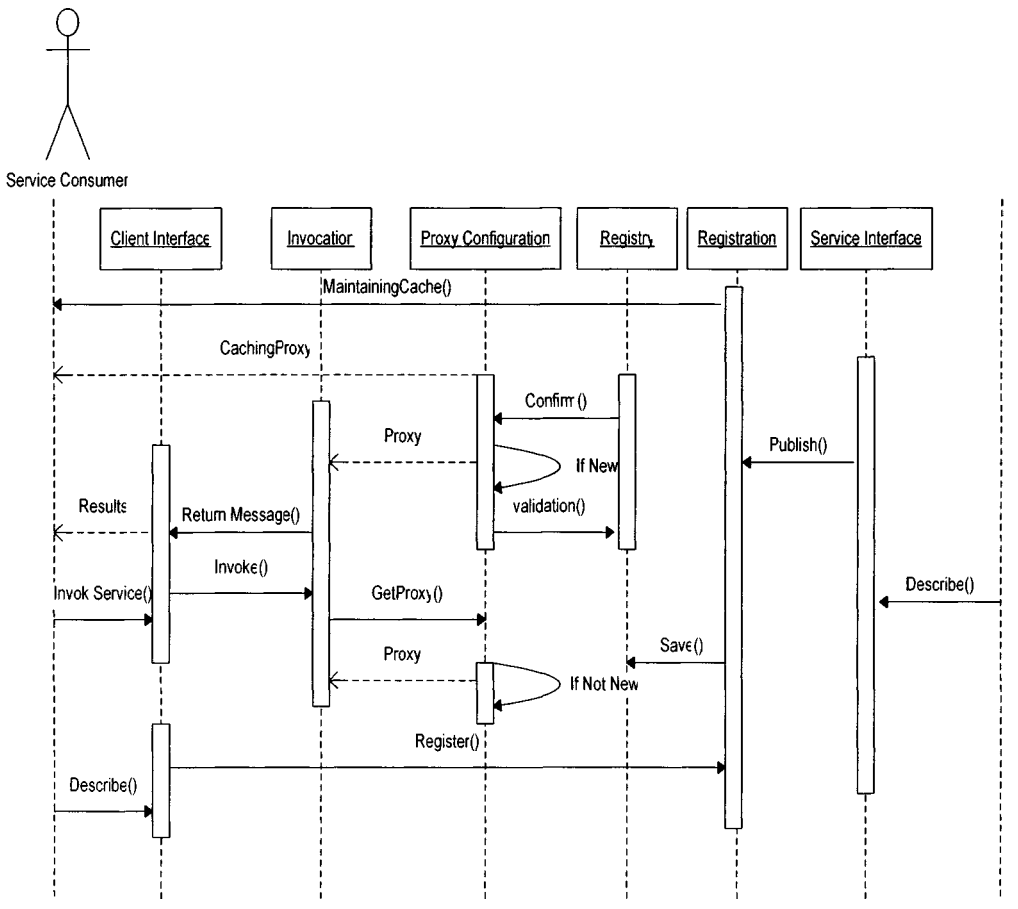

Figure 3-4 The DWSIF UML Sequence Diagram

Figure 3-5 shows the DWSIF invocation activity with extra details. During the Checked WS Proxy process, the Proxy Configuration (Figure 3-1) will check the existing proxy in the client caching location. If the proxy already existed, the Proxy Configuration (Figure 3-1) will use this proxy. Otherwise, it will create a new one and save it at the client caching location for the next invocation. Before the Invocation engine uses the proxy to invoke a method of the Web service, it will check whether or not the method is changed by looking for a new flag. If the new flag exists, the Invocation process will get an updated method profile from Registry and cache it for

the next invocation. Finally, it fulfills the client request by mapping this profile information and invoking the Web service.

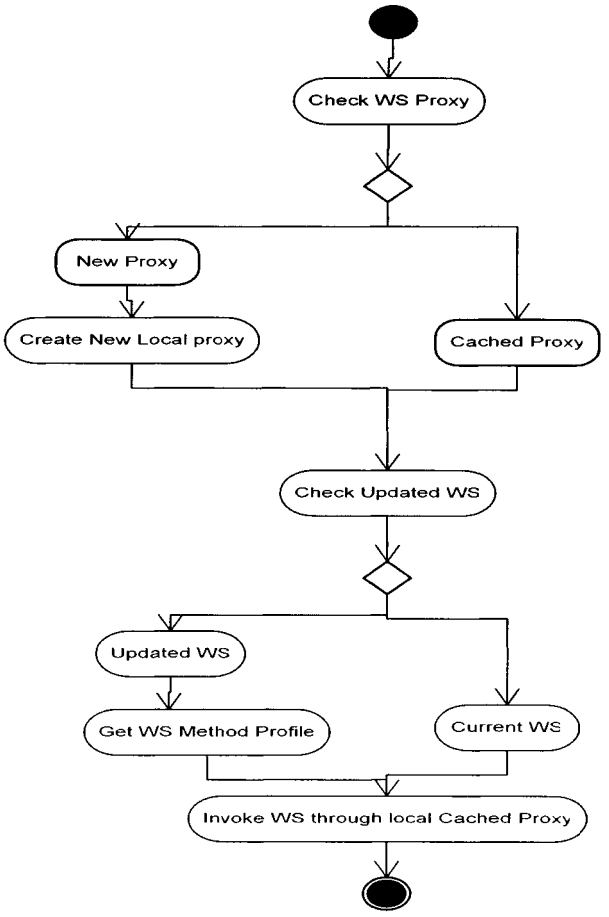

Figure 3-5 The DWSIF invocation activity

3.4 The DWSIF enabling technologies

As we can see, the DWSIF is a complicated framework. Fortunately, the current technologies are capable of implementing the DWSIF. Caching, Reflection and Late Binding, Relational Database, and Secure File Transfer are excellent candidates for building the DWSIF.

- Caching

Caching is widely used in the Web system and the embedded system for boosting the system performance. By implementing caching technique, the system can store frequently used data or object for repeated requests, so that the system does not have to generate the same requested data over and over again. The DWSIF not only caches the data, but also caches the proxy as a binary library form in the client location.

- Reflection and Late Binding

  The current generation compliers, such as .Net and J2EE, all have the capability to get the Meta data of an object during run time. Late binding is also widely used to build a component-based system. It allows binding of an object at run time. DWSIF uses these two techniques as a foundation to invoke Web services dynamically.

- Relational Database

  The Relational Database is used to store the additional information needed for dynamic invocation. Also, there are many available database drivers for accessing the data from an application directly. Therefore, the DWSIF builds its registry as a relational database, so that the clients and the service providers can access their information easily.

- Secure File Transfer

  The Secure File Transfer is a mature file transfer protocal. It already has a few excellent implementations, such as secure shell (SSH). The DWSIF will use a secure file transfer method to maintain the client cache location.

3.5 Conclusion

As a stand-alone system, the DWSIF is a platform where the client applications can invoke a Web service dynamically, and the service providers and their clients can register or maintain their information. Under the SOA, the DWSIF is a service broker that enforces the contracts between the Web service and its clients from breaking by both sides. It will address the challenges including maintenance, reliability, and performance, which the Web services and their clients are facing. The ultimate goal of the DWSIF is that, if a connected system uses the DWSIF as its service broker, the service providers and their customers will never worry about breaking the client applications due to changing a Web service interface. It can be built from existing approved technologies.

CHAPTER 4

THE DWSIF PROTOTYPE

Based on our DWSIF design mentioned above, we built a prototype framework to implement the dynamic invocation process. The featured structure is schematically shown in Figure 4-1. The prototype framework includes a DWSIF Service Registry that contains a DWSIF Registration Web Site (See Figure 4-1) – an implementation of Registration Process (See Figure 3-2), and DWSIF Registry Database (See Figure 4-1) – an implementation of Registry (See Figure 3-2), and a DWSIF Library (See Figure 4-1) – an implementation of Dynamic Invoker and Registration (See Figure 3-2).

Through this structure, we were able to test the prototype framework to verify the framework performance related to the target design. For conducting this test, we built a client Web application – Travel Site, and a few Travel Web services (See Figure 4-1).

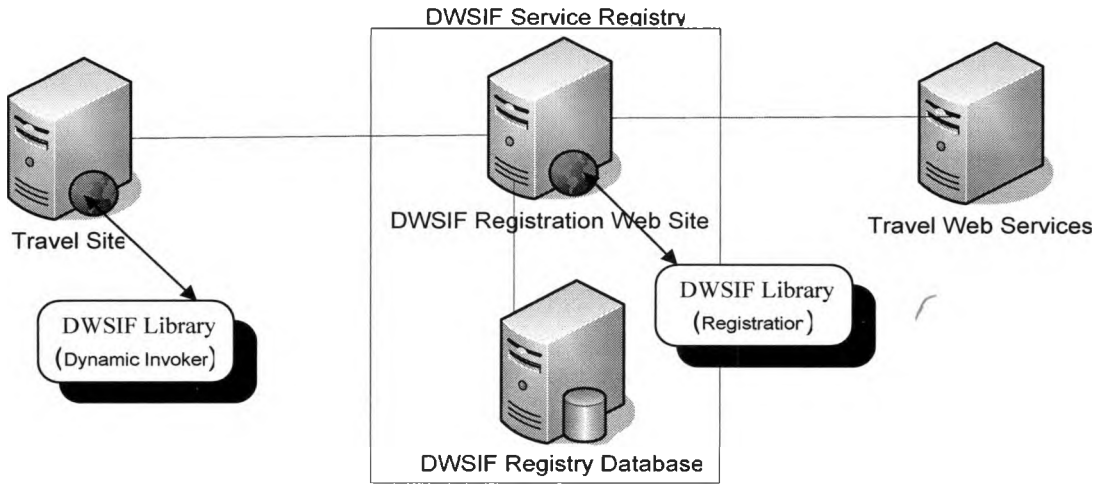

Figure 4-1 the DWSIF Prototype Architecture

4.1 The DWSIF Service Registry

The service providers use the Service Registry to register their information, such as the service WSDL information. The service clients use the Service Registry to register their information, such as the client configuration information.

After this provider deploys the new or updated Web services successfully, the event that a service provider takes to register a Web service will be generated in three steps. They are briefly described below.

Step 1 – The service provider logs into the DWSIF Service Registry by selecting provider radio button (See Figure 4-2).

Figure 4-2 Registry Login Screen

Step 2 – The providers go to the Web service Registration screen (See Figure 4-3). They can either register a new Web Service, or update an existing Web service, by selecting Version radio button. By picking a language dropdown list, the Registry will be able to track the information about the programming language in which the Web service is written. This will help the invocation engine handle the interoperability issues between a Web service and its client applications in the future. It will not affect the invocation processes. After clicking the Register button, the Service Registration and Maintenance process (See Figure 3-2) is fired. The new or updated Web service information is stored in the database. At the end of the registration, the user will be notified whether or not the process is successful.

Figure 4-3 Web service registration screen

Step 3 – The providers can update the Web service with additional information beside its WSDL information. This includes description for each Web service method, and the parameters' information of each Web method (See Figure 4-3). The Registry database will capture all the changes, and then notify the service clients by updating the Web service method profiles, which are the xml files containing the Web service method's Meta data in client cache location (See Figure 4-5). The prototype system uses Secure Shell (SSH) to transfer the profile information.

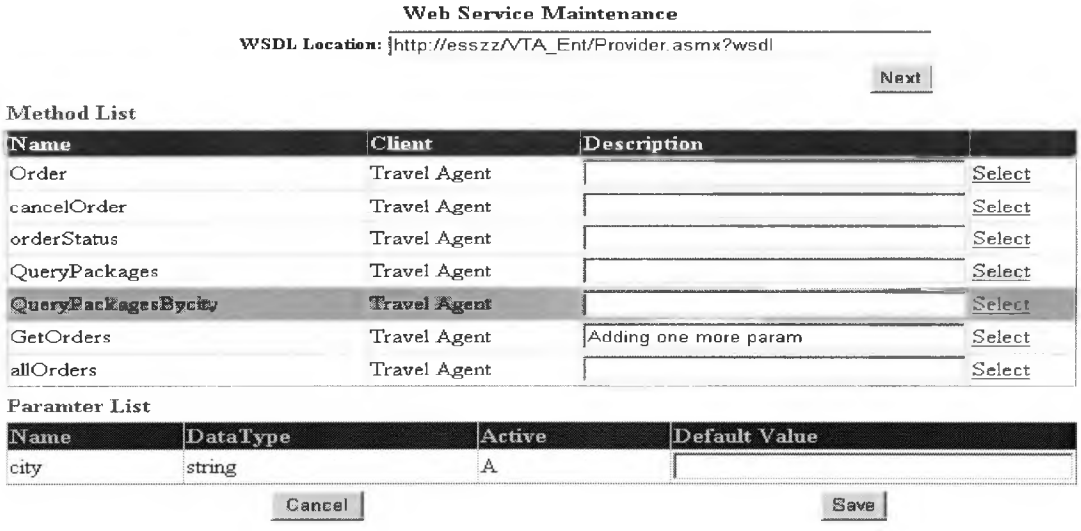

Figure 4-4 Web service maintenance screen

When the service consumers need to update their information such as cache location, an event will be generated in two steps. They are described below.

Step 1 – The consumers login to the DWSIF Service Registry by selecting client radio button (See Figure 4-2).

Step 2 – The consumers go to the Client Update Notification screen (See Figure 4-4). The consumers can update their flag and cookie locations. These information will instruct service providers where they need to send a notification flag and the updated Web service method Meta data profiles. This notification flag, Newmethod.flg, notifies the dynamic invoker – the Web service is updated since the last invocation, so that the dynamic invoker will regenerate a new proxy, vta_ent.dll, and cache it in the cache location (See Figure 4-6). The Figure 4-7 shows a sample profile. This profile will show dynamic invoker what Web service method interface looks like after the update, so that the dynamic invoker can map new interface to the one recognized by the client applications. By caching these frequently used data, and the related components, the overall performance of the framework will be dramatically improved.

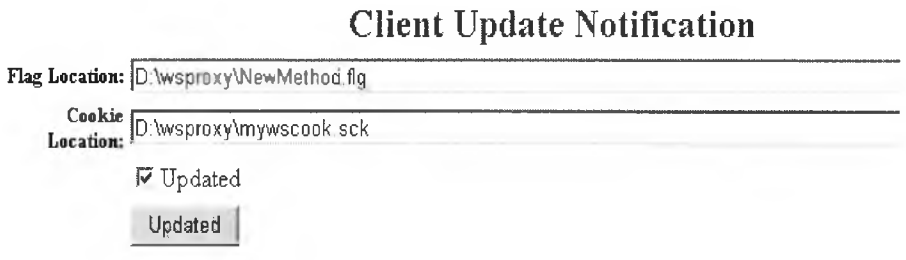

Figure 4-5 Client registration screen

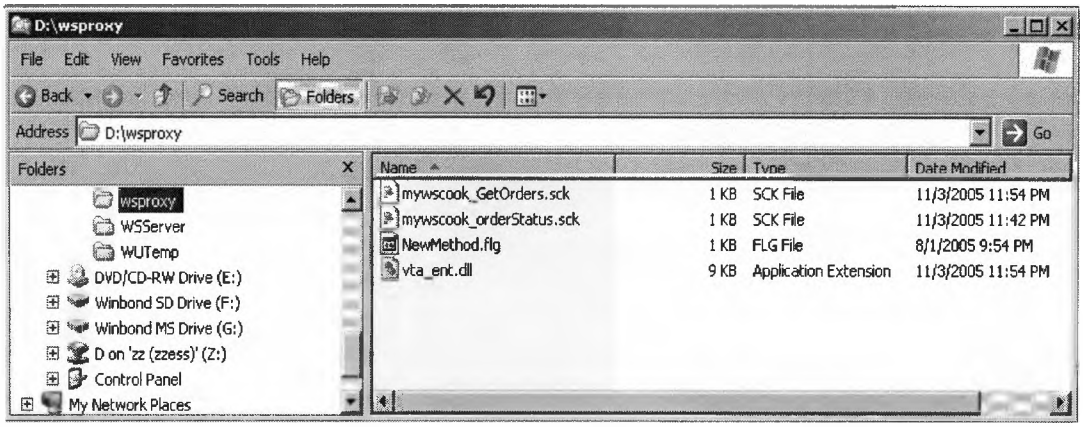

Figure 4-6 Client cache location

```
<Method>
  <GetOrders>
    <params>
      <param name="Order_Date" active="A" vtype="string" dvalue=""></param>
      <param name="status" active="N" vtype="string" dvalue=""></param>
    </params>
  </GetOrders>
</Method>
```

Figure 4-7 a sample of Web service method profile – mywscook_Getorders.sck

4.2 The DWSIF Library

There are two sets of libraries, one for the Dynamic Invoker and another one for the Registration.

4.2.1 The Dynamic Invoker (See Figure 4-8-a)

This library is an implementation of the DWSIF Dynamic Invoker (See Figure 3-2). They are the key components.

DBAccess is the data access layer. All components access Registry database through the DBAccess component.

Invocation is the Invoking Engine component (See Figures 3-2, 3-3).

Stub is the Proxy Configuration Engine component (See Figures 3-2, 3-3).

Method is the object representing the Web service method (See Figure 3-3).

MethodList is the collection of Method objects.

Parameter is the object representing the parameter of the Web service method (See Figure 3-3).

Parameter List is the collection of Parameter objects.

Wsdl is the object representing the WSDL document (See Figure 3-3).

SCookie is the object representing Web service method profile.

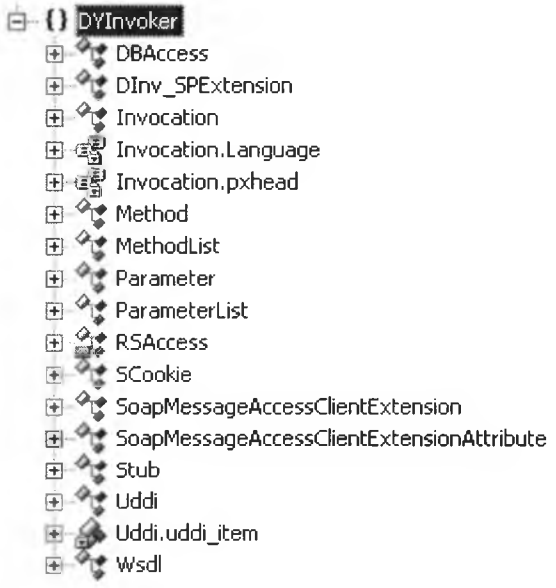

Figure 4-8-a Dynamic Invoker Library

4.2.2 The Registration (See Figure 4-8-b)

This library is an implementation of the DWSIF Registration (See Figure 3-2). The key components of this library are listed below:

DBAccess is the data access layer. All components access Registry database through the DBAccess component.

RegisterClient is the Client Registration & Maintenance component (See Figures 3-2, 3-3).

RegisterWSDL is the Service Registration & Maintenance component (See Figures 3-2, 3-3).

Method is the object representing Web service method (See Figure 3-3).

MethodList is the collection of Method objects.

Parameter is the object representing the parameter of Web service method (See Figure 3-3).

Parameter List is the collection of Parameter objects.

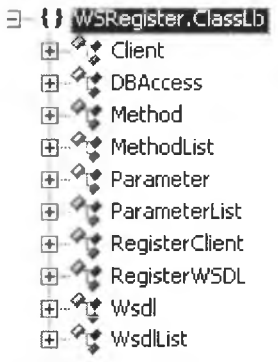

Figure 4-8-b Registration Library

4.3 The DWSIF Registry Database (See Figure 4-9)

The schema of the DWSIF Registry Database is shown in Figure 4-8. The tables and their brief descriptions are listed below:

PXHType and **PXHeader** store general header information for a Web service proxy.

PXLanguage stores the information about the programming language building this proxy.

Proxy represents a client Web service proxy, and has its general information such as version and namespace.

PXMethod contains the information about each method of the Web service proxy, such as method name and returning parameter.

PXParametersIn stores information about input parameters of each method.

Valuetype defines the value type for input or output parameters.

Client stores information about service client applications, such as cache location.

WSProfile contains Web service information from service provider perspective.

WSProvider contains service providers' information.

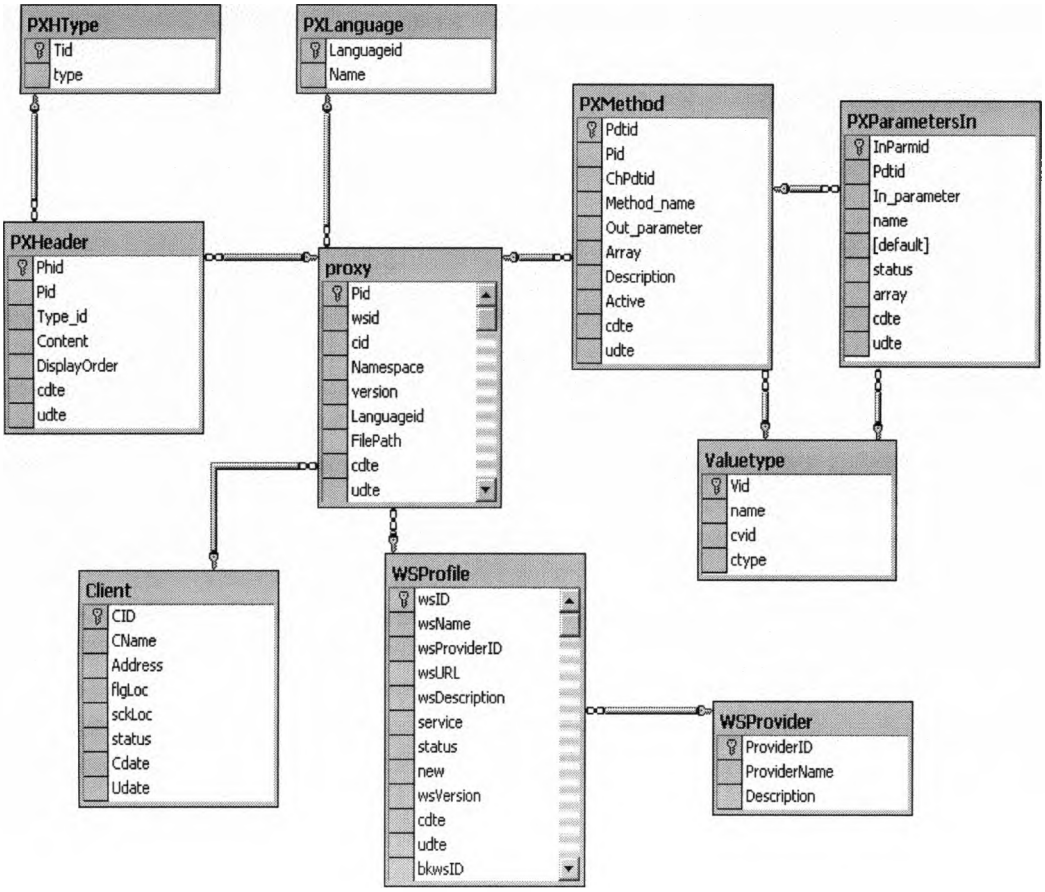

Figure 4-9 the DWSIF Registry Database

4.4 The DWSIF testing

The testing is conducted in the MS Windows environment. The .Net Web services are hosted in IIS 6.0 Web server. The J2EE Web service is hosted in Bea Weblogic Server 8.0. Both Web servers are running on Window 2003 server. The Registry is built in MS SQL Server 2000.

There are two .Net version Travel Web services (See Figures 4-10-a, 4-10-b), and one J2EE version Travel Web service (See Figure 4-10-c). Because these two .Net Web services have the same interface, they can be the backup for each other. There is a Web application as a service client - Travel site (See Figure 4-11). It uses

the DWSIF to invoke the Travel Web service, and the J2EE version Travel Web service dynamically and also invokes them statically. The SInvoke buttons will fire static invocation. The DInvoke buttons will fire dynamic invocations (See Figure 4-11).

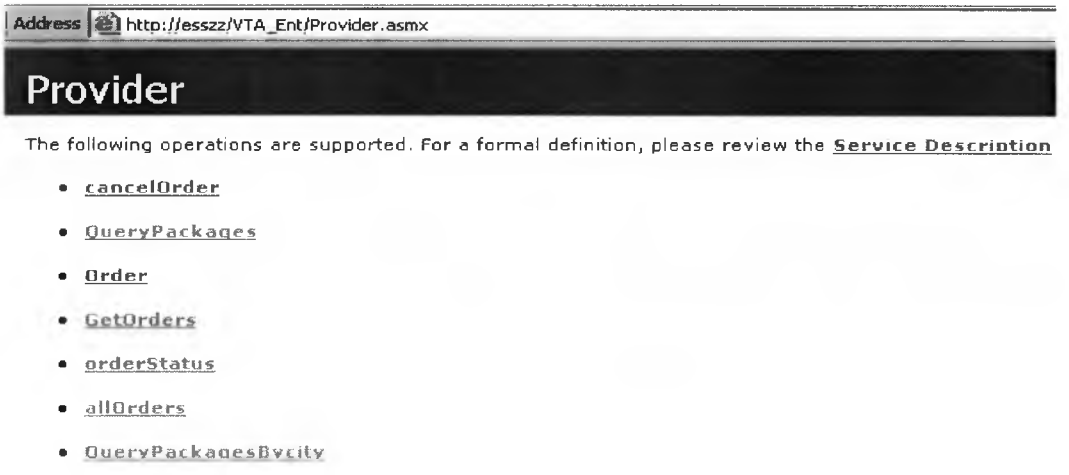

Figure 4-10-a VTA ENT .Net Travel Web service

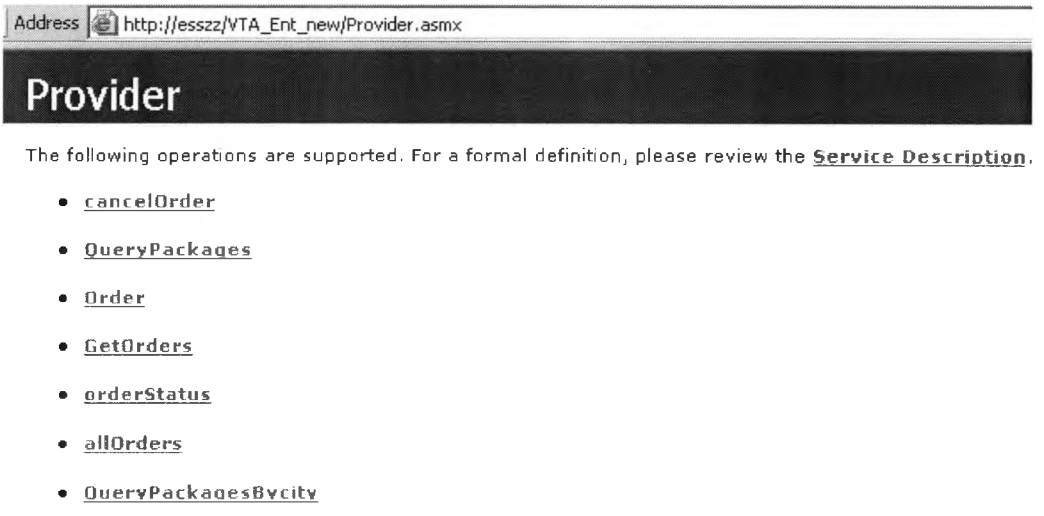

Figure 4-10-b VTA ENT backup .Net Travel Web service

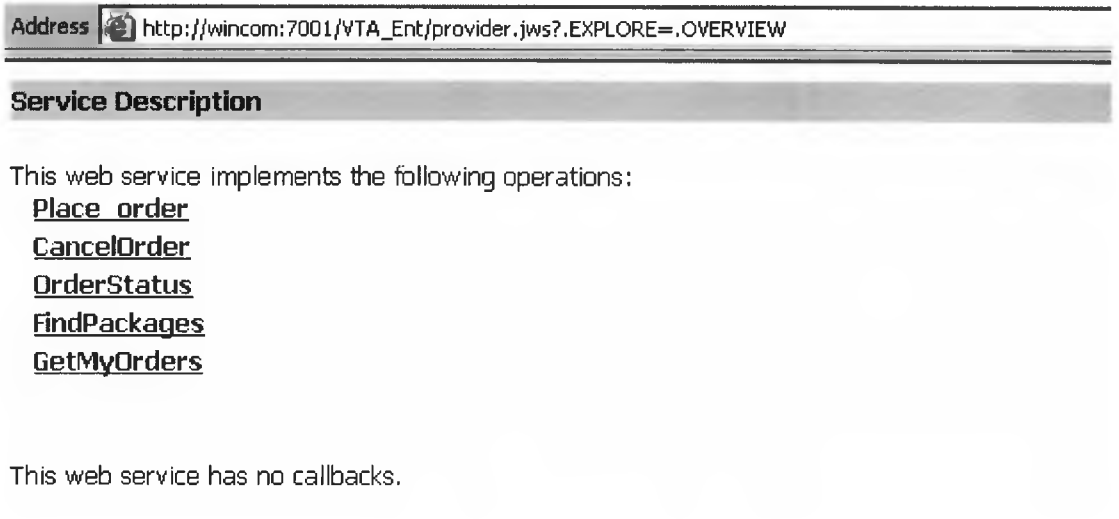

Figure 4-10-c VTA ENT Travel J2EE Web service

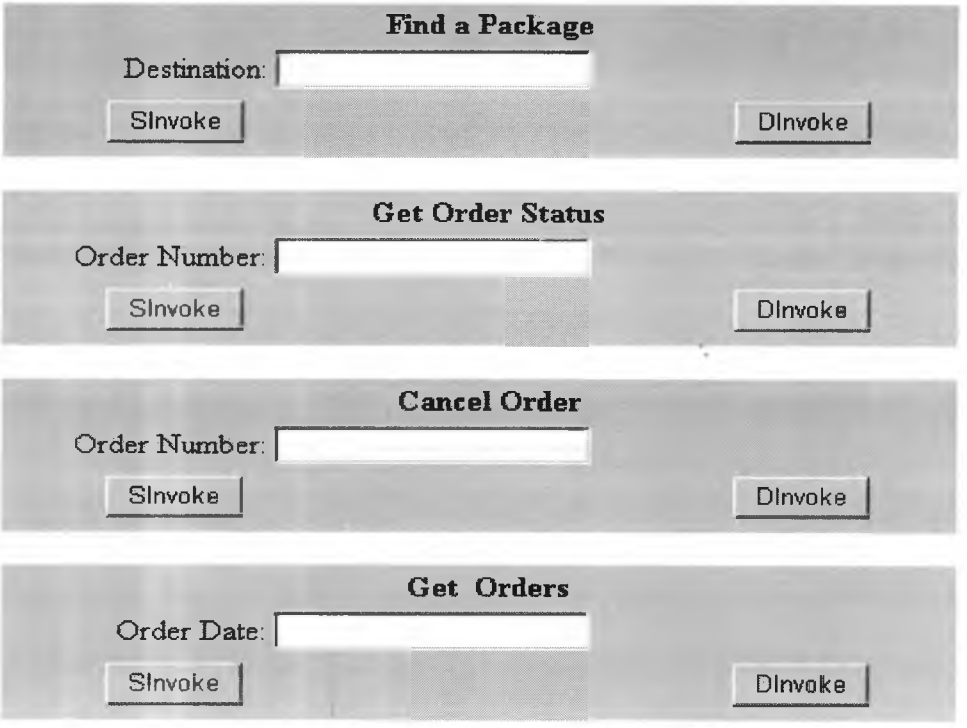

Figure 4-11 Travel Client Web Application

4.5 DWSIF Testing result

Through a well organized test plan, we focused our efforts on the assessment of the following functional related indexes: maintainability, reliability, and performance. The testing procedures, the results, and the analysis are described in detail below:

4.5.1 Maintainability Testing

For this test, our goal is to show that the client dynamic invocation code does not need to be changed by using the DWSIF, if a Web service method interface is changed.

During this test, the following steps were conducted.

Step 1 – In the client web application Get Order section (See Figure 4-11), we entered an order date, and then clicked SInvoke and DInvoke buttons to statically and dynamically invoke Web method. Both got a correct result.

```
[WebMethod]
public string GetOrders(string Order_Date)
```

Step 2 - We added an additional input parameter, status, into this Web method.

```
[WebMethod]
public string GetOrders(string Order_Date, string status)
```

Step 3 - We deployed the updated Web service to IIS server.

Step 4 - We went through Web service Registration process (See Figures 4-2, 4-4) to update the Registry.

Step 5 - We repeated step 1. The client static invocation failed. The client dynamic invocation was successful.

Step 6 - We modified the static invocation code to include the additional parameter, status, and then recompiled the application.

Step 7 – We repeated step 5. Both got a correct result.

Step 8 – We repeated steps 1-7 at Cancel Order section. At this time, we changed the return data type from **short** to **bool** on cancelOrder method.

From

```
[WebMethod]
public short cancelOrder(int order_id)
```

To

```
[WebMethod]
public bool cancelOrder(int order_id)
```

Step 9 – We repeated steps 1-7 at Get Order Status section. At this time, we changed the data type from Integer to String on cancelOrder method.

From

```
[WebMethod]
public string orderStatus(int order_id)
```

To

```
[WebMethod]
public string orderStatus(string order_nbr)
```

Step 10 – We repeated steps 1-7 at Find a Package section. At this time, we removed an input parameter.

From

```
[WebMethod]
public Package[] QueryPackagesBycity(string city, bool all)
```

To

```
[WebMethod]
public Package[] QueryPackagesBycity(string city)
```

During this test, we recorded the number of lines of changed code (LOCC), and other efforts, such as re-referencing Web service, recompiling, retesting, and redeploying client program, under both static and dynamic invocation conditions. Tables 4-1-a and 4-1-b have these test results for both static and dynamic invocations.

Table 4-1-a Static invocation maintainability result

Method	Action	LOCC	Re-Reference	Recompile & Retest	Redeploy
GetOrders	Adding an input parameter	1	Yes	Yes	Yes
cancelOrder	Changing data type of return value	2	Yes	Yes	Yes
orderStatus	Changing data type of input parameter	1	Yes	Yes	Yes
Package	Removing an input parameter	2	Yes	Yes	Yes

Table 4-1-b Dynamic invocation maintainability result

Method	Action	LOCC	Re-Reference	Recompile & Retest	Redeploy
GetOrders	Adding an input parameter	0	No	No	No
cancelOrder	Changing data type of return value	0	No	No	No
orderStatus	Changing data type of input parameter	0	No	No	No
Package	Removing an input parameter	0	No	No	No

According to the data in Table 4-1-a and 4-1-b, if a Web service changes its interface, the client application using static invocation has to re-reference Web service, change application source code, recompile the code, retest, and redeploy the application to its production environment. On the other hand, the client application using dynamic invocation does not need to do these maintenance tasks, and the client application can still run well as before.

During the software life cycle, each task has a cost associated with it. The static invocation will bring each client additional maintenance cost, when the Web service makes a method interface change during its maintenance stage. Since the DWSIF can encapsulate the method interface change, and provide an interface that is acceptable to client at all time, the dynamic invocation will not bring each client additional maintenance tasks and cost by using it.

In the maintainability test, we only use a standard small size client application to invoke a standard small size demo Web service. When the client application is bigger and more complex, the maintenance costs for recompiling, retesting, and redeploying the application are higher. Also, if the client is another service that has many of its own clients, and they all use static invocation, the maintenance cost may cascade to each level of clients. As a result, the total maintenance cost for a Web service will be unpredictably high under the changing service method interface condition. But, by using the

DWSIF, no matter what kind of client it is, or how many there are, the dynamic invocation will not increase its client maintenance cost a bit, under the changing service method interface condition.

4.5.2 Reliability Testing

For this test, our goal is to show that a client dynamic invocation process will be successful by using the DWSIF, if the requested Web service becomes unavailable for some reasons, but its static invocation process will fail.

During this test, the following steps were conducted.

Step 1 - In the client application (See Figure 4-11), we entered an order number at Get Order Status section, then clicked SInvoke and DInvoke buttons to statically and dynamically invoke Web method of an existing Web service. We got correct results under both conditions.

Step 2 – Repeated Step 1, but we entered an order number at Cancel Order section (See Figure 4-11).

Step 3 – We went to Web service Registration screen (See Figure 4-3) to register a new Web service, which had the same interface as the current one.

Step 4 – We went to IIS 6.0 console, and stopped the primary Web service running to make that Web service unavailable.

Step 5 – We repeated step 1. The client static invocation failed. The client dynamic invocation was successful.

Step 6 – We repeated step 2. The client static invocation failed. The client dynamic invocation was successful.

Step 7 – We went to IIS 6.0 console and started the primary Web service again.

Step 8 – We repeated step 5. We got correct results under both conditions.

Step 9 – We repeated step 6. We got correct results under both conditions.

During this test, the Web service and its client were unchanged, and the same operations were repeated. The related data are organized and put into Table 4-2-a and Table 4-2-b.

Table 4-2-a Get Order Status operation result

Outside factor	Static Invocation	Dynamic Invocation
All normal	Success	Success
Web Service unavailable	Fail	Success

Table 4-2-b Cancel Order operation result

Outside factor	Static Invocation	Dynamic Invocation
All normal	Success	Success
Web Service unavailable	Fail	Success

In this test, even the client application, the client environment, and Web service were unchanged, the factors that could make the requesting Web service become unavailable, such as Web server malfunction, operating system malfunction, and server hardware failure could cause client static

invocation process to fail. On the other hand, client dynamic invocation process can get a new proxy from the DWSIF to invoke an available Web service through this proxy successfully.

According to our test result, if we use the formula - **Probability of the Reliability = (Times of Success) / (Times of Invocation),** to calculate the probability of the reliability of the Invocation, we can have the following conclusion.

Let 'A' represent the number of invocations under Web service available condition. And Let 'U' represent the number of invocations under Web service unavailable condition. Both 'A' and 'U' are integers greater than 0.

For the static invocation: Probability of the Reliability = A / (A+ U) * 100% < 100

For the dynamic invocation: Probability of the Reliability = (A + U) / (A + U) * 100% = 10

Therefore, the reliability of a static invocation is lower then a dynamic invocation, which is powered by the DWSIF.

4.5.3 Performance Testing

For this test, our goal is to show that, by using the DWSIF, client dynamic invocation response time can match the static invocation response time, after the first time a client invokes a new Web service.

During this test, the following steps were conducted.

Step 1 – Cleared the client cached proxy.

Step 2 – In client application (See Figure 4-11), we entered an order number at Find a Package section, then clicked SInvoke and DInvoke buttons to statically and dynamically invoke Web method of an existing Web service, and time the operation.

Step 3 - In client application (See Figure 4-11), we entered an order number at Get Orders section, then clicked SInvoke and DInvoke buttons to statically and dynamically invoke Web method of an existing Web service, and timed the operation.

Step 4 – Repeated steps 2 -3.

Step 5 - Repeated steps 2 -3.

Step 6 – Updated client code, and pointed the Web service Uri to a J2EE Web service, which had identical interfaces with .Net one.

Step 7 – Repeated steps 1-5.

The collected data for .Net version Web service are organized and put into Table 4-3-a, and, the data for .J2EE version Web service are put into Table 4-3-b.

Table 4-3-a .Net version Web service Invocation response time

Operations	Static Invocation	Dynamic Invocation
1st Find a Package section	2 second	5.9 second
2nd Find a Package section	0.15 second	0.21 second
3rd Find a Package section	0.18 second	0.26 second
1st Get Orders	0.83 second	0.89 second
2nd Get Orders	0.10 second	0.10 second
3rd Get Orders	0.01 second	0.03 second

Table 4-3-b J2EE version Web service Invocation response time

Operations	Static Invocation	Dynamic Invocation
1st Find a Package section	2.2 second	6.2 second
2nd Find a Package section	0.26 second	0.26 second
3rd Find a Package section	0.27 second	0.33 second
1st Get Orders	0.83 second	0.91 second
2nd Get Orders	0.13 second	0.21 second
3rd Get Orders	0.12 second	0.17 second

According to Table 4-3-a and 4-3-b, we can have the figure 4-12.

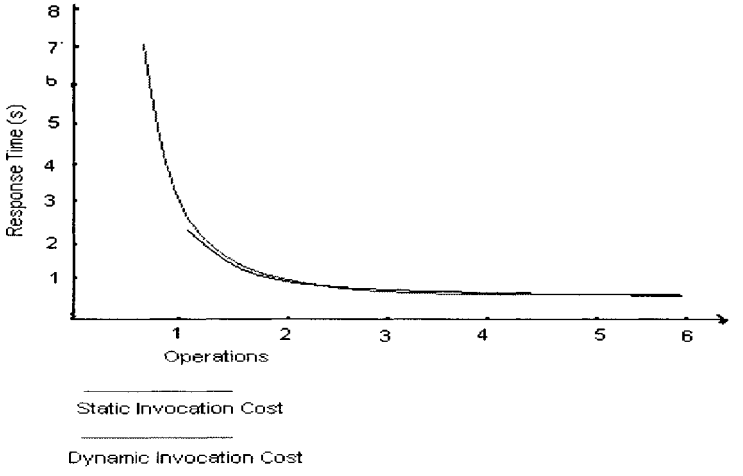

Figure 4-12 Invocation performance

According to our test result, at the first invocation, the static invocation is faster than dynamic invocation because, during the run time, dynamic invocation needs to build the proxy and cache it to client location through the DWSIF and static invocation does not need to do these tasks. But, after initial invocation, for all subsequent ones, the dynamic invocation can use the local cache proxy to invoke the Web service. As a result, by using the DWSIF, the dynamic invocation performance can match the static invocation.

4.6 Conclusion

The analysis based on the collected data from the three tests mentioned above shows that, by using the DWSIF, the dynamic invocation of a Web service has significant advantages over the static invocation in maintainability and reliability. Meanwhile, the performance in the case of using the dynamic invocation can match that using the static invocation.

CHAPTER 5

CONCLUDING REMARKS

5.1 Summary and Conclusion

Our analysis for the current Internet technology shows that the Web service and its client maintain a loosely coupled relationship. The Web service only provides its service whenever its clients invoke it. Therefore, we target the Web service invocation process and increase the capability of service broker as the main research topic. For achieving our research goal, we designed the Dynamic Web Service Invocation Framework (DWSIF) and an enhanced version of Service Broker.

Through designing the DWSIF comprehensively, implementing a prototype of the DWSIF by using all current approved technologies successfully, testing the framework precisely, and analyzing the test data objectively, we have shown that the Dynamic Web service invocation can serve its client better than static invocation, particularly in maintainability and reliability without sacrificing the performance. Our solution can dramatically decrease the system maintenance cost, increase the probability of system reliability, and significantly improve the performance of a dynamic invocation process.

Based on our research results, we believe that the invocation process has a fundamental impact on how flexible a Web service can be consumed by its clients.

5.2 Future Works

The relentless growth in Internet functionality and bandwidth have enabled a new wave of innovations that are transforming the way all types of organizations collaborate and interact with partners, both within and across organizational boundaries, forming a connected system. SOA is key to achieve the vision of a connected system. Dynamic invocation of Web service is critical for supporting SOA. Our results have shown that the DWSIF technique has a number of advantages over existing dynamic invocation mechanisms. However, due to limited resources and time, the DWSIF design is still at the developmental stage. The related techniques need to be developed in width and depth. The following items are some of the features that need to be enhanced before it can replace current implementation of UDDI registry:

- Registration process needs to be enhanced, for example it should be able to parse WSDL complex data type, and store them in the Registry database.
- Search interfaces need to be added to the Registration system, so that service consumers can use it to locate the services, which they are looking for easily.
- The security mechanism needs to be implemented in the Invocation engine, so that the framework can comply with W3C Web service security specifications, such as authentication between Web services and its clients, and encryption and decryption of SOAP packages.

The reliability and the maintainability are assessed based on simple Web services. More assessment must be conducted using commercial strength Web services and the client systems.

BIBLIOGRAPHY

Brown K. (2005). Security Features in WSE 3.0. *MSDN Magazine*, 5, 10-12.

Computer Science Department of UT Austin. (2005). Software Reliability, 2 paragraphs. Retrieved: July 22, 2005, from http://www.cs.utexas.edu /users/ethics/SoftwareStandards/sub/SoftWareReliability.html

Computer Associates Inc. (2005). Unicenter Web Services Distributed Management, 1 paragraph. Retrieved: July 28, 2005, from http://www3.ca.com/solutions/Product.aspx?ID=4714

Cerami E. (2002). Web Services FAQ, 1 paragraphs. Retrieved August 23, 2005, from http://www.oreillynet.com/pub/a/webservices/2002/02/12 /webservicefaqs.htm

Davydov M. (2005). Ease Web Service invocation with dynamic decoupling, 13 paragraphs. Retrieved: July 26, 2005, from http://www-128.ibm.com /developerworks/webservices/library/ws-soa-adventure1

Fang C., Liang D., Chen C., & Lin P. (2005). A Preliminary Study of Suppressing Redundant Nested Invocations from a Web Service with Active Replication, *International Journal of Web Services Research*, 1, 51-63

IBM Corp. (2005). Web Services Invocation Framework, 13 paragraphs. Retrieved July 22, 2005, from http://ws.apache.org/wsif

Irani R. (2002). Introduction, In WROX (Eds), (pp 1-30), *Professional Java Web Services*, New York, Peer Information Inc.

Microsoft Inc. (2005). DCOM is based on DCE RPC. DCOM information, 3 paragraphs. Retrieved August 30, 2005, from: http://www.microsoft.com /com/tech/dcom.asp

Morris M. & Chong F. (2005). Architectural Issues in Managing Web Services in Connected Systems, *MSDN Magazine*, 9, 16-18.

OASIS (2004). Web Services Security (WS-Security), 6 paragraphs. Retrieved: August 21, 2005, from http://www.oasisopen.org/committees/tc_home.php ?wg_abbrev=wss

OMG.org. (2005). The OMG CORBA/IIOP specification, 2 paragraphs. Retrieved July 12, 2005, from http://www.omg.org/technology/documents/formal /corba_iiop.htm

Oracle Inc. (2005). Document Style Web Services and Dynamic Invocation of Web Services, 13 paragraphs. Retrieved: July 16, 2005, from: http://www.oracle.com/technology/sample_code/tech/java/j2ee/jintdemo/tutorials/ webservices.html

Plummer D. (2002). Web Services FAQ, 2 paragraphs. Retrieved August 25, 2005, from http://www.oreillynet.com/pub/a/webservices/2002/02/12/ webservicefaqs.htm

Shan, T. C. & Hua, W. W. (2005). Service-Oriented Solution Framework for Internet Banking, *International Journal of Web Services Research,* 3, 29-48

Sun Microsystems Inc. (2003). The Java RMI specification, 4 paragraphs. Retrieved August 30, 2005, from http://java.sun.com/j2se/1.4.2/ docs/guide/rmi/spec/rmi-objmodel5.html

SYSTINET Inc. (2002). Introduction to Web Service Architecture, 6 paragraphs. Retrieved August 23, 2005, from http://www.mywebservices.org

Swartz S. (2005). Introduction to Indigo, 1 paragraph. Retrieved August 12, 2005, from http://msdn.microsoft.com/msdntv/episode.aspx?xml=episodes /en/20050407indigoss/manifest.xml

Urban Bettag. (2005). W3C Web Services Workshop Positioning Paper, 3 paragraphs. Retrieved August 23, 2005, from http://www.w3.org /2001/03/WSWS-popa/paper03

W3C. (2002). Web Services Activity, 3 paragraphs. Retrieved August 23, 2005, from http://www.w3.org/2002/ws/

Webopedia.com. (2004). EDI, 1 paragraph. Retrieved July 12, 2005, from: http://www.webopedia.com/TERM/E/EDI.html

Werner, C., Buschmann, C. & Fischer, S. (2005). WSDL-Driven SOAP Compression, *International Journal of Web Services Research*, 2, 18-35

Weyer C. (2004). DynWsLib-A .NET Library to dynamically call Web services at runtime, 5 paragraphs. Retrieved July 21, 2005, from: http://www.thinktecture.com/Resources/Software/DynWsLib/default

Yang A. (2004). Setting Up a Web Services Security Infrastructure, *Business Integration Journal*, 9, 3-5.

APPENDIX I

Web Service Source Code

Provider.asmx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Data.SqlClient;
using System.Web;
using System.Web.Services;

namespace Vta_Ent
{
    /// <summary>
    /// Summary description for Service1.
    /// </summary>
    public class Provider : System.Web.Services.WebService
    {
          private DBAccess dba;
          private string _city = "";
          private string _Edate = "";

          public Provider()
          {
                //CODEGEN: This call is required by the ASP.NET Web
Services Designer
                InitializeComponent();
                dba = new DBAccess("Data Source=Esszz; uid=ent_web;
pwd=ent666; Initial Catalog=EntProvider");
          }

          #region Component Designer generated code

          //Required by the Web Services Designer
          private IContainer components = null;

          /// <summary>
          /// Required method for Designer support - do not modify
          /// the contents of this method with the code editor.
          /// </summary>
          private void InitializeComponent()
          {

          }
```

```
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
              if(disposing && components != null)
              {
                    components.Dispose();
              }
              base.Dispose(disposing);
        }

        #endregion

        // WEB SERVICE EXAMPLE
        // The HelloWorld() example service returns the string Hello
World
        // To build, uncomment the following lines then save and build
the project
        // To test this web service, press F5

        public string city
        {
              get
              {
                    return _city;
              }
              set
              {
                    _city = value;
              }
        }
        public string eventDate
        {
              get
              {
                    return _Edate;
              }
              set
              {
                    _Edate = value;
              }
        }
        [WebMethod]
        public int Order(Order ord)
        {
              SqlDataReader read = dba.exec_sql("exec sp_enter_order " +
ord.packID + ", '" + ord.cardType + "','" + ord.cardNum + "','" +
ord.cardExpDate + "','" + ord.custName + "','" + ord.custAddr + "','" +
ord.email + "','" + ord.phone + "'," + ord.tprice);

              read.Read();
              return read.GetInt32(0);
        }

        [WebMethod]
        public bool cancelOrder(int order_id)
```

```
        {
              try
              {
                    return dba.run_sqlcommand(dba.create_cmd("exec
sp_cancelorder " + order_id));
              }
              catch(Exception ex)
              {
                    throw ex;
              }

        }

        [WebMethod]
        public string orderStatus(int order_id)
        {
              try
              {
                    SqlDataReader rdr = dba.exec_sql("exec
sp_getorderStatus " + order_id);
                    rdr.Read();
                    return "Status:" + rdr.GetString(0);
              }
              catch(Exception ex)
              {
                    throw ex;
              }
        }

        [WebMethod]
        public Package[] QueryPackages()
        {
              SqlDataReader rdr;
              ArrayList pcks = new ArrayList();
              if(_city != "")
              {
                    rdr = dba.exec_sql("exec sp_query_packages 1, '" +
_city + "'");
              }
              else
              {
                    rdr = dba.exec_sql("exec sp_query_packages 2, '" +
_Edate + "'");
              }
              while(rdr.Read())
              {
                    Package pck = new Package(rdr.GetInt32(0),
rdr.GetInt32(3), rdr.GetInt32(6));
                    pck.company = rdr.GetString(2);
                    pck.Edate = rdr.GetDateTime(4);
                    pck.name = rdr.GetString(1);
                    pck.price =  rdr.GetSqlMoney(5).ToDouble();

                    pcks.Add(pck);
              }
              if(pcks!=null)
              {
```

```
                    Package[] packs = new Package[pcks.Count];
                    for(int i=0; i<pcks.Count; i++)
                    {
                          packs[i]=(Package)pcks[i];
                    }
                    return packs;
              }
              else
              {
                    return null;
              }
        }

        [WebMethod]
        public Package[] QueryPackagesBycity(string city)
        {
              SqlDataReader rdr;
              ArrayList pcks = new ArrayList();
              rdr = dba.exec_sql("exec sp_query_packages 1, '" + city +
"'");
              while(rdr.Read())
              {
                    Package pck = new Package(rdr.GetInt32(0),
rdr.GetInt32(3), rdr.GetInt32(6));
                    pck.company = rdr.GetString(2);
                    pck.Edate = rdr.GetDateTime(4);
                    pck.name = rdr.GetString(1);
                    pck.price =  rdr.GetSqlMoney(5).ToDouble();

                    pcks.Add(pck);
              }
              if(pcks!=null)
              {
                    Package[] packs = new Package[pcks.Count];
                    for(int i=0; i<pcks.Count; i++)
                    {
                          packs[i]=(Package)pcks[i];
                    }
                    return packs;
              }
              else
              {
                    return null;
              }
        }

        [WebMethod]
        public string GetOrders(string Order_Date, string status)
        {
              SqlDataReader rdr;
              string ords = "";
              string sqlstr = "";
              if(status.Length > 0)
              {
                    sqlstr = "exec sp_getAllorder '" + Order_Date + "',
'" + status + "'";
              }
```

```
                else
                {
                      sqlstr = "exec sp_getAllorder '" + Order_Date + "'";
                }
                rdr = dba.exec_sql(sqlstr);
                while(rdr.Read())
                {
                      ords += "<order>";
                      ords += "<ordNum>" + rdr.GetInt32(0).ToString() +
"</ordNum>";
                      ords += "<status>" + rdr.GetString(11) +"</status>";
                      ords += "<custName>" + rdr.GetString(6) +
"</custName>";

                      ords += "</order>";
                }
                return ords;
          }

          [WebMethod]
          public Order[] allOrders(string Order_Date)
          {
                SqlDataReader rdr;
                ArrayList ords = new ArrayList();
                rdr = dba.exec_sql("exec sp_getAllorder '" + Order_Date +
"'");
                while(rdr.Read())
                {
                      Order ord = new Order(rdr.GetInt32(0),
rdr.GetInt32(1), rdr.GetString(3), rdr.GetString(4),rdr.GetString(6));
                      ord.status = rdr.GetString(11);
                      ord.OrdDate = rdr.GetDateTime(2).ToString();
                      ord.phone = rdr.GetString(9);
                      ord.tprice = rdr.GetSqlMoney(10).ToDouble();
                      ord.custName = rdr.GetString(6);

                      ords.Add(ord);
                }

                if(ords!=null)
                {
                      Order[] orders = new Order[ords.Count];
                      for(int i=0; i<ords.Count; i++)
                      {
                            orders[i]=(Order)ords[i];
                      }
                      return orders;
                }
                else
                {
                      return null;
                }
          }

    }
}
```

Package.cs

```
using System;

namespace Vta_Ent
{
    /// <summary>
    /// Summary description for Package.
    /// </summary>
    public class Package
    {
          private int _id;
          private string _name;
          private string _company;
          private int _contentID;
          private DateTime _Edate;
          private double _price;
          private int _locationID;

          public Package()
          {
                //
          }
          public Package(int id, int contentID, int locationID)
          {
                this._id = id;
                this._locationID = locationID;
                this._contentID = contentID;
          }
          public Package(int id, int contentID, int locationID, string
name)
          {
                this._id = id;
                this._locationID = locationID;
                this._contentID = contentID;
                this._name = name;
          }
          public Package(int id, int contentID, int locationID, string
name, string company)
          {
                this._id = id;
                this._locationID = locationID;
                this._contentID = contentID;
                this._name = name;
                this._company = company;
          }

          public int ID
          {
                get
                {
                      return _id;
                }
                set
                {
                      _id = value;
```

```
        }
    }
    public int contentID
    {
        get
        {
            return _contentID;
        }
        set
        {
            _contentID = value;
        }
    }
    public int locationID
    {
        get
        {
            return _locationID;
        }
        set
        {
            _locationID = value;
        }
    }
    public double price
    {
        get
        {
            return _price;
        }
        set
        {
            _price = value;
        }
    }
    public string name
    {
        get
        {
            return _name;
        }
        set
        {
            _name = value;
        }
    }
    public string company
    {
        get
        {
            return _company;
        }
        set
        {
            _company = value;
        }
    }
```

```
        public DateTime Edate
        {
              get
              {
                    return _Edate;
              }
              set
              {
                    _Edate = value;
              }
        }
    }
}
```

Order.cs

```
using System;

namespace Vta_Ent
{
    /// <summary>
    /// Summary description for Order.
    /// </summary>
    public class Order
    {
        private int _ordNum=0;
        private int _packID;
        private string _cardType;
        private string _cardNum;
        private string _custName;
        private string _custAddr;
        private DateTime _cardExpDate;
        private string _OrdDate;
        private string _email;
        private string _phone;
        private double _tprice;
        private string _status;

        public Order()
        {
            //
        }
        public Order(int packID)
        {
            _packID = packID;
        }
        public Order(int ordID, int packID)
        {
            _ordNum = ordID;
            _packID = packID;
        }
        public Order(int ordID, int packID, string cardType, string
cardNum, string custAddr)
        {
            _ordNum = ordID;
```

```
            _packID = packID;
            _cardType = cardType;
            _cardNum = cardNum;
            _custAddr = custAddr;
        }

        public int ordNum{get{return _ordNum;} set{_ordNum = value;}}
        public int packID{get{return _packID;} set{_packID = value;}}
        public string cardType{get{return _cardType;} set{_cardType =
value;}}
        public string cardNum{get{return _cardNum;} set{_cardNum =
value;}}
        public string custName{get{return _custName;} set{_custName =
value;}}
        public string custAddr{get{return _custAddr;} set{_custAddr =
value;}}
        public string email{get{return _email;} set{_email = value;}}
        public string phone{get{return _phone;} set{_phone = value;}}
        public string status{get{return _status;} set{_status = value;}}
        public double tprice{get{return _tprice;} set{_tprice = value;}}
        public DateTime cardExpDate{get{return _cardExpDate;}
set{_cardExpDate = value;}}
        public string OrdDate{get{return _OrdDate;} set{_OrdDate =
value;}}
    }
}
```

DBAcess.cs

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace Vta_Ent
{
    /// <summary>
    /// Summary description for DBAccess.
    /// </summary>
    public class DBAccess
    {
        SqlConnection sql_conn;

        public DBAccess(string StrConn)
        {
            sql_conn = new SqlConnection(StrConn);
            try
            {
                sql_conn.Open();
            }
            catch(Exception ex)
            {
                writetolog(ex.Message);
                sql_conn = null;
            }
```

```
        }
        ~DBAccess()
        {
              //if (sql_conn.State == ConnectionState.Open)
              //{
              //    sql_conn.Close();
              //}
        }
        private void writetolog(string logmessage)
        {
              Console.WriteLine("Hello World.");
        }

        public SqlDataReader exec_sql(string str_sql)
        {
              SqlDataReader rdr;
              //SqlConnection sql_conn = get_conn();
              try
              {
                    SqlCommand SCmd = new SqlCommand(str_sql, sql_conn);
                    rdr = SCmd.ExecuteReader();
                    return rdr;
              }
              catch(Exception ex)
              {
                    writetolog(ex.Message);
                    rdr = null;
                    throw ex;
              }
              finally
              {
                    //sql_conn.Dispose();
              }

        }

        public DataSet get_dataSet(string str_sql, string dsname)
        {
              DataSet ds = new DataSet();
              //SqlConnection sql_conn = get_conn();
              try
              {
                    SqlDataAdapter da = new SqlDataAdapter(str_sql,
sql_conn);
                    //SqlCommandBuilder bld = new SqlCommandBuilder(da);
                    da.Fill(ds, dsname);
              }
              catch(Exception ex)
              {
                    writetolog(ex.Message);
                    ds = null;
              }
              //sql_conn.Close();
              return ds;
        }
        //for Pagenation
```

```
        public DataSet get_dataSet(string str_sql, int st_row, int
nt_rows, string dsname)
        {
              DataSet ds = new DataSet();
              //SqlConnection sql_conn = get_conn();
              try
              {
                    SqlDataAdapter da = new SqlDataAdapter(str_sql,
sql_conn);
                    //SqlCommandBuilder bld = new SqlCommandBuilder(da);
                    da.Fill(ds, st_row, nt_rows, dsname);
              }
              catch(Exception ex)
              {
                    writetolog(ex.Message);
                    ds = null;
              }
              //sql_conn.Close();
              return ds;
        }

        public SqlCommand create_cmd(string sqlstr)
        {
              SqlCommand cmd;

              try
              {
                    cmd = new SqlCommand(sqlstr, sql_conn);
              }
              catch(Exception ex)
              {
                    writetolog(ex.Message);
                    cmd = null;
              }
              //cmd.Connection.Close();
              return cmd;

        }

        public short run_sqlcommand(SqlCommand cmd)
        {
              short res = 0;
              try
              {
                    cmd.ExecuteNonQuery();
                    res = 1;
              }
              catch(Exception ex)
              {
                    res = 2;
                    writetolog(ex.Message);
              }
              cmd.Connection.Close();
              return res;
        }
   }
}
```

Web.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <system.web>

    <!--  DYNAMIC DEBUG COMPILATION
          Set compilation debug="true" to enable ASPX debugging.
Otherwise, setting this value to
          false will improve runtime performance of this application.
          Set compilation debug="true" to insert debugging symbols (.pdb
information)
          into the compiled page. Because this creates a larger file that
executes
          more slowly, youeshould set this value to true only when
debugging and to
          false at all other times. For more information, refer to the
documentation about
          debugging ASP.NET files.
    -->
    <compilation
         defaultLanguage="c#"
         debug="true"
    />
    <!--  CUSTOM ERROR MESSAGES
          Set customErrorsemode="On" or "RemoteOnly" to enable custom
error messages, "Off" to disable.
          Add <error> tagsefor each of the errors you want to handle.

          "On" Always display custom (friendly) messages.
          "Off" Always display detailed ASP.NET error information.
          "RemoteOnly" Display custom (friendly) messages only to users
not running
           on the local Web server. This setting is recommended for
security purposes, so
           that you do notedisplay application detail information to
remote clients.
    -->
    <customErrors
    mode="RemoteOnly"
    />

    <!--  AUTHENTICATION
          This section sets the authentication policies of the
application. Possible modes are "Windows",
          "Forms", "Passport" and "None"

          "None" No authentication is performed.
          "Windows" IIS performs authentication (Basic, Digest, or
Integrated Windows) according to
           its settings for the application. Anonymous access must be
disabled in IIS.
          "Forms" You provide a custom form (Web page) for users to enter
their credentials, and then
```

```
            you authenticate them in your application. A user credential
token is stored in a cookie.
            "Passport" Authentication is performed via a centralized
authentication service provided
            by Microsoft that offers a single logon and core profile
services for member sites.
    -->
    <authenti˜ation mode="Windows" />

   <!--  AUTHORIZATION
            This section sets the authorization policies of the application.
You can allow or deny access
            to application resources by user or role. Wildcards: "*" mean
everyone, "?" means anonymous
            (unauthenticated) users.
    -->

    <authorization>
        <allow users="*" /> <!-- Allow all users -->
            <!--  <allow     users="[comma separated list of users]"
                             e  roles="[comma separated list of roles]"/>
                  <deny      e  users="[comma separated list of users]"
                             e  roles="[comma separated list of roles]"/>
            -->
    </authorization>

    <!--  APPLICATION-LEVEL TRACE LOGGING
            Application-level tracing enables trace log output for every
page within an application.
            Set trace enabled="true" to enable application trace logging.
If pageOutput="true", the
            trace information will be displayed at the bottom of each page.
Otherwise, you can view the
            application trace log by browsing the "trace.axd" page from your
web application
            root.
    -->
    <trace
        enabled="false"
        requestLimit="10"
        pageOutput="false"
        traceMode="SortByTime"
         localOnly="true"
    />

    <!--  SESSION STATE SETTINGS
            By default ASP.NET uses cookies to identify which requests
belong to a particular session.
            If cookies are not available, a session can be tracked by adding
a session identifier to the URL.
            To disable cookies, set sessionState cookieless="true".
    -->
    <sessionState
            mode="InProc"
            stateConnectionString="tcpip=127.0.0.1:42424"
            sqlConnextionString="data
source=127.0.0.1;Trusted_Connection=yes"
```

```
            cookieless="false"
            timeout="20"
    />

    <!--  GLOBALIZATION
          This section sets the globalization settings of the application.
    -->
    <globalization
            requestEncoding="utf-8"
            responseEncoding="utf-8"
   />

 </system.web>

</configuration>
```

CCard.java

```
public class CCard implements java.io.Serializable
{
    private String _cnum;
    private String _ctype;
    private String _cexpDate;

    public CCard()
    {
    }
    public CCard(String cnum, String ctype, String cexpDate)
    {
        this._cnum = cnum;
        this._ctype = ctype;
        this._cexpDate = cexpDate;
    }

    public void setCnum(String cnum)
    {
        this._cnum = cnum;
    }
    public String getCnum(){return _cnum;}

    public void setCtype(String ctype)
    {
        this._ctype = ctype;
    }
    public String getCtype(){return _ctype;}

    public void setCexpDate(String cexpDate)
    {
        this._cexpDate = cexpDate;
    }
    public String getCexpDate(){return _cexpDate;}
}
```

Customer.java

```
public class Customer implements java.io.Serializable
{
    private String _custName;
    private String _custAddr;
    private String _custEmail;
    private String _custPhone;

    public Customer()
    {

    }
    public Customer(String cname, String caddr, String cmail, String
cphone)
    {
        this._custAddr = caddr;
        this._custEmail = cmail;
        this._custName = cname;
        this._custPhone = cphone;
    }

    public void setcustAddr(String caddr)
    {
        this._custAddr = caddr;
    }
    public String getcustAddr()
    {
        return _custAddr;
    }

    public void setcustEmail(String cmail)
    {
        this._custEmail = cmail;
    }
    public String getcustEmail()
    {
        return _custEmail;
    }

    public void setcustName(String cname)
    {
        this._custName = cname;
    }
    public String getcustName()
    {
        return _custName;
    }

    public void setcustPhone(String cphone)
    {
        this._custPhone = cphone;
    }
    public String getcustPhone()
    {
        return _custPhone;}}
```

Package.java

```
public class Package implements java.io.Serializable
{
    private int _id;
    private String _Name;
    private String _Company;
    private int _ContentID;
    private String _Edate;
    private double _Price;
    private int _locationID;

    public Package()
    {
    }
    public Package(String name, String Company, String Edate)
    {
        this._Edate = Edate;
        this._Company = Company;
        this._Name = name;
    }

    public String getCompany()
    {
        return _Company;
    }
    public String getEdate()
    {
        return _Edate;
    }
    public String getName()
    {
        return _Name;
    }
    public void setContentID(int cid)
    {
        this._ContentID=cid;
    }
    public int getContentID()
    {
        return this._ContentID;
    }
    public void setPackageID(int pid)
    {
        this._id = pid;
    }
    public int getPackageID()
    {
        return this._id;
    }
    public void setLocationID(int lid)
    {
        this._locationID = lid;
    }
    public int getLocationID()
```

```
    {
        return this._locationID;
    }
    public void setPrice(double price)
    {
        this._Price = price;
    }
    public double getPrice()
    {
        return this._Price;
    }
}
```

provider.jws

```
import java.util.*;
import java.text.*;
import javax.naming.*;
import java.rmi.RemoteException;
import java.sql.*;
import javax.sql.*;
import weblogic.xml.schema.binding.*;
import weblogic.xml.stream.*;

/**
 * @common:xmlns
 */
public class provider implements com.bea.jws.WebService
{


    static final long serialVersionUID = 1L;

    /**
     * @common:operation
     */
    public int Place_order(int PackID, Customer cust, CCard crd, double
tprice) throws SQLException, RemoteException, NamingException
    {
        PreparedStatement pstmt = null;
        Connection conn = null;
        String sqlStatement = null;
        Statement stmt = null;
        ResultSet rs = null;
        String P_Name = null;

        try
        {
            conn = getConnection();
            sqlStatement = "exec sp_enter_order ?,?,?,?,?,?,?,?,?";
            pstmt = conn.prepareStatement(sqlStatement);
            pstmt.setInt(1,PackID);
            pstmt.setString(2,crd.getCtype());
            pstmt.setString(3,crd.getCnum());
            pstmt.setString(4,crd.getCexpDate());
            pstmt.setString(5,cust.getcustName());
            pstmt.setString(6,cust.getcustAddr());
            pstmt.setString(7,cust.getcustEmail());
            pstmt.setString(8,cust.getcustPhone());
            pstmt.setDouble(9,tprice);
            rs = pstmt.executeQuery();
            //return current Order ID
            rs.next();

            return rs.getInt(0);
        }
        catch (SQLException x)
        {
            throw new RemoteException("Can not insert New Order", x);
```

```
            //return "null";
        }
        finally
        {
            try
            {
                if(pstmt != null)
                    pstmt.close();
            }
            catch (Exception ex) { }
            try
            {
                if(conn != null)
                    conn.close();
            }
            catch (Exception ex) { }
        }
    }

    /**
     * @common:operation
     */
    public int CancelOrder(int OrdID) throws SQLException,
RemoteException, NamingException
    {
        PreparedStatement pstmt = null;
        Connection conn = null;
        String sqlStatement = null;
        try
        {
            conn = getConnection();
            sqlStatement = "exec sp_cancelorder ?";
            pstmt = conn.prepareStatement(sqlStatement);
            pstmt.setInt(1,OrdID);
            pstmt.executeUpdate();
            return 1;
        }
        catch (SQLException x)
        {
            throw new RemoteException("Can not Cancel order", x);
        }
        finally
        {
            try
            {
                if(pstmt != null)
                    pstmt.close();
            }
            catch (Exception ex) { }
            try
            {
                if(conn != null)
                    conn.close();
            }
            catch (Exception ex) { }
        }
    }
```

```
    /**
     * @common:operation
     */
    public String OrderStatus(int OrdID) throws SQLException,
RemoteException, NamingException
    {
        String status = null;
        String sqlStatement = null;
        PreparedStatement pstmt = null;
        Connection conn = null;
        ResultSet rs = null;
        int count = 0;
        try
        {
            conn = getConnection();
            if(conn == null)
            {
                return "New";
            }
            else
            {
                sqlStatement = "exec sp_getOrderStatus ?";
                //sqlStatement = "select * from vta_EnT_Orders where
P_Ord_ID = ?";
                pstmt = conn.prepareStatement(sqlStatement);
                pstmt.setInt(1,OrdID);
                rs = pstmt.executeQuery();
                while(rs.next())
                {
                    status = rs.getString("status");
                    count += 1;
                }
                if(count > 0)
                {
                    return status;
                }
                else
                {
                    return "None";
                }

            }
        }
        catch (SQLException x)
        {
            throw new RemoteException("Can not Query table", x);
            //return "null";
        }
        finally
        {
            try
            {
                if(pstmt != null)
                    pstmt.close();
            }
            catch (Exception ex) { }
```

```
            try
            {
                if(conn != null)
                    conn.close();
            }
            catch (Exception ex) { }
        }
    }

    /**
     * @common:operation
     */
    public String FindPackages(String loc) throws SQLException,
RemoteException, NamingException
    {
        PreparedStatement pstmt = null;
        Connection conn = null;
        String sqlStatement = null;
        int type = 1;
        try
        {
            conn = getConnection();
            if(loc !="")
            {
               sqlStatement = "exec sp_query_packages ?, ?";
               pstmt = conn.prepareStatement(sqlStatement);
               if(type == 1 || type ==2)
               {
                  pstmt.setInt(1,type);
               }
               else
               {
                  pstmt.setInt(1,6);
               }
               pstmt.setString(2,loc);
            }
            ResultSet rs = pstmt.executeQuery();
            //Vector packages = new Vector();
            //Package[] packages;
            //Package epk;
            StringBuffer sb = new StringBuffer();
            sb.append("\n");
            while (rs.next())
            {
                //try
                //{
                //    epk = new Package(rs.getString("P_Name"),
rs.getString("P_Company"), rs.getString("P_E_Date"));
               // }
                //catch (SQLException x)
                //{
                //    throw new RemoteException("Can not getDate", x);
                    //return "null";
                //}
                try
                {
                    sb.append("  <ns:package>  \n");
```

```
                    sb.append("    <ns:P_ID>" +
rs.getInt("P_ID")+"</ns:P_ID>\n");
                    sb.append("    <ns:Package_name>"
+rs.getString("P_Name")+"</ns:Package_name>\n");
                    sb.append("    <ns:Company>"
+rs.getString("P_Company")+"</ns:Company>\n");
                    sb.append("    <ns:E_Date>"
+rs.getString("P_E_Date")+"</ns:E_Date>\n");
                    sb.append("
<ns:LocationID>"+rs.getString("P_LocationID")+"</ns:LocationID>\n");
                    sb.append("
<ns:ContentID>"+rs.getString("P_ContentID")+"</ns:ContentID>\n");
                    sb.append("    <ns:price>"
+rs.getDouble("P_Price")+"</ns:price>\n");
                    sb.append("  </ns:package>\n");
                }
                catch (SQLException x)
                {
                    throw new RemoteException("Can not get type right!",
x);
                    //return "null";
                }
                //packages.add(epk);

            }
            sb.append("\n");
            return sb.toString();
        }
        catch (SQLException x)
        {
            throw new RemoteException("Can not find Packages", x);
            //return "null";
        }
        finally
        {
            try
            {
                if(pstmt != null)
                    pstmt.close();
            }
            catch (Exception ex) { }
            try
            {
                if(conn != null)
                    conn.close();
            }
            catch (Exception ex) { }
        }
    }

    /**
     * @common:operation
     */
    public String GetMyOrders(String Order_Date) throws SQLException,
RemoteException, NamingException
    {
        PreparedStatement pstmt = null;
```

```
        Connection conn = null;
        String sqlStatement = null;
        try
        {
            conn = getConnection();
            if(Order_Date !="")
            {
               sqlStatement = "exec sp_getAllorder ?";
               pstmt = conn.prepareStatement(sqlStatement);
               pstmt.setString(1,Order_Date);
            }
            ResultSet rs = pstmt.executeQuery();
            return writeToString(rs);
        }
        catch (SQLException x)
        {
            throw new RemoteException("Can not find Packages", x);
            //return "null";
        }
        finally
        {
            try
            {
                if(pstmt != null)
                    pstmt.close();
            }
            catch (Exception ex) { }
            try
            {
                if(conn != null)
                    conn.close();
            }
            catch (Exception ex) { }
        }

    }

    //Private memebers
    private Connection getConnection() throws SQLException,
RemoteException, NamingException
    {

         try
         {

            Driver myDriver =
(Driver)Class.forName("weblogic.jdbc.mssqlserver4.Driver").newInstance();

            String drURL = "jdbc:weblogic:mssqlserver4";
            Properties locprop = new Properties();
            locprop.put("server", "esszz");
            locprop.put("db", "EntProvider");
            locprop.put("user", "ent_web");
            locprop.put("password", "ent666");

            Connection conn = myDriver.connect(drURL, locprop);
            return conn;
```

```
         }
        catch(SQLException e)
        {
            System.out.println("SQLException");
            return null;
        }
        catch(ClassNotFoundException c)
        {
            System.out.println("ClassNotFoundException");
            return null;
        }
        catch(InstantiationException In)
        {
            System.out.println("InstantiationException");
            return null;
        }
        catch(IllegalAccessException Il)
        {
            System.out.println("IllegalAccessException");
            return null;
        }


   }

   private String writeToString(ResultSet rs) throws SQLException,
RemoteException, NamingException
   {
        StringBuffer sb = new StringBuffer();
        sb.append("\n");
        while(rs.next())
        {
            sb.append("  <ns:order>  \n");
            sb.append("    <ns:Order_number>" +
rs.getInt(1)+"</ns:Order_number>\n");
            sb.append("    <ns:Package_name>"
+rs.getString(13)+"</ns:Package_name>\n");
            sb.append("    <ns:Order_date>"
+rs.getDate(3)+"</ns:Order_date>\n");
            sb.append("    <ns:Customer>"
+rs.getString(7)+"</ns:Customer>\n");
            sb.append("
<ns:Status>"+rs.getString(12)+"</ns:Status>\n");
            sb.append("    <ns:price>" +rs.getDouble(11)+"</ns:price>\n");
            sb.append("  </ns:order>\n");
        }
        sb.append("\n");
        return sb.toString();
   }

}
```

APPENDIX II

Registration Source Code

RegisterWSDL.cs

```
using System;
using System.IO;
using System.Xml;
using System.Xml.Schema;
using System.Xml.Serialization;
using System.Data;
using System.Text;
using System.Data.SqlClient;
using Tamir.SharpSsh;

namespace WSRegister.ClassLb
{
   /// <summary>
   /// RegisterWSDL will register new or updated web service wsdl info.
   /// </summary>
   public class RegisterWSDL
   {
         #region private member variables
         /*****************************************************/
         private String _result;
         private bool _new;
         private int _languageID;
         /*****************************************************/
         #endregion private member variables

         #region properties
         /*****************************************************/
         public string Result{get{return _result;} set{_result = value;}}
         public bool New{get{return _new;}set{_new = value;}}
         public int LanguageID{get{return _languageID;} set{_languageID =
value;}}
         /*****************************************************/
         #endregion properties

         #region constructors
         /*****************************************************/
         public RegisterWSDL()
         {
               _result = "";
         }
         /*****************************************************/
         #endregion constructors
```

```
        #region public methods
        /*************************************************************/
        public WsdlList Get_WSList(int providerID)
        {
              WsdlList wsdls = new WsdlList();
              try
              {
                    DBAccess dba = new DBAccess();
                    SqlDataReader rd;

                    rd = dba.exec_sql("exec sp_getUDDIByProvider " +
providerID.ToString());
                    while(rd.Read())
                    {
                          Wsdl wsdl = new Wsdl();
                          wsdl.wid = rd.GetInt32(0);
                          wsdl.Location = rd.GetString(3);
                          wsdls.Add(wsdl);
                    }
                    return wsdls;
              }
              catch(Exception ex)
              {
                    throw new Exception(ex.Message);
              }
        }
        public MethodList Get_MDList(int wsID)
        {
              MethodList methods = new MethodList();
              try
              {
                    DBAccess dba = new DBAccess();
                    SqlDataReader rd;

                    rd = dba.exec_sql("exec sp_get_methodsbyWsID " +
wsID.ToString());
                    while(rd.Read())
                    {
                          Method method = new Method();
                          method.methodID = rd.GetInt32(0);
                          method.name = rd.GetString(1);
                          method.description = rd.GetString(2);
                          method.client = new Client();
                          method.client.name = rd.GetString(5);
                          methods.Add(method);
                    }
                    return methods;
              }
              catch(Exception ex)
              {
                    throw new Exception(ex.Message);
              }
        }
        public MethodList Get_MDList(string wsUri)
        {
              MethodList methods = new MethodList();
              try
```

```
            {
                  DBAccess dba = new DBAccess();
                  SqlDataReader rd;

                  rd = dba.exec_sql("exec sp_get_methodsbyWsUri '" +
wsUri + "'");
                  while(rd.Read())
                  {
                        Method method = new Method();
                        method.methodID = rd.GetInt32(0);
                        method.name = rd.GetString(1);
                        method.description = rd.GetString(2);
                        method.client = new Client();
                        method.client.name = rd.GetString(5);
                        methods.Add(method);
                  }
                  return methods;
            }
            catch(Exception ex)
            {
                  throw new Exception(ex.Message);
            }
      }
      public ParameterList Get_ParmaterList(int mtdID)
      {
            ParameterList pars = new ParameterList();
            try
            {
                  DBAccess dba = new DBAccess();
                  SqlDataReader rd;

                  rd = dba.exec_sql("exec sp_GetInParametersByPdtid " +
mtdID.ToString());
                  while(rd.Read())
                  {
                        Parameter par = new Parameter();
                        par.parmID = rd.GetInt32(0);
                        par.name = rd.GetString(3);
                        par.valueType = rd.GetString(9);
                        par.Active = rd.GetString(5);
                        if (rd.IsDBNull(4))
                        {
                              par.DefaultValue = "";
                        }
                        else
                        {
                              par.DefaultValue = rd.GetString(4);
                        }
                        pars.Add(par);
                  }
                  return pars;
            }
            catch(Exception ex)
            {
                  throw new Exception(ex.Message);
            }
      }
```

```
        public short update_method(Method _method)
        {
              DBAccess dba = new DBAccess();
              string filename = @"D:\wsproxy\mywscook_" + _method.name +
".sck";
              ClearCache(filename);
              return dba.exec_cmd("sp_upPXMethod " + _method.methodID +
", '" + _method.description + "'");
        }
        public short update_parameter(Parameter _parm)
        {
              DBAccess dba = new DBAccess();
              if (_parm.DefaultValue.Length > 0)
              {
                    return dba.exec_cmd("sp_upPXParametersIn " +
_parm.parmID + ", 0, '" + _parm.DefaultValue + "'");
              }
              else
              {
                    return dba.exec_cmd("sp_upPXParametersIn " +
_parm.parmID + ", 0");
              }
        }
        public bool Register_WS(string _wsdl)
        {
              //save new wsdl info to db
              try
              {
                    int pid;
                    MethodList methods;

                    //get parent id
                    pid = GetParentId(_wsdl);

                    //build new methodlist and save them
                    methods = CreateMethods(_wsdl);

                    //build new parameterlist and save them
                    if (RegisterMethods(methods, pid))
                    {
                          _result = "Registeration is successful!!";

                          //set new flg file, delete cache
                          ClearCache(GetCachePath(_wsdl));

                          return true;
                    }
                    else
                    {
                          _result = "Registeration is fail!!";
                          return false;
                    }
              }
              catch(Exception ex)
              {
                    throw new Exception(ex.Message);
              }
```

```
        }
        /*************************************************************/
        #endregion public methods

        #region private methods
        /*************************************************************/
        private int GetParentId(string wsdlUrl)
        {
              DBAccess dba = new DBAccess();
              SqlDataReader rd;
              string pname = "Ent Inc.";
              try
              {
                    if(_new) //add new one
                    {
                          DBAccess dbaLoc = new DBAccess();
                          SqlDataReader rdLoc;
                          rdLoc = dbaLoc.exec_sql("exec sp_insWSProfiler
1 , '" + pname + "', '" + wsdlUrl + "', 'new service for Ent',
'Provider'");
                          rdLoc.Read();
                          int wsid = rdLoc.GetInt32(0);

                          // check wsdl record first
                          rd = dba.exec_sql("exec sp_AddProxy " +
wsid.ToString() + ", 1" + ", 'VTAEnt', '1.0.0.0', " +
_languageID.ToString());

                    }
                    else
                    {
                          rd = dba.exec_sql("exec sp_get_proxybywsdl
@wsdlUrl= '" + wsdlUrl + "'");
                    }
                    //get Pid
                    if (rd.HasRows)
                    {
                          rd.Read();
                          return rd.GetInt32(0);
                    }
                    else
                    {
                          return 0;
                    }
              }
              catch(Exception ex)
              {
                    throw new Exception(ex.Message);
              }
        }

        private MethodList CreateMethods(string strwsdl)
        {
              string mname="";
              string prefix = "";
              XmlDocument xmldoc = new XmlDocument();
              xmldoc.Load(strwsdl);
```

```
            MethodList methods = new MethodList();

            if (xmldoc.GetElementsByTagName("s:schema").Count == 0)
            {
                  prefix = "x";
            }

            XmlNodeList _nodes = xmldoc.GetElementsByTagName(prefix +
"s:schema")[0].ChildNodes;

            Method method = new Method();
            try
            {
                  Method locmethod =  new Method();
                  foreach(XmlNode _node in _nodes)
                  {
                        if(_node.Name== (prefix + "s:element") &&
_node.FirstChild !=  null)
                        {
                              //in parameters
                              if(mname !=
_node.Attributes["name"].Value.Replace("Response",""))
                              {
                                    locmethod =  new Method();
                              }
                              if
(_node.Attributes["name"].Value.IndexOf("Response", 0) <= 0)
                              {
                                    ParameterList pars = new
ParameterList();

                                    locmethod.name =
_node.Attributes["name"].Value;
                                    mname = locmethod.name;
                                    if (_node.FirstChild.FirstChild !=
null)
                                    {
                                          foreach(XmlNode _inode in
_node.FirstChild.FirstChild.ChildNodes)
                                          {
                                                Parameter par = new
Parameter();
                                                par.valueType =
_inode.Attributes["type"].Value.Substring(_inode.Attributes["type"].Value.
IndexOf(":", 0)+1);
                                                if
(par.valueType.IndexOf("ArrayOf", 0) >=0)
                                                {
                                                      par.valueType =
par.valueType.Substring(par.valueType.IndexOf("ArrayOf", 0) + 7);
                                                      par.array = true;
                                                }
                                                else
                                                {
                                                      par.array =
false;
```

```
                                                }
                                                par.parmID =
GetTypeId(par.valueType);
                                                par.name =
_inode.Attributes["name"].Value;
                                                par.methodname =
locmethod.name;
                                                pars.Add(par);
                                          }
                                          locmethod.inValueTypes =
pars;
                                    }
                                    else
                                    {
                                          locmethod.inValueTypes =
null;
                                    }
                              }
                              else //return type
                              {
                                    //method.name =
_node.Attributes["name"].Value;
                                    Parameter par = new Parameter();

                                    par = new Parameter();
                                    XmlNode _inode =
_node.FirstChild.FirstChild.FirstChild;
                                    par.valueType =
_inode.Attributes["type"].Value.Substring(_inode.Attributes["type"].Value.
IndexOf(":", 0)+1);
                                    if
(par.valueType.IndexOf("ArrayOf", 0) >=0)
                                    {
                                          par.valueType =
par.valueType.Substring(par.valueType.IndexOf("ArrayOf", 0) + 7);
                                          locmethod.array = true;
                                    }
                                    else
                                    {
                                          locmethod.array = false;
                                    }
                                    par.parmID =
GetTypeId(par.valueType);
                                    par.name =
_inode.Attributes["name"].Value;
                                    par.methodname = method.name;
                                    locmethod.outValueType = par;
                                    //after return type, collect the
method
                                    methods.Add(locmethod);
                              }

                        }
                        else
                        {
                              //complex type
                        }
```

```
                }

                return methods;
            }
            catch(Exception ex)
            {
                throw new Exception(ex.Message);
            }
        }

        private bool RegisterMethods(MethodList _methods, int _parentId)
//save it to db
        {
            int ar, rs=0, pdit=0, Inparmin = 0;
            SqlDataReader read;
            //save _methods
            try
            {
                foreach(Method mtd in _methods)
                {
                    DBAccess dba = new DBAccess();
                    ar = 0;
                    if (mtd.array)
                    {
                        ar = 1;
                    }
                    read = dba.exec_sql("exec sp_addPXMethod " +
_parentId + ", '" + mtd.name + "', " + mtd.outValueType.parmID + "," + ar
+ ",'" + mtd.description + "'");
                    read.Read();
                    pdit = read.GetInt32(0);
                    read = null;
                    //save params
                    if (mtd.inValueTypes != null)
                    {
                        foreach(Parameter pmt in
mtd.inValueTypes)
                        {
                            DBAccess dbaloc = new DBAccess();
                            ar = 0;
                            if (pmt.array)
                            {
                                ar = 1;
                            }
                            read = dbaloc.exec_sql("exec
sp_AddPXParametersIn " + pdit + ", '" + pmt.name + "', " + pmt.parmID +
"," + ar);
                            if (read.HasRows)
                            {
                                read.Read();
                                Inparmin = read.GetInt32(0);
                                read = null;
                            }
                        }
                        //validate methods
                        DBAccess dbalc = new DBAccess();
```

```
                                    rs = dbalc.exec_cmd("exec
sp_validate_child_PXParametersIn " + pdit);
                              }
                        }
                        if (Inparmin >= 0 && pdit > 0 && rs == 1)
                          return true;
                        else
                          return false;
                  }
                  catch(Exception ex)
                  {
                        throw new Exception(ex.Message);
                  }
            }

            private int GetTypeId(string type)
            {
                  SqlDataReader read;
                  try
                  {
                        DBAccess dba = new DBAccess();
                        read = dba.exec_sql("exec sp_getValuetype '" + type +
"'");
                        read.Read();
                        return read.GetInt32(0);
                  }
                  catch(Exception ex)
                  {
                        throw new Exception(ex.Message);
                  }
            }
            private bool ClearCache(string _filepath)
            {
                  // clear the cached assembly file for this WSDL
                  try
                  {
                        Scp cp = new Scp();
                        cp.To(@"C:\temp\NewMethod.flg", "esszz",
@"D:\wsproxy\NewMethod.flg", "Administrator", "zz888888");
                        //clear cach
                        return ssh_command("esszz", "Administrator",
"zz888888", "del " + _filepath);
                  }
                  catch(Exception e)
                  {
                        throw new Exception(e.Message);
                  }
            }

            private string GetCachePath(string wsdlURL)
            {
                  string lfilepath="";
                  DBAccess dba = new DBAccess();
                  SqlDataReader rd = dba.exec_sql("sp_get_proxybywsdl",
"@wsdlurl", "string", wsdlURL);
                  while(rd.Read())
                  {
```

```
                lfilepath = rd.GetString(6);
            }
            return lfilepath;
        }

        private bool CreateSCookie(string methodName, string filename,
string wsdlUrl)
        {
            bool Inp = false;
            StringBuilder sb = new StringBuilder();
            string vtype = "";
            sb.Append("<Method>\n");
            sb.Append("  <" + methodName + ">\n");
            DBAccess dba = new DBAccess();
            SqlDataReader rd = dba.exec_sql("exec sp_GetWSMethod
@wsUrl= '" + wsdlUrl + "', @name='" + methodName + "'");
            if (rd.HasRows)
            {
                //methodParams.Clear();
                DBAccess dbaloc = new DBAccess();
                rd.Read();
                int Pdtid = rd.GetInt32(4);
                SqlDataReader rdp = dbaloc.exec_sql("exec
sp_GetInParametersByPdtid @Pdtid=" + Pdtid.ToString());
                sb.Append("    <params>\n");
                while(rdp.Read())
                {
                    if (!rdp.IsDBNull(4))
                    {
                        vtype = rdp.GetString(4);
                    }
                    //build xml string
                    sb.Append("      <param name=\"" +
rdp.GetString(3) + "\"" + " active=\"" + rdp.GetString(5) + "\"" + "
vtype=\"" + rdp.GetString(9) + "\"" + " dvalue=\"");
                    if (rdp.GetString(5) == "N")
                    {
                        sb.Append(vtype);
                    }
                    sb.Append("\"></param>\n");
                }
                sb.Append("    </params>\n");
                Inp  = true;
            } //if
            sb.Append("  </" + methodName + ">\n");
            sb.Append("</Method>\n");
            if (Inp)
            {
                Inp = creatfile(filename, sb.ToString());
            }
            return Inp;
        }
        private bool creatfile(string path, string strVal)
        {
            try
            {
```

```
                    FileStream sb = new FileStream(@"C:\temp\method.sck",
FileMode.OpenOrCreate);
                    StreamWriter sw = new StreamWriter(sb);
                    sw.Write(strVal);
                    sw.Close();
                    //copy to client location
                    Scp cp = new Scp();
                    cp.To(@"C:\temp\method.sck", "esszz", path,
"Administrator", "zz888888");
                    //clear cach
                    return true;
              }
              catch(Exception ex)
              {
                    throw new Exception(ex.Message);
              }
        }
        private string GetWsdl(string WsdlUri)
        {
              Wsdl mwsdl = new Wsdl();
              return mwsdl.GetWsdlFromUri(WsdlUri);
        }
        private bool ssh_command(string host, string username, string
pwd, string command)
        {
              try
              {
                    SshStream ssh = new SshStream(host, username, pwd);
                    //Sets the end of response character
                    ssh.Prompt = "#";
                    //Remove terminal emulation characters
                    ssh.RemoveTerminalEmulationCharacters = true;
                    ssh.Write( command );
                    ssh.Close(); //Close the connection
                    return true;
              }
              catch(Exception ex)
              {
                    throw new Exception(ex.Message);
              }

        }
        /*************************************************************/
        #endregion private methods

   } //class

} //namespace
```

RegisterClient.cs

```
using System;
using System.IO;

namespace WSRegister.ClassLb
{
   /// <summary>
   /// Summary description for RegisterClient.
   /// </summary>
   public class RegisterClient
   {
         #region private member variables
         /*************************************************************/
         private string _FlgLocation;
         private string _SCKLocation;
         private int _languageID;
         /*************************************************************/
         #endregion private member variables

         #region properties
         /*************************************************************/
         public string FlgLocation{get{return _FlgLocation;}
set{_FlgLocation = value;}}
         public string SCKLocation{get{return
_SCKLocation;}set{_SCKLocation = value;}}
         public int LanguageID{get{return _languageID;} set{_languageID =
value;}}
         /*************************************************************/
         #endregion properties

         public RegisterClient()
         {
               //
               // TODO: Add constructor logic here
               //
         }
         public bool check()
         {
               return lcheck(_FlgLocation);
         }
         public bool check(string _flgloc)
         {
               _FlgLocation = _flgloc;
               return lcheck(_flgloc);
         }

         public bool update()
         {
               return lupdate(_FlgLocation, _SCKLocation);
         }
         public bool update(string _flgloc, string _sckloc)
         {
               _FlgLocation = _flgloc;
               _SCKLocation = _sckloc;
               return lupdate(_flgloc, _sckloc);
```

```
        }

        private bool lupdate(string _flgloc, string _sckloc)
        {
              try
              {
                    if(File.Exists(_flgloc))
                    {
                          File.Delete(_flgloc);
                    }
                    if(File.Exists(_sckloc))
                    {
                          File.Delete(_sckloc);
                    }
                    return true;
              }
              catch(Exception ex)
              {
                    throw new Exception(ex.Message);
              }
        }
        private bool lcheck(string _flgloc)
        {
              return File.Exists(_flgloc);
        }
   }
}
```

Client.cs

```
using System;

namespace WSRegister.ClassLb
{
   /// <summary>
   /// Summary description for Client.
   /// </summary>
   public class Client
   {
         #region private member variables
         /*************************************************************/
         // constant string to build stub

         private String _name;
         private int _id;
         private string _address;
         private string _filelocation;
         private string _scklocation;
         private bool _active;
         private DateTime _cdte;
         private DateTime _udte;

         /*************************************************************/
         #endregion private member variables

         #region constructors
         /*************************************************************/
         public Client()
         {
               //
               // TODO: Add constructor logic here
               //
         }
         #endregion constructors

         #region properties
         /*************************************************************/
         public string name{get{return _name;} set{_name = value;}}
         public int clientID{get{return _id;} set{_id = value;}}
         public string address{get{return _address;}set{_address =
value;}}
         public bool active{get{return _active;}set{_active = value;}}
         public string filelocation{get{return _filelocation;}
set{_filelocation = value;}}
         public string scklocation{get{return _scklocation;}
set{_scklocation = value;}}
         public DateTime created_date{get{return _cdte;} set{_cdte =
value;}}
         public DateTime updated_date{get{return _udte;} set{_udte =
value;}}
         /*************************************************************/
         #endregion properties
   }
}
```

Method.cs

```
using System;

namespace WSRegister.ClassLb
{
   /// <summary>
   /// Summary description for Method.
   /// </summary>
   public class Method
   {
         #region private member variables
         /*****************************************************************/
         // constant string to build stub

         private String _name;
         private int _id;
         private Client _client;
         private string _description;
         //private string _proxyname;
         private bool _active;
         private bool _array;
         private Parameter _outValueType;
         private ParameterList _inValueTypes;
         private DateTime _cdte;
         private DateTime _udte;

         /*****************************************************************/
         #endregion private member variables

         #region constructors
         /*****************************************************************/
         public Method()
         {
               //
               // TODO: Add constructor logic here
               //
         }
         #endregion constructors

         #region properties
         /*****************************************************************/
         public string name{get{return _name;} set{_name = value;}}
         public int methodID{get{return _id;} set{_id = value;}}
         public Client client{get{return _client;} set{_client = value;}}
         public string description{get{return
_description;}set{_description = value;}}
         public bool active{get{return _active;}set{_active = value;}}
         public bool array{get{return _array;}set{_array = value;}}
         public Parameter outValueType{get{return _outValueType;}
set{_outValueType = value;}}
         public ParameterList inValueTypes{get{return _inValueTypes;}
set{_inValueTypes = value;}}
         public DateTime created_date{get{return _cdte;} set{_cdte =
value;}}
```

```
        public DateTime updated_date{get{return _udte;} set{_udte =
value;}}

        /**************************************************************/
        #endregion properties


    }
}
```

MethodList.cs

```
using System;
using System.Collections;

namespace WSRegister.ClassLb
{
   /// <summary>
   /// Summary description for MethodList.
   /// </summary>
   public class MethodList : CollectionBase
   {
         public Method this[ int index ]
         {
               get
               {
                     return( (Method) List[index] );
               }
               set
               {
                     List[index] = value;
               }
         }

         public void Add(Method value)
         {
               List.Add(value);
         }
         public void Remove(Method value)
         {
               List.Remove(value);
         }
         public int IndexOf(Method value)
         {
               return( List.IndexOf(value) );
         }
         public void Insert( int index, Method value )
         {
               List.Insert( index, value );
         }
   }
}
```

Parameter.cs

```
using System;

namespace WSRegister.ClassLb
{
   /// <summary>
   /// Summary description for Parameter.
   /// </summary>
   public class Parameter
   {
         #region private member variables
         /*************************************************************/
         // constant string to build stub

         private string _name;
         private int _id;
         private string _active;
         private bool _array;
         private string _methodname;
         private string _inValueType;
         private string _inDefaultValue;
         private DateTime _cdte;
         private DateTime _udte;

         /*************************************************************/
         #endregion private member variables

         #region constructors
         /*************************************************************/
         public Parameter()
         {
               //
               // TODO: Add constructor logic here
               //
         }
         #endregion constructors

         #region properties
         /*************************************************************/
         public string name{get{return _name;} set{_name = value;}}
         public int parmID{get{return _id;} set{_id = value;}}
         public string Active{get{return _active;} set{_active = value;}}
         public bool array{get{return _array;}set{_array = value;}}
         public string methodname{get{return _methodname;}set{_methodname
= value;}}
         public string valueType{get{return _inValueType;}set{_inValueType
= value;}}
         public string DefaultValue{get{return
_inDefaultValue;}set{_inDefaultValue = value;}}
         public DateTime created_date{get{return _cdte;} set{_cdte =
value;}}
         public DateTime updated_date{get{return _udte;} set{_udte =
value;}}

         /*************************************************************/
```

```
        #endregion properties
    }
}
```

ParameterList.cs

```
using System;
using System.Collections;

namespace WSRegister.ClassLb
{
   /// <summary>
   /// Summary description for ParameterList.
   /// </summary>
   public class ParameterList : CollectionBase
   {
         public Parameter this[ int index ]
         {
               get
               {
                     return( (Parameter) List[index] );
               }
               set
               {
                     List[index] = value;
               }
         }

         public void Add(Parameter value)
         {
               List.Add(value);
         }
         public void Remove(Parameter value)
         {
               List.Remove(value);
         }
         public int IndexOf(Parameter value)
         {
               return( List.IndexOf(value) );
         }
         public void Insert( int index, Parameter value )
         {
               List.Insert( index, value );
         }
   }
}
```

Wsdl.cs

```
using System;
using System.IO;
using System.Net;
using System.Web.Services.Description;
using System.Web.Services.Discovery;
using System.Web.Services.Protocols;
using System.Collections;
using System.Text;
using System.Xml;
using System.Diagnostics;
using System.Data;
using System.Data.SqlClient;


namespace WSRegister.ClassLb
{
    /// <summary>
    /// Summary description for Wsdl.
    /// </summary>
    public class Wsdl
    {

        #region private member variables
        /*****************************************************************/
        // constant string to build stub

        private String _name;
        private String _description;
        private String _location;
        private String _version;
        private String _binding;
        private bool _updated;
        private String _filepath;
        private int _wid;
        private DateTime _updateTime;
        private MethodList _methodlist;
        private ParameterList _parameterlist;

        /*****************************************************************/
        #endregion private member variables

        #region constructors
        /*****************************************************************/
        public Wsdl()
        {
            //no thing
        }
        public Wsdl(int wid)
        {
            _wid = wid;
            DBAccess dba = new DBAccess();
            SqlDataReader rd = dba.exec_sql("sp_get_wsdlbyID", "@id",
"int", wid);
            _name = rd.GetString(1);
```

```
                _description = "N/A";
                _location = rd.GetString(3);
                _updated = rd.GetBoolean(7);
                _version = rd.GetString(8);

        }
        public Wsdl(string wsUrl)
        {
                _location = wsUrl;
                DBAccess dba = new DBAccess();
                SqlDataReader rd = dba.exec_sql("sp_get_wsdlbyWsUrl",
"@wsURL", "string", wsUrl);
                _wid = rd.GetInt32(0);
                _name = rd.GetString(1);
                _description = "N/A";
                _updated = rd.GetBoolean(7);
                _version = rd.GetString(8);

        }
        public Wsdl(int wid, string name)
        {
                _wid = wid;
                _name = name;
        }
        #endregion constructors

        #region detructors
        /*************************************************************/
        ~Wsdl()
        {
                //_sdr.Close();
        }
        #endregion destructors

        #region properties
        /*************************************************************/
        public String name{get{return _name;} set{_name = value;}}
        public String description{get{return
_description;}set{_description = value;}}
        public String Location{get{return _location;}set{_location =
value;}}
        public String Filepath{get{return _filepath;}set{_filepath =
value;}}
        public bool Updated{get{return _updated;} set{_updated = value;}}
        public String Version{get{return _version;} set{_version =
value;}}
        public MethodList MethodList{get{return _methodlist;}
set{_methodlist = value;}}
        public ParameterList ParameterList{get{return _parameterlist;}
set{_parameterlist = value;}}
        public String BindingType{get{return _binding;} set{_binding =
value;}}
        public int wid {get{return _wid;} set{_wid = value;}}
        public DateTime UpdateTime
        {
                get{ return _updateTime;}
                set{_updateTime = value;}
```

```
        }

        /*************************************************************/
        #endregion properties

        #region public methods
        /*************************************************************/

        public string Get_WSDL()
        {
              return Get_WSDL(_location+"?wsdl");
        }
        public string Get_WSDL(string location)
        {
              XmlDocument xmldoc = get_wsdlDoc(location);

              string wsdlStub = xmldoc.ToString();

              return wsdlStub;
        }
        public bool save_wsdl_file(string wsdlUri)
        {
              try
              {
                    StreamWriter Swtr = new StreamWriter("/DYInvoker/"+
name + ".wsdl");
                    Swtr.Write(GetWsdlFromUri(wsdlUri));
                    return true;
              }
              catch(Exception ex)
              {
                    throw new Exception(ex.Message);
              }
        }
        public string GetWsdlFromUri(string uri)
        {
              _location = uri;
              WebRequest req = WebRequest.Create(uri);
              WebResponse result = req.GetResponse();

              Stream ReceiveStream = result.GetResponseStream();
              Encoding encode = System.Text.Encoding.GetEncoding("utf-
8");
              StreamReader sr = new StreamReader(ReceiveStream, encode);

              string wsdlSource = sr.ReadToEnd();
              sr.Close();
              //Debug.Write(wsdlSource);
              return wsdlSource;
        }

        public string GetWsdlFromFile(string fileFullPathName)
        {
              FileInfo fi = new FileInfo(fileFullPathName);

              if(fi.Extension == "wsdl")
              {
```

```
                    FileStream fs = new FileStream(fileFullPathName,
FileMode.Open, FileAccess.Read);
                    StreamReader sr = new StreamReader(fs);

                    char[] buffer = new char[(int)fs.Length];
                    sr.ReadBlock(buffer, 0, (int)fs.Length);
                    sr.Close();
                    _filepath = fileFullPathName;
                    return new string(buffer);
              }

              throw new Exception("This is not a WSDL file");
        }

        /*****************************************************************/
        #endregion public methods

        #region private methods
        /*****************************************************************/
        private XmlDocument get_wsdlDoc(string location)
        {
              try
              {
                    XmlDocument xmldoc = new XmlDocument();
                    xmldoc.Load(location + "?wsdl");
                    return xmldoc;
              }
              catch(Exception ex)
              {
                    throw new Exception(ex.Message);
              }
        }

        /*****************************************************************/
        #endregion private methods
   }
}
```

WsdlList.cs

```
using System;
using System.Collections;
namespace WSRegister.ClassLb
{
   /// <summary>
   /// Summary description for WsdlList.
   /// </summary>
   public class WsdlList : CollectionBase
   {
         public Wsdl this[ int index ]
         {
               get
               {
                     return( (Wsdl) List[index] );
               }
               set
               {
                     List[index] = value;
               }
         }

         public void Add(Wsdl value)
         {
               List.Add(value);
         }
         public void Remove(Wsdl value)
         {
               List.Remove(value);
         }
         public int IndexOf(Wsdl value)
         {
               return( List.IndexOf(value) );
         }
         public void Insert( int index, Wsdl value )
         {
               List.Insert( index, value );
         }

   }
}
```

WebService.aspx

```
<%@ Page language="c#" Codebehind="WebService.aspx.cs" AutoEventWireup="false"
Inherits="WSRegister.WebForm1" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <title>Provider</title>
        <meta content="Microsoft Visual Studio .NET 7.1" name="GENERATOR">
        <meta content="C#" name="CODE_LANGUAGE">
        <meta content="JavaScript" name="vs_defaultClientScript">
        <meta content="http://schemas.microsoft.com/intellisense/ie5"
name="vs_targetSchema">
    </HEAD>
    <body>
        <form id="Form1" method="post" runat="server">
            <div align="center">
                <table width="80%">
                    <tr>
                        <td align="center"><STRONG><FONT
color="#0033cc" size="5">Web Service Registeration</FONT></STRONG></td>
                    </tr>
                    <tr>
                        <td align="center">
                            <table width="100%">
                                <tr>
                                    <td style="WIDTH: 85px"
align="right"><FONT size="2"><STRONG>*WSDL Url:</STRONG></FONT></td>
                                    <td><asp:textbox
id="txtwsdlUrl" runat="server" Width="513px"></asp:textbox></td>
                                </tr>
                                <tr>
                                    <td></td>
                                    <td>

    <asp:RequiredFieldValidator id="Reqvt" runat="server" Font-Size="XX-Small"
ErrorMessage="Please Fill WSDL Url"

    Display="Dynamic"
ControlToValidate="txtwsdlUrl"></asp:RequiredFieldValidator></td>
                                </tr>
                                <tr>
                                    <td style="WIDTH: 85px"
align="right"><FONT size="2"><STRONG>Language:</STRONG></FONT></td>
                                    <td>
```

```
                                                    <asp:DropDownList
id="ddLanguage" runat="server">
                                                        <asp:ListItem
Value="1">C#</asp:ListItem>
                                                        <asp:ListItem
Value="2">VB.Net</asp:ListItem>
                                                        <asp:ListItem
Value="3">Java</asp:ListItem>

    </asp:DropDownList></td>
                                                </tr>
                                                <tr>
                                                    <td style="WIDTH: 85px;
HEIGHT: 16px" align="right"><FONT
size="2"><STRONG>Version:</STRONG></FONT></td>
                                                    <td style="HEIGHT: 16px">
                                                        <asp:RadioButtonList
id="Rad_Version" runat="server" RepeatDirection="Horizontal" Font-Size="X-Small"

    AutoPostBack="True" Height="8px">
                                                        <asp:ListItem
Value="U" Selected="True">Updated</asp:ListItem>
                                                        <asp:ListItem
Value="N">New</asp:ListItem>

    </asp:RadioButtonList></td>
                                                </tr>
                                                <tr>
                                                    <td style="WIDTH:
85px"></td>
                                                    <td>
                                                        <asp:Button
id="But_submit" runat="server"
Text="Register"></asp:Button>        &
nbsp;           
                                                        <INPUT type="reset"
value="Reset"></td>
                                                </tr>
                                            </table>
                                            <asp:Label id="lblFeedback" runat="server"
Font-Size="X-Small"></asp:Label></td>
                                        </tr>
                                        <tr>
                                            <td></td>
                                        </tr>
                                    </table>
```

```
                    </div>
                </form>
    </body>
</HTML>
```

WebService.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace WSRegister
{
   /// <summary>
   /// Summary description for WebForm1.
   /// </summary>
   public class WebForm1 : System.Web.UI.Page
   {
         protected System.Web.UI.WebControls.Button But_submit;
         protected System.Web.UI.WebControls.Label lblFeedback;
         protected System.Web.UI.WebControls.RadioButtonList Rad_Version;
         protected System.Web.UI.WebControls.RequiredFieldValidator Reqvt;
         protected System.Web.UI.WebControls.TextBox txtwsdlUrl;
         protected System.Web.UI.WebControls.DropDownList ddLanguage;

         private void Page_Load(object sender, System.EventArgs e)
         {
               // Put user code to initialize the page here
               lblFeedback.Text = "";
               if (!Page.IsPostBack)
               {
                     Rad_Version.SelectedIndex = 0;
               }
         }

         #region Web Form Designer generated code
         override protected void OnInit(EventArgs e)
         {
               //
               // CODEGEN: This call is required by the ASP.NET Web Form
Designer.
               //
               InitializeComponent();
               base.OnInit(e);
         }

         /// <summary>
         /// Required method for Designer support - do not modify
         /// the contents of this method with the code editor.
         /// </summary>
         private void InitializeComponent()
         {
               this.But_submit.Click += new
System.EventHandler(this.But_submit_Click);
```

```
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion

        private void But_submit_Click(object sender, System.EventArgs e)
        {
            ClassLb.RegisterWSDL reg = new ClassLb.RegisterWSDL();
            if (Rad_Version.SelectedIndex ==0)
                reg.New = false;
            else
                reg.New = true;
            reg.LanguageID = Convert.ToInt16(ddLanguage.SelectedValue);
            reg.Register_WS(txtwsdlUrl.Text);
           lblFeedback.Text = reg.Result;
        }

    }
}
```

Servicemaintian.aspx

```
<%@ Page language="c#" Codebehind="Servicemaintian.aspx.cs"
AutoEventWireup="false" Inherits="WSRegister.Servicemaintian"
ValidateRequest="false"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <title>Servicemaintian</title>
        <meta name="GENERATOR" Content="Microsoft Visual Studio .NET 7.1">
        <meta name="CODE_LANGUAGE" Content="C#">
        <meta name="vs_defaultClientScript" content="JavaScript">
        <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
    </HEAD>
    <body MS_POSITIONING="GridLayout">
        <form id="Form1" method="post" runat="server">
            <div align="center">
                <table width="66%">
                    <tr>
                        <td colspan="3"
align="center"><STRONG><FONT color="#000099">Web Service
Maintenance</FONT></STRONG></td>
                    </tr>
                    <tr>
                        <td align="right" width="30%">
                            <asp:Label id="lblWSDL" runat="server"
Font-Bold="True" Font-Size="Smaller">WSDL Location:</asp:Label></td>
                        <td colspan="2" width="70%">
                            <asp:TextBox id="txtwsdl" runat="server"
Width="100%"></asp:TextBox></td>
                    </tr>
                    <tr>
                        <td align="right" colspan="3">
                            <asp:Button id="bt_next" runat="server"
Text="Next"></asp:Button>
                        </td>
                    </tr>
                </table>
                <table width="90%">
                    <tr>
                        <td><b><font color="#006699">Method
List</font></b></td>
                    </tr>
                    <tr>
                        <td align="center">
```

```
                                        <asp:DataGrid id="dgmethos"
runat="server" Width="100%" AutoGenerateColumns="False">
                                            <SelectedItemStyle
BackColor="#CC99FF"></SelectedItemStyle>
                                            <AlternatingItemStyle
Wrap="False"></AlternatingItemStyle>
                                            <HeaderStyle Font-Bold="True"
ForeColor="White" BackColor="Maroon"></HeaderStyle>
                                            <Columns>
                                                <asp:BoundColumn
Visible="False" DataField="methodID" ReadOnly="True"
HeaderText="methodID"></asp:BoundColumn>
                                                <asp:BoundColumn
DataField="name" HeaderText="Name"></asp:BoundColumn>
                                                <asp:TemplateColumn
HeaderText="Client">
                                                    <ItemTemplate>
                                                        <asp:Label id
="lblclient" Width="100%" text='<%# DataBinder.Eval(Container.DataItem, "Client.name")
%>' Runat ="server"/>
                                                    </ItemTemplate>
                                                </asp:TemplateColumn>
                                                <asp:TemplateColumn
HeaderText="Description">
                                                    <ItemTemplate>
                                                        <asp:TextBox
id ="txtdesc" Width="100%" text='<%# DataBinder.Eval(Container.DataItem, "description")
%>' Runat ="server"/>
                                                    </ItemTemplate>
                                                </asp:TemplateColumn>
                                                <asp:ButtonColumn
Text="Select" CommandName="Select"></asp:ButtonColumn>
                                            </Columns>
                                        </asp:DataGrid>
                                    </td>
                                </tr>
                            </table>
                            <table width="90%">
                                <tr>
                                    <td><b><font color="#006699">Paramter
List</font></b></td>
                                </tr>
                                <tr>
                                    <td align="center">
                                        <asp:DataGrid id="dgparams"
runat="server" Width="100%" AutoGenerateColumns="False">
```

```
                                                    <SelectedItemStyle
BackColor="#CC99FF"></SelectedItemStyle>
                                                    <AlternatingItemStyle
Wrap="False"></AlternatingItemStyle>
                                                    <HeaderStyle Font-Bold="True"
ForeColor="#C0C0FF" BackColor="Blue"></HeaderStyle>
                                                    <Columns>
                                                        <asp:BoundColumn
Visible="False" DataField="parmID" HeaderText="ParmID"></asp:BoundColumn>
                                                        <asp:BoundColumn
DataField="name" HeaderText="Name"></asp:BoundColumn>
                                                        <asp:BoundColumn
DataField="valueType" HeaderText="DataType"></asp:BoundColumn>
                                                        <asp:BoundColumn
DataField="Active" HeaderText="Active"></asp:BoundColumn>
                                                        <asp:BoundColumn
DataField="DefaultValue" HeaderText="Default Value"
Visible="False"></asp:BoundColumn>
                                                        <asp:TemplateColumn
HeaderText="Default Value">
                                                            <ItemTemplate>
                                                                <asp:TextBox
id =txtDVal Width="100%" text='<%# DataBinder.Eval(Container.DataItem,
"DefaultValue") %>' Runat ="server"/>
                                                            </ItemTemplate>
                                                        </asp:TemplateColumn>
                                                    </Columns>
                                                </asp:DataGrid>
                                            </td>
                                        </tr>
                                    </table>
                                    <table width="90%">
                                        <tr>
                                            <td align="center">
                                                <asp:Button id="bt_cancel" runat="server"
Text="Cancel"></asp:Button></td>
                                            <td align="center">
                                                <asp:Button id="bt_save" runat="server"
Text="Save"></asp:Button></td>
                                        </tr>
                                    </table>
                                </div>
                            </form>
                        </body>
                    </HTML>
```

Servicemaintian.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using WSRegister.ClassLb;
using Tamir.SharpSsh;

namespace WSRegister
{
   /// <summary>
   /// Summary description for Servicemaintian.
   /// </summary>
   public class Servicemaintian : System.Web.UI.Page
   {
         protected System.Web.UI.WebControls.TextBox txtwsdl;
         protected System.Web.UI.WebControls.Button bt_next;
         protected System.Web.UI.WebControls.DataGrid dgmethos;
         protected System.Web.UI.WebControls.DataGrid dgparams;
         protected System.Web.UI.WebControls.Button bt_cancel;
         protected System.Web.UI.WebControls.Button bt_save;
         protected System.Web.UI.WebControls.Label lblWSDL;

         private void Page_Load(object sender, System.EventArgs e)
         {
               // Put user code to initialize the page here
               if (!Page.IsPostBack)
               {
                     //binddgmethos();
               }
         }

         #region Web Form Designer generated code
         override protected void OnInit(EventArgs e)
         {
               //
               // CODEGEN: This call is required by the ASP.NET Web Form
Designer.
               //
               InitializeComponent();
               base.OnInit(e);
         }

         /// <summary>
         /// Required method for Designer support - do not modify
         /// the contents of this method with the code editor.
         /// </summary>
         private void InitializeComponent()
         {
```

```
            this.dgmethos.ItemCommand += new
System.Web.UI.WebControls.DataGridCommandEventHandler(this.dgmethos_Select
Command);
            this.bt_save.Click += new
System.EventHandler(this.bt_save_Click);
            this.bt_next.Click += new
System.EventHandler(this.bt_next_Click);
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion

        private void bt_next_Click(object sender, System.EventArgs e)
        {
            //bind wsdls
            //ddWsdls.DataBind();
            //ddWsdls.Items.Insert(0, new ListItem("None", "0"));
            binddgmethos();
        }

        private void dgmethos_SelectCommand(object sender,
System.Web.UI.WebControls.DataGridCommandEventArgs e)
        {
            //get paramters
            binddgparams(Convert.ToInt32(e.Item.Cells[0].Text));
        }

        private void binddgmethos()
        {
            //int wsid=0;
            ClassLb.RegisterWSDL reg = new ClassLb.RegisterWSDL();
            /*for(int i=0; i < ddWsdls.Items.Count; i++)
            {
                if (ddWsdls.Items[i].Text == txtwsdl.Text)
                {
                    wsid = Convert.ToInt32(ddWsdls.Items[i].Value);
                }
            }*/
            dgmethos.DataSource = reg.Get_MDList(txtwsdl.Text);
            dgmethos.DataBind();
        }

        private void binddgparams(int pthid)
        {
            ClassLb.RegisterWSDL reg = new ClassLb.RegisterWSDL();
            dgparams.DataSource = reg.Get_ParmaterList(pthid);
            dgparams.DataBind();
        }

        protected ClassLb.WsdlList GetWDSLs()
        {
            ClassLb.RegisterWSDL reg = new ClassLb.RegisterWSDL();

            return reg.Get_WSList(1);
        }

        private void bt_save_Click(object sender, System.EventArgs e)
```

```
        {
            int mrst = 0, prst = 0;
            Method method;
            Parameter parm;
            TextBox tx;
            ClassLb.RegisterWSDL reg = new ClassLb.RegisterWSDL();

            foreach(DataGridItem dgrow in dgmethos.Items)
            {
                method = new Method();
                method.methodID =
Convert.ToInt32(dgrow.Cells[0].Text);
                tx = (TextBox)dgrow.FindControl("txtdesc");
                method.description = tx.Text;
                mrst = reg.update_method(method);
            }

            if (mrst == 1)
            {
                foreach(DataGridItem dgrow in dgparams.Items)
                {
                    parm = new Parameter();
                    parm.parmID =
Convert.ToInt32(dgrow.Cells[0].Text);
                    tx = (TextBox)dgrow.FindControl("txtDVal");
                    parm.DefaultValue = tx.Text;
                    prst = reg.update_parameter(parm);
                }
            }
        }

    }
}
```

Webclient.aspx

```
<%@ Page language="c#" Codebehind="WebClient.aspx.cs" AutoEventWireup="false"
Inherits="WSRegister.WebForm2" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <title>Client</title>
        <meta name="GENERATOR" Content="Microsoft Visual Studio .NET 7.1">
        <meta name="CODE_LANGUAGE" Content="C#">
        <meta name="vs_defaultClientScript" content="JavaScript">
        <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
    </HEAD>
    <body>
        <form id="Form1" method="post" runat="server">
            <div align="center">
                <table width="80%">
                    <tr>
                        <td align="center"><STRONG><FONT
color="#0033cc" size="5">Client Update Notification</FONT></STRONG></td>
                    </tr>
                    <tr>
                        <td align="center">
                            <table width="100%">
                                <tr>
                                    <td style="WIDTH: 85px"
align="right"><FONT size="2"><STRONG>Flag Location:</STRONG></FONT></td>
                                    <td><asp:textbox
id="txtfilepath" runat="server" Width="513px"></asp:textbox></td>
                                </tr>
                                <tr>
                                    <td style="WIDTH: 85px"
align="right"><FONT size="2"><STRONG>Cookie Location:</STRONG></FONT></td>
                                    <td>
                                        <asp:textbox
id="txtCookie" runat="server" Width="513px"></asp:textbox></td>
                                </tr>
                                <tr>
                                    <td style="WIDTH: 85px"
align="right"></td>
                                    <td>
                                        <asp:CheckBox
id="chk_update" runat="server" Text="Updated"></asp:CheckBox></td>
                                </tr>
                                <tr>
```

```
                                                    <td style="WIDTH: 85px"
align="right"></td>
                                                    <td></td>
                                                </tr>
                                                <tr>
                                                    <td style="WIDTH:
85px"></td>
                                                    <td>
                                                        <asp:Button
id="But_submit" runat="server"
Text="Updated"></asp:Button>        &
nbsp;           </td>
                                                </tr>
                                            </table>
                                            <asp:Label id="lblFeedback" runat="server"
Font-Size="X-Small"></asp:Label></td>
                                    </tr>
                                    <tr>
                                        <td></td>
                                    </tr>
                                </table>
                            </div>
                        </form>
    </body>
</HTML>
```

Webclient.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace WSRegister
{
    /// <summary>
    /// Summary description for WebForm2.
    /// </summary>
    public class WebForm2 : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.Button But_submit;
        protected System.Web.UI.WebControls.CheckBox chk_update;
        protected System.Web.UI.WebControls.TextBox txtfilepath;
        protected System.Web.UI.WebControls.TextBox txtCookie;
        protected System.Web.UI.WebControls.Label lblFeedback;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                txtfilepath.Text = @"D:\wsproxy\NewMethod.flg";
                txtCookie.Text = @"D:\wsproxy\mywscook.sck";
            }
            ClassLb.RegisterClient myclt = new
ClassLb.RegisterClient();
            chk_update.Checked = myclt.check(txtfilepath.Text);
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET Web Form
Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
```

```
            this.But_submit.Click += new
System.EventHandler(this.But_submit_Click);
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion

        private void But_submit_Click(object sender, System.EventArgs e)
        {
            try
            {
                ClassLb.RegisterClient myclt = new
ClassLb.RegisterClient();
                myclt.update(txtfilepath.Text, txtCookie.Text);
                lblFeedback.Text = "Update is successful!!";
            }
            catch(Exception ex)
            {
                lblFeedback.Text = ex.Message;
            }

        }
    }
}
```

APPENDIX III

Dynamic Invoker Source Code

Stub.cs

```
using System;
using System.CodeDom;
using System.ComponentModel;
using System.CodeDom.Compiler;
using System.Diagnostics;
using System.Collections;
using System.Web.Services.Description;
using System.Web.Services.Discovery;
using System.Web.Services.Protocols;
using Microsoft.CSharp;
using System.IO;
using System.Reflection;
using System.Xml;
using System.Xml.Schema;
using System.Xml.Serialization;
using System.Text;
using System.Net;
using System.Data;
using System.Data.SqlClient;

namespace DYInvoker
{
    /// <summary>
    /// Summary description for Stub.Generate proxe based on WSDL
    /// </summary>
    public class Stub
    {
        #region private member variables
        /*****************************************************************/
        private int _sID;
        //private short _level;
        private string _from;
        private string _wsdlUrl;
        private DateTime _Cdate;
        private DateTime _UPDdate;
        private StreamReader _sdr;
        //private string _partialfilename;
        private string _filepath;
        private string _pdata;
        private Assembly _ClientProxy = null;
        private object proxyInstance = null;
        //private string wsdl;
        private string typeName;
```

```
        private string methodName;
        private string protocolName = "Soap";
        private string proxySource;
        private ServiceDescriptionImporter sdi;

        /*************************************************************/
        #endregion private member variables

        #region constructors
        /*************************************************************/
        public Stub()
        {
              //
              // TODO: Add constructor logic here
              //
        }
        public Stub(int ID, DateTime cdate)
        {
              _sID = ID;
              _Cdate = cdate;
        }
        public Stub(int ID, DateTime cdate, Wsdl _wsdl)
        {
              _sID = ID;
              _Cdate = cdate;
              _wsdlUrl = _wsdl.Location;
              _filepath = GetCachePath(_wsdlUrl);
              //_filepath = "mpt.dll";
              if (_wsdl.Updated)
              {
                    bool bcl = ClearCache(_filepath);
              }
              proxyInstance = GetProxyFromWsdl(_wsdlUrl);
        }
        public Stub(int ID, DateTime cdate, string wsdlLocation)
        {
              _sID = ID;
              _Cdate = cdate;
              _wsdlUrl = wsdlLocation;
              _filepath = GetCachePath(_wsdlUrl);
              //_filepath = "mpt.dll";
              proxyInstance = GetProxyFromWsdl(_wsdlUrl);
        }

        public Stub(int ID, DateTime cdate, string wsdlLocation, string
inTypeName)
        {
              _sID = ID;
              _Cdate = cdate;
              _wsdlUrl = wsdlLocation;
              typeName = inTypeName;
              _filepath = GetCachePath(_wsdlUrl);
              proxyInstance = GetProxyFromWsdl(_wsdlUrl);
        }

        public Stub(int ID, DateTime cdate, string wsdlLocation, string
inTypeName, string inMethodName)
```

```
        {
              _sID = ID;
              _Cdate = cdate;
              _wsdlUrl = wsdlLocation;
              typeName = inTypeName;
              methodName = inMethodName;
              _filepath = GetCachePath(_wsdlUrl);
              proxyInstance = GetProxyFromWsdl(_wsdlUrl);
        }
        /*************************************************************/
        #endregion constructors

        #region properties
        /*************************************************************/
        public int SID{get{return _sID;} set{_sID = value;}}
        public string TypeName{get{return typeName;} set{typeName =
value;}}
        public Assembly ClientProxy{get{return _ClientProxy;}
set{_ClientProxy = value;}}
        public object ProxyInstance{get{return proxyInstance;}
set{proxyInstance = value;}}
        public String Filepath{get{return _filepath;}set{_filepath =
value;}}
        public string From{get{return _from;} set{_from = value;}}
        public string Result{get{return _pdata;} set{_pdata = value;}}
        public DateTime CDate{get{return _Cdate;} set{_Cdate = value;}}
        public DateTime UPDDate{get{return _UPDdate;} set{_UPDdate =
value;}}

        /*************************************************************/
        #endregion properties

        #region public methods
        /*************************************************************/
        public string Create_Proxy()
        {
              return Create_Proxy(_wsdlUrl);
        }
        public string Create_Proxy(string svrUri)
        {
              string _filename;
              //multiple users can invoke this method at same time
              Random rd = new Random();
              _filename = "stub" + rd.Next(10000).ToString() + ".cs";
              Process.Start("wsdl.exe", "/out:" + _filename + " " +
svrUri);
              _sdr = new StreamReader(_filename);
              return _filename;

        }
        public MethodList Build_MethodsOrderly()
        {
              MethodList methods = new MethodList();
              //sort the methods
              if (sdi.ServiceDescriptions.Count > 0)
              {
```

```
                    foreach(ServiceDescription sd in
sdi.ServiceDescriptions)
                    {
                          if(sd != null)
                          {
                                //string param =
mdNode.Attributes["name"].ToString();
                                foreach(Operation op in
sd.Bindings[0].Operations)
                                {
                                      Method md = new Method();
                                      md.name = op.Name;
                                      md.active = true;
                                      methods.Add(md);
                                }
                          }
                    }
              }

              return methods;
        }

        public string build_Ondemanproxy(MethodList methods)
        {
              return "N/A";

        }
        public object GetCustProxyFromWSDL(string wsdluri, MethodList
methods)
        {
              object proxy = null;

              InjectExtension(typeof(SoapMessageAccessClientExtension));

              //check if WS alavilable, if now, clear cache and get its
back up
              wsdluri = CheckWS(wsdluri);

              if(CheckCache(_filepath))  //Check WSDL changed or not
              {
                    _ClientProxy = Assembly.LoadFrom(_filepath);
                    // create proxy instance
                    proxy = CreateInstance(typeName);
              }
              else
              {
                    //build wsdl XML doc
                    string strWsdl = GetWsdl(_wsdlUrl); //get wsdl doc
                    foreach(Method method in methods)
                    {
                          strWsdl = strWsdl.Replace(method.name, "");
                    }
                    //finally return partial wsdl string
                    proxy = BuildProxyFromWsdl(strWsdl);
              }

              return proxyInstance;
```

```
        }
        public object GetProxyFromWsdl(string wsdluri)
        {
              object proxy = null;


   //InjectExtension(typeof(SoapMessageAccessClientExtension));

              //check if WS alavilable, if now, clear cache and get its
back up
              wsdluri = CheckWS(wsdluri);

              if (File.Exists(@"D:\wsproxy\NewMethod.flg"))
              {
                    ClearCache(_filepath);
              }

              if(CheckCache(_filepath))  //Check WSDL changed or not
              {
                    _ClientProxy = Assembly.LoadFrom(_filepath);
                    // create proxy instance
                    proxy = CreateInstance(typeName);

              }
              else
              {
                    string strWsdl = GetWsdl(wsdluri); //get wsdl doc
                    proxy = BuildProxyFromWsdl(strWsdl);

              }
              return proxy;
        }
        /**************************************************************/
        #endregion public methods

        #region private methods
        /**************************************************************/
        private object BuildProxyFromWsdl(string strWsdl)
        {
              try
              {
                    Uddi uddi = new Uddi();

                    // Use an XmlTextReader to get the Web Service
description
                    StringReader wsdlStringReader = new
StringReader(strWsdl);
                    XmlTextReader tr = new
XmlTextReader(wsdlStringReader);
                    ServiceDescription sd = ServiceDescription.Read(tr);
                    tr.Close();

                    // WSDL service description importer
                    CodeNamespace cns = new CodeNamespace("DYProxy");
                    sdi = new ServiceDescriptionImporter();
                    //sdi.AddServiceDescription(sd, null, null);
```

```
                    // check for optional imports in the root WSDL
                    sdi = uddi.CheckForImports(_wsdlUrl, sdi);

                    sdi.ProtocolName = protocolName;
                    sdi.Import(cns, null);

                    // change the base class
                    CodeTypeDeclaration ctDecl = cns.Types[0];
                    cns.Types.Remove(ctDecl);
                    ctDecl.BaseTypes[0] = new
CodeTypeReference("DYInvoker.DInv_SPExtension");
                    cns.Types.Add(ctDecl);

                    // source code generation
                    CSharpCodeProvider cscp = new CSharpCodeProvider();
                    ICodeGenerator icg = cscp.CreateGenerator();
                    StringBuilder srcStringBuilder = new StringBuilder();
                    StringWriter sw = new StringWriter(srcStringBuilder);
                    icg.GenerateCodeFromNamespace(cns, sw, null);
                    proxySource = srcStringBuilder.ToString();
                    sw.Close();

                    // assembly compilation
                    CompilerParameters cp = new CompilerParameters();
                    cp.ReferencedAssemblies.Add("System.dll");
                    cp.ReferencedAssemblies.Add("System.Xml.dll");

   cp.ReferencedAssemblies.Add("System.Web.Services.dll");
                    cp.ReferencedAssemblies.Add("System.Data.dll");

   cp.ReferencedAssemblies.Add(System.Reflection.Assembly.
                          GetExecutingAssembly().Location);

                    cp.GenerateExecutable = false;
                    cp.GenerateInMemory = false;
                    cp.IncludeDebugInformation = false;

                    ICodeCompiler icc = cscp.CreateCompiler();
                    CompilerResults cr =
icc.CompileAssemblyFromScurce(cp, proxySource);

                    if(cr.Errors.Count > 0)
                          throw new Exception(string.Format(@"Build
failed: {0} errors", cr.Errors.Count));

                    _ClientProxy = cr.CompiledAssembly;
                    //rename temporary assembly in order to cache it for
later use
                    cacheAssembly(cr.PathToAssembly);

                    // create proxy instance
                    return CreateInstance(typeName);
              }
              catch(Exception ex)
              {
                    _ClientProxy = null;
                    throw new Exception(ex.Message);
```

```
            }
        }
        private string create_proxyCode(int idx)
        {
            string hd="";
            DBAccess dba = new DBAccess();
            SqlDataReader rd = dba.exec_sql("sp_get_proxycode",
"@type_id", "int", idx);
            while(rd.Read())
            {
                hd += rd.GetString(2) + "\n";
            }
            return hd;

        }
        private string create_method(string mName, string returnType,
ArrayList types)
        {
            string md;

   md="[System.Web.Services.Protocols.SoapDocumentMethodAttribute(\"http:/
/tempuri.org/" + mName + "\", RequestNamespace=\"http://tempuri.org/\",
ResponseNamespace=\"http://tempuri.org/\",
Use=System.Web.Services.Description.SoapBindingUse.Literal,
ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]\
n";
            string pars="", vars="";
            int idx=0;

            //This will make a list of paramaters
            foreach(string type in types)
            {
                pars += type + " var"+idx.ToString() + ", ";
                vars+= "var"+idx.ToString() + ", ";
                idx++;
            }

            //Remove last two chars , and space
            pars = pars.Remove(pars.Length-2, 2);
            vars = vars.Remove(vars.Length-2, 2);

            md+="public " + returnType + " " + mName + "("+ pars
+"){\n";
            md+=" object[] results = this.Invoke(\"" + mName + "\", new
object[] {" + vars +"});\n";
            md+=" return (("+ returnType + ")(results[0]));\n";
            md+="}\n";

            md+="public System.IAsyncResult BeginOrder(" + pars +",
System.AsyncCallback callback, object asyncState) {";
            md+="   return this.BeginInvoke(\""+ mName + "\", new
object[] {ord}, callback, asyncState);}\n";

            md+="public" + returnType + "End"+ mName +
"(System.IAsyncResult asyncResult) {\n";
            md+="   object[] results = this.EndInvoke(asyncResult);\n";
            md+="   return ((int)(results[0]));}\n";
```

```
                md+="\n";

                return md;

        }
        private string create_type(string tpName, ArrayList types)
        {
                string md;

   md="[System.Xml.Serialization.XmlTypeAttribute(Namespace=\"http://tempu
ri.org/\")]\n";
                int idx=0;
                md += "public class " + tpName + "{\n";
                //This will make a list of paramaters
                foreach(string type in types)
                {
                        md += "     public " + type + " var"+idx.ToString() +
";\n";
                        idx++;
                }
                md+="}\n";
                return md;
        }

        private string GetWsdl(string WsdlUri)
        {
                Wsdl mwsdl = new Wsdl();
                return mwsdl.GetWsdlFromUri(WsdlUri);
        }

        private string CheckWS(string WSDLUri)
        {
                string LUri = WSDLUri.Substring(0, WSDLUri.IndexOf("?",
0));
                try
                {
                        WebRequest wrq = WebRequest.Create(LUri);
                        wrq.Timeout = 10000;
                        WebResponse wrb = wrq.GetResponse();
                }
                catch(Exception ex)
                {
                        if(ex.Message.Length >0)
                        {
                                //if not clear cache
                                if (ClearCache(_filepath))
                                {
                                        //get backup WSDLUri
                                        DBAccess dba = new DBAccess();
                                        SqlDataReader rd = dba.exec_sql("exec
sp_get_backupWSbywsdl @wsUrl= '" + WSDLUri + "'");
                                        while(rd.HasRows)
                                        {
                                                rd.Read();
                                                WSDLUri = rd.GetString(3);
                                        }
                                }
```

```
                }
            }
            return WSDLUri;
        }
        private bool CheckCache(string _filepath) //if changed cache will
be deleted by batch process
        {
            if(File.Exists(_filepath))
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        private bool ClearCache(string _filepath)
        {
            // clear the cached assembly file for this WSDL
            try
            {
                File.Delete(_filepath);
                return true;
            }
            catch(Exception e)
            {
                e = e;
                return false;
            }
        }
        private string GetCachePath(string wsdlURL)
        {
            string lfilepath="";
            DBAccess dba = new DBAccess();
            SqlDataReader rd = dba.exec_sql("sp_get_proxybywsdl",
"@wsdlurl", "string", wsdlURL);
            while(rd.Read())
            {
                lfilepath = rd.GetString(6);
            }
            return lfilepath;
        }

        private object CreateInstance(string objTypeName)
        {
            // check whether the type is already created or not
            if (objTypeName == "" || objTypeName == null)
            {
                foreach (Type ty in ClientProxy.GetTypes())
                {
                    if(ty.BaseType ==
typeof(DYInvoker.DInv_SPExtension))
                    {
                        objTypeName = ty.Name;
                        break;
                    }
                }
```

```
            }
            Type t = _ClientProxy.GetType("DYProxy." + objTypeName);
            return Activator.CreateInstance(t);
      }
      static void InjectExtension(Type extension)
      {
            Assembly assBase;
            Type webServiceConfig;
            object currentProp;
            PropertyInfo propInfo;
            object[] value;
            Type myType;
            object[] objArray;
            object myObj;
            FieldInfo myField;

            try
            {
                  assBase = typeof(SoapExtensionAttribute).Assembly;
                  webServiceConfig =

   assBase.GetType("System.Web.Services.Configuration.WebServicesConfigura
tion");

                  if (webServiceConfig == null)
                        throw new Exception("Error ...");

                  currentProp = webServiceConfig.GetProperty("Current",
                        BindingFlags.NonPublic | BindingFlags.Static |
                        BindingFlags.Instance | BindingFlags.Public
                        ).GetValue(null, null);
                  propInfo =
webServiceConfig.GetProperty("SoapExtensionTypes",
                        BindingFlags.NonPublic | BindingFlags.Static |
                        BindingFlags.Instance | BindingFlags.Public
                        );
                  value = (object[])propInfo.GetValue(currentProp,
null);
                  myType = value.GetType().GetElementType();
                  objArray = (object[])Array.CreateInstance(myType,
(int)value.Length + 1);
                  Array.Copy(value, objArray, (int)value.Length);

                  myObj = Activator.CreateInstance(myType);
                  myField = myType.GetField("Type",
                        BindingFlags.NonPublic | BindingFlags.Static |
                        BindingFlags.Instance | BindingFlags.Public
                        );
                  myField.SetValue(myObj, extension,
                        BindingFlags.NonPublic | BindingFlags.Static |
                        BindingFlags.Instance | BindingFlags.Public |
BindingFlags.SetField,
                        null, null
                        );
                  objArray[(int)objArray.Length - 1] = myObj;
                  propInfo.SetValue(currentProp, objArray,
                        BindingFlags.NonPublic | BindingFlags.Static |
```

```
                              BindingFlags.Instance | BindingFlags.Public |
BindingFlags.SetProperty,
                              null, null, null
                              );
                }
                catch (Exception e)
                {
                      throw new Exception("Stub-InjectExtension:", e);
                }
          }
          private void cacheAssembly(string pathToAssembly)
          {
                File.Copy(pathToAssembly, _filepath);
          }
          private Wsdl getWsdl(string location, string version, string
name)
          {
                Wsdl _wsdl = new Wsdl();
                if(location.IndexOf("wsdl", 0) > 0)
                {
                      _wsdl.Location = location;
                      _wsdl.name = name;
                      _wsdl.Version = version;
                }
                else
                {
                      if (location != "")
                      {
                            //location is ws location, and find wsdl
location
                      }
                      else
                      {
                            _wsdl = null;
                            throw new Exception("Invalid WSDL ID!!");
                      }
                }
                return _wsdl;
          }
          private Wsdl getWsdl(int wld)
          {
                Wsdl _wsdl;
                if(wld > 0)
                {
                      _wsdl = new Wsdl(wld);
                }
                else
                {
                      _wsdl = null;
                      throw new Exception("Invalid WSDL ID!!");
                }
                return _wsdl;
          }
          private string get_Remote_wsdlVersion(string url)
          {
                //get new wsdl version
                //Check Availablity No use
```

```
            Uri siteUri = new Uri(url);

            try
            {
                  WebRequest wr = WebRequest.Create(siteUri);
                  WebResponse ws = wr.GetResponse();
                  if (ws.ResponseUri.AbsoluteUri == url)
                     return "1.1.0";
                  else
                        return "1.1.1";
            }
            catch(Exception ex)
            {
                  return ex.Message;
            }

      }
      /*****************************************************************/
      #endregion private methods
   }
}
```

Invocation.cs

```
using System.IO;
using System.Reflection;
using System.Xml;
using System.Xml.Schema;
using System.Xml.Serialization;
using System.Text;
using System.Net;
using System.Net.Sockets;
using Microsoft.Web.Services2;
using System.Data;
using System.Data.SqlClient;

namespace DYInvoker
{
   /// <summary>
   /// Summary description for Class1.
   /// This has all invoking relative methods.
   /// </summary>

   public class Invocation
   {
         enum Language {Java=1, CSP=2, VB=3};
         enum pxhead
{assembly=1,HAttribute=2,MAttribute=3,TypeAttribute=4};

         #region private member variables

         //private Uddi _uddi;
         //private string _location="";
         //private Assembly ass = null;
         private object proxyInstance = null;
         //private string wsdl;
         private string wsdlUrl;
         //private string typeName;
         private string methodName;
         //private string protocolName = "Soap";
         private ArrayList methodParams;

         #endregion private member variables

         #region Public propety
         public string SoapRequest
         {
               get
               {
                     PropertyInfo propInfo =
proxyInstance.GetType().GetProperty("SoapRequest");
                     object result = propInfo.GetValue(proxyInstance,
null);
                     System.Text.UTF8Encoding enc = new
System.Text.UTF8Encoding();

                     return enc.GetString((byte[])result);
               }
```

```
        }

        public string SoapResponse
        {
              get
              {
                    PropertyInfo propInfo =
proxyInstance.GetType().GetProperty("SoapResponse");
                    object result = propInfo.GetValue(proxyInstance,
null);
                    System.Text.UTF8Encoding enc = new
System.Text.UTF8Encoding();

                    return enc.GetString((byte[])result);
              }
        }
        public string MethodName
        {
              get
              {
                    return methodName;
              }
              set
              {
                    methodName = value;
              }
        }
        public string WsdlUrl
        {
              get
              {
                    return wsdlUrl;
              }
              set
              {
                    wsdlUrl = value;
              }
        }
        #endregion Public property

        #region constructors

        public Invocation()
        {
              methodParams = new ArrayList();
        }

        #endregion constructors

        #region public methods

        public Stub Creat_proxyfromWSDL(string clientName, string wsUrl)
        {
              Wsdl _wsdl = new Wsdl(wsUrl);
              wsdlUrl = wsUrl;
              Random rd = new Random(10000);
```

```
                Stub myStub =  new Stub(rd.Next(0, 20000), DateTime.Now,
_wsdl);
                myStub.From = clientName;

                if(_wsdl.BindingType.ToUpper() == "HTTP")
                {
                      //create a stub object
                      if(myStub.ClientProxy == null)
                      {
                            myStub.Result = "bad";
                      }
                      else
                      {
                            myStub.Result = "good";
                      }

                }

                return myStub;
          }

          public object InvokeCall(Stub stubInstance, string methodName)
          {
                try
                {
                      MethodInfo minfo =
stubInstance.ProxyInstance.GetType().GetMethod(methodName);
                      //check the method name if new method swap it and add
it to methodParams

                      if(File.Exists(@"D:\wsproxy\NewMethod.flg"))
                      {
                            CheckInParamters(methodName);
                            Debug.WriteLine("count:" +
methodParams.Count.ToString());
                      }

                      return minfo.Invoke(stubInstance.ProxyInstance,
methodParams.ToArray());
                }
                catch(Exception e)
                {
                      return e;
                }
          }

          public void AddParameter(object param)
          {
                methodParams.Add(param);
          }

          #endregion public methods

          #region private methods
          private bool CheckInParamters(string methodName)
          {
                bool Inp = false;
```

```
                string filename = @"D:\wsproxy\mywscook_" + methodName +
".sck";
                string rem = "";
                ParameterList plt = new ParameterList();

                if (File.Exists(filename))
                {
                      Inp = true;
                }
                else //build profiler
                {
                      Inp  = CreateSCookie(methodName, filename);
                }
                if (Inp)
                {
                      //load from sck xml doc
                      XmlDocument xmldoc = new XmlDocument();
                      xmldoc.Load(filename);
                      //build Parms List  Method/GetOrders/params/param
                      XmlNodeList _nodes = xmldoc.SelectNodes("Method/" +
methodName + "/params/param");
                      foreach(XmlNode _node in _nodes)
                      {
                            Parameter par = new Parameter();
                            par.name = _node.Attributes["name"].Value;
                            par.Active = _node.Attributes["active"].Value;
                            par.DefaultValue =
_node.Attributes["dvalue"].Value;
                            par.valueType =
_node.Attributes["vtype"].Value;
                            plt.Add(par);
                      }
                }
                for(int idx=0; idx < plt.Count; idx++)
                {
                      if (plt[idx].Active == "I")
                      {
                            rem = rem + idx.ToString() + ",";
                      }
                      else if (plt[idx].Active == "N")
                      {
                            if (plt[idx].valueType == "int")
                            {
                                  if (plt[idx].DefaultValue.Length > 0)
                                  {

   methodParams.Add(Convert.ToInt32(plt[idx].DefaultValue));
                                  }
                                  else
                                  {

   methodParams.Add(Convert.ToInt32("0"));
                                  }
                            }
                            else if(plt[idx].valueType == "string")
                            {
```

```
   methodParams.Add(Convert.ToString(plt[idx].DefaultValue));
                              }
                              else if(plt[idx].valueType == "datetime")
                              {
                                    if (plt[idx].DefaultValue.Length > 0)
                                    {

   methodParams.Add(Convert.ToDateTime(plt[idx].DefaultValue));
                                    }
                                    else
                                    {

   methodParams.Add(Convert.ToDateTime("1900-01-01"));
                                    }
                              }
                        }
                  }     //for
                  //remove inactive param
                  //Note:if service has changed more than twice, but client
hasn't changed once, it will be a issue
                  if (rem.Length >= 1)
                  {
                        rem = rem.Substring(0, rem.Length-1);
                        char [] sp = {','};
                        string [] ids = rem.Split(sp);
                        foreach(string idx in ids)
                        {
                              methodParams.RemoveAt(Convert.ToInt32(idx));
                        }
                  }
                  return true;
            }
            private bool CreateSCookie(string methodName, string filename)
            {
                  bool Inp = false;
                  StringBuilder sb = new StringBuilder();
                  string vtype = "";
                  sb.Append("<Method>\n");
                  sb.Append("  <" + methodName + ">\n");
                  DBAccess dba = new DBAccess();
                  SqlDataReader rd = dba.exec_sql("exec sp_GetWSMethod
@wsUrl= '" + wsdlUrl + "', @name='" + methodName + "'");
                  if (rd.HasRows)
                  {
                        //methodParams.Clear();
                        DBAccess dbaloc = new DBAccess();
                        rd.Read();
                        int Pdtid = rd.GetInt32(4);
                        SqlDataReader rdp = dbaloc.exec_sql("exec
sp_GetInParametersByPdtid @Pdtid=" + Pdtid.ToString());
                        sb.Append("    <params>\n");
                        while(rdp.Read())
                        {
                              if (!rdp.IsDBNull(4))
                              {
                                    vtype = rdp.GetString(4);
```

```
                        }
                        //build xml string
                        sb.Append("      <param name=\"" +
rdp.GetString(3) + "\"" + " active=\"" + rdp.GetString(5) + "\"" + "
vtype=\"" + rdp.GetString(9) + "\"" + " dvalue=\"");
                        if (rdp.GetString(5) == "N")
                        {
                              sb.Append(vtype);
                        }
                        sb.Append("\"></param>\n");
                  }
                  sb.Append("    </params>\n");
                  Inp  = true;
            } //if
            sb.Append("  </" + methodName + ">\n");
            sb.Append("</Method>\n");
            if (Inp)
            {
                  Inp = creatfile(filename, sb.ToString());
            }
            return Inp;
      }
      private bool creatfile(string path, string strVal)
      {
            try
            {
                  FileStream sb = new FileStream(path,
FileMode.OpenOrCreate);
                  StreamWriter sw = new StreamWriter(sb);
                  sw.Write(strVal);
                  sw.Close();
                  return true;
            }
            catch(Exception ex)
            {
                  throw new Exception(ex.Message);
            }
      }
      private void ResetInternalState()
      {
            //string typeName, protocolName;
            //typeName = "";
            methodName = "";
            //protocolName = "Soap";
            methodParams.Clear();
            proxyInstance = null;
      }

      //add description attribute, so that registry will pull this info
into DB
      #endregion private methods
   }
}
```

SCookie.cs

```
using System;
using System.IO;
using System.Collections;
using System.Text;

namespace DYInvoker
{
    /// <summary>
    /// Summary description for SCookie.
    /// </summary>
    public class SCookie
    {
        #region private member variables
        /*************************************************************/
        private int _sID;
        private string _client;
        private string _serverIP;
        private string _version;
        private DateTime _Cdate;
        private string _pdata;

        /*************************************************************/
        #endregion private member variables

        #region constructors
        /*************************************************************/
        public SCookie()
        {
            _sID = 0;

        }
        public SCookie(int ID, DateTime cdate)
        {
            _sID = ID;
            _Cdate = cdate;
        }
        /*************************************************************/
        #endregion constructors

        #region properties
        /*************************************************************/
        public int CID{get{return _sID;} set{_sID = value;}}
        public string Client{get{return _client;} set{_client = value;}}
        public string ServerIP{get{return _serverIP;} set{_serverIP =
value;}}
        public string Version{get{return _version;} set{_version =
value;}}
        public string Data{get{return _pdata;} set{_pdata = value;}}
        public DateTime CDate{get{return _Cdate;} set{_Cdate = value;}}

        /*************************************************************/
        #endregion properties

        #region public methods
```

```
        /*************************************************************/
        public void create_scookie(string fileloc)
        {
              //store scooike info into db
              StreamWriter sd = new StreamWriter(fileloc);
              sd.Write(_sID.ToString() + "~" + _client + "~" + _serverIP
+ "~" + _version + "~" + _pdata + "~" +_Cdate.ToString());
        }
        public void set_scookie(string strcook)
        {
              string[] aryCookParm = null;
              aryCookParm = strcook.Split('~');
              _sID = Convert.ToInt32(aryCookParm[0]);
              _client = aryCookParm[1];
              _serverIP = aryCookParm[2];
              _version = aryCookParm[3];
              _pdata = aryCookParm[4];
              _Cdate = Convert.ToDateTime(aryCookParm[5]);

        }
        /*************************************************************/
        #endregion public methods

        #region private methods
        /*************************************************************/

        /*************************************************************/
        #endregion private methods
    }
}
```

DBAccess.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Diagnostics;

namespace DYInvoker
{
   /// <summary>
   /// Summary description for DBAccess.
   /// </summary>
   public class DBAccess
   {
         SqlConnection sql_conn;
         private const string StrConn = "Data Source=Esszz;
uid=WS_Invoker; pwd=ws888888; Initial Catalog=WSReg";

         public DBAccess()
         {
               // Add code
               sql_conn = new SqlConnection(StrConn);
               try
               {
                     sql_conn.Open();
               }
               catch(Exception ex)
               {
                     writetolog(ex.Message);
                     sql_conn = null;
               }

         }
         ~DBAccess()
         {
               //
         }

         internal SqlDataReader exec_sql(string str_sql)
         {
               SqlDataReader rdr;
               //SqlConnection sql_conn = get_conn();
               try
               {
                     SqlCommand SCmd = new SqlCommand(str_sql, sql_conn);
                     rdr = SCmd.ExecuteReader();
                     return rdr;
               }
               catch(Exception ex)
               {
                     writetolog(ex.Message);
                     rdr = null;
                     throw ex;
               }

         }
```

```
        internal SqlDataReader exec_sql(string str_sql, string parmname,
string type, object v)
        {
              try
              {
                    SqlCommand cmd = new SqlCommand(str_sql);
                    SqlDataReader rdr = null;
                    cmd.CommandType = CommandType.StoredProcedure;
                    if (type == "int")
                    {
                          cmd.Parameters.Add(parmname, SqlDbType.Int);
                    }
                    else if(type == "string")
                    {
                          cmd.Parameters.Add(parmname,
SqlDbType.VarChar);
                    }
                    cmd.Parameters[parmname].Value = v;
                    cmd.Connection = sql_conn;
                    //cmd.Connection.Open();

                    rdr = cmd.ExecuteReader();
                    return rdr;
              }
              catch(Exception ex)
              {
                    writetolog(ex.Message);
                    throw ex;
              }
        }

        internal short exec_cmd(string sqlstr)
        {
              SqlCommand cmd;

              try
              {
                    cmd = new SqlCommand(sqlstr, sql_conn);
                    return run_sqlcommand(cmd);
              }
              catch(Exception ex)
              {
                    writetolog(ex.Message);
                    cmd = null;
                    return -1;
              }
        }

        private short run_sqlcommand(SqlCommand cmd)
        {
              short res = 0;
              try
              {
                    cmd.ExecuteNonQuery();
                    res = 1;
              }
```

```
                catch(Exception ex)
                {
                      res = 2;
                      writetolog(ex.Message);
                }
                cmd.Connection.Close();
                return res;
          }

          private void writetolog(string logmessage)
          {
                Debug.Listeners.Add(new
TextWriterTraceListener(Console.Out));
                Debug.AutoFlush = true;
                Debug.Indent();
                Debug.WriteLine("Entering Main");
                Console.WriteLine("Hello World.");
                Debug.WriteLine("Exiting Main");
                Debug.Unindent();
          }

    }
}
```

RSAccess.cs

```
using System;
using System.Diagnostics;
using System.Threading;

namespace DYInvoker
{
   /// <summary>
   /// Summary description for RSAccess.
   /// </summary>
   internal class RSAccess : Exception
   {
         public RSAccess()
         {
               //
               // TODO: Add constructor logic here
               //
         }
         #region private memeber variables
         private string str_Message = "";
         private string srcName = "";
         private EventLog appLog = new EventLog("WebApps");
         private string _connString = "";
         private Exception pMExcept = null;
         private int pvType =1;
         #endregion private memeber variables

         #region constructors & destractor
         public RSAccess(string pStr) : base(pStr)
         {
               str_Message = pStr;
         }
         public RSAccess(string pStr, string SourceName) : base(pStr)
         {
               str_Message = pStr;
               srcName = SourceName;

         }
         public RSAccess(string pStr, Exception pExcept, string
SourceName) : base(pStr, pExcept)
         {
               str_Message = pStr;
               pMExcept = pExcept;
               srcName = SourceName;
         }
         ~RSAccess()
         {
               //MEventSource.Dispose();
         }
         #endregion constructors & destractor

         #region public member variables
         public int publish_type
         {
               get
```

```
        {
                return pvType;
        }
        set
        {
                pvType = value;
        }
}
public string conStr
{
        get
        {
                return _connString;
        }
        set
        {
                _connString = value;
        }
}
#endregion public member variables


#region public methods
public void GetPublish()
{
        if (pvType ==1)
        {
                pvPublisher_Log();
        }

}
public void GetPublish_Log(int reval)
{
        switch (reval)
        {
                case -1 :
                        str_Message ="Failed Service Insert";
                        break;
                case -2 :
                        str_Message = "Attribute Type Insert Failed";
                        break;
                case -3 :
                        str_Message = "Failed Attribute Insert";
                        break;
        }
        pvPublisher_Log();
}
#endregion public methods

#region private meothods

private void pvPublisher_Log()
{
        if (!EventLog.SourceExists(srcName))
        {
                EventLog.CreateEventSource(srcName, "WebApps");
        }
```

```
            appLog.Source = srcName;
            System.Console.Out.WriteLine("Error: " + str_Message);

        }
        #endregion private meothods

    }
}
```

DInv_SPExtension.cs

```
using System;
using System.IO;
using System.Xml;
using System.Text;
using System.Web.Services;
using System.Web.Services.Protocols;

namespace DYInvoker
{
   /// <summary>
   /// Summary description for DInv_SPExtension.
   /// </summary>
   [System.Web.Services.WebServiceBindingAttribute(Name="DInv_SPExtensionS
oap", Namespace="DYInvoker")]
   public class DInv_SPExtension : SoapHttpClientProtocol
   {
         public DInv_SPExtension()
         {
               //
               // TODO: Add constructor logic here
               //
         }
         public bool check_contract(string contract)
         {
               return true;
         }
         public bool check_auth()
         {
               return true;
         }
         private byte[] m_SoapRequestMsg = null;
         private byte[] m_SoapResponseMsg = null;

         public byte[] SoapRequest
         {
               get
               {
                     return m_SoapRequestMsg;
               }
               set
               {
                     m_SoapRequestMsg = value;
               }
         }

         public byte[] SoapResponse
         {
               get
               {
                     return m_SoapResponseMsg;
               }
               set
               {
                     m_SoapResponseMsg = value;
```

```
            }
        }

    }

    public class SoapMessageAccessClientExtension : SoapExtension
    {
        Stream oldStream = null;
        Stream newStream = null;

        public override object GetInitializer(LogicalMethodInfo
methodInfo, SoapExtensionAttribute attribute)
        {
            return null;
        }

        public override object GetInitializer(Type t)
        {
            return typeof(SoapMessageAccessClientExtension);
        }

        public override void Initialize(object initializer)
        {
            //
        }

        public override void ProcessMessage(SoapMessage message)
        {
            switch (message.Stage)
            {
                case SoapMessageStage.BeforeSerialize:
                    break;

                case SoapMessageStage.AfterSerialize:
                    StoreRequestMessage(message);
                    // Pass it off as the actual stream
                    Copy(newStream, oldStream);
                    // Indicate for the return that we don't wish
to chain anything in
                    break;

                case SoapMessageStage.BeforeDeserialize:
                    StoreResponseMessage(message);
                    // Pass it off as the actual stream
                    break;

                case SoapMessageStage.AfterDeserialize:
                    break;

                default:
                    throw new ArgumentException("Invalid Soap
Message stage [" + message.Stage + "]", "message");
            }
        }

        public override Stream ChainStream(Stream stream)
        {
```

```
                // Store old
                oldStream = stream;
                newStream = new MemoryStream();

                // Return new stream
                return newStream;
          }

          private void StoreRequestMessage(SoapMessage message)
          {
                // Rewind the source stream
                newStream.Position = 0;

                // Store message in our slot in the SoapHttpClientProtocol-
derived class
                byte[] bufEncSoap = new Byte[newStream.Length];
                newStream.Read(bufEncSoap, 0, bufEncSoap.Length);

    ((DInv_SPExtension)(((SoapClientMessage)message).Client)).SoapRequest =
bufEncSoap;
          }

          private void StoreResponseMessage(SoapMessage message)
          {
                Stream tempStream = new MemoryStream();
                Copy(oldStream, tempStream);

                // Store message in our slot in the SoapHttpClientProtocol-
derived class
                byte[] bufEncSoap = new Byte[tempStream.Length];
                tempStream.Read(bufEncSoap, 0, bufEncSoap.Length);

    ((DInv_SPExtension)(((SoapClientMessage)message).Client)).SoapResponse
= bufEncSoap;

                Copy(tempStream, newStream);
          }

          void Copy(Stream from, Stream to)
          {
                if (from.CanSeek == true)
                      from.Position = 0;
                TextReader reader = new StreamReader(from);
                TextWriter writer = new StreamWriter(to);
                writer.WriteLine(reader.ReadToEnd());
                writer.Flush();
                if (to.CanSeek == true)
                      to.Position = 0;
          }
    }

    [AttributeUsage(AttributeTargets.Method)]
    public class SoapMessageAccessClientExtensionAttribute :
SoapExtensionAttribute
    {
          private int priority;
```

```
        public override Type ExtensionType
        {
              get { return typeof(SoapMessageAccessClientExtension); }
        }

        public override int Priority
        {
              get
              {
                    return priority;
              }
              set
              {
                    priority = value;
              }
        }

    }
}
```

Uddi.cs

```
using System;
using System.Xml.Schema;
using System.Collections;
using System.Data.SqlClient;
using System.Web.Services.Description;
using System.Web.Services.Discovery;
using System.Web.Services.Protocols;

namespace DYInvoker
{
   /// <summary>
   /// Summary description for Uddi.
   /// This is for identifiy(find or verify) the web service(new or
exsting)
   /// </summary>
   ///
   public class Uddi
   {
         #region private member variables
         /*************************************************************/
         private int _providerID;
         private String _pname;
         private String _sname;
         private String _version;
         private String _description;
         private String _location;
         private String _SKey;
         private int _value;
         private DateTime _updateTime;
         private DBAccess dba;

         private struct uddi_item
         {
               private int _PID;
               private string _PName;
               public int PID{get{return _PID;} set{_PID = value;}}
               public string PName{get{return _PName;} set{_PName =
value;}}
         }

         /*************************************************************/
         #endregion private member variables

         #region constructors
         /*************************************************************/
         public Uddi()
         {
               dba = new DBAccess();
         }
         public Uddi(string skey)
         {
               _SKey = skey;
         }
         public Uddi(string skey, int sval)
```

```
        {
              _value = sval;
        }
        #endregion constructors

        #region properties
        /*************************************************************/
        public int ProviderID{get{return _providerID;} set{_providerID =
value;}}
        public String Provider{get{return _pname;} set{_pname = value;}}
        public String Service{get{return _sname;} set{_sname = value;}}
        public string Version{get{return _version;} set{_version =
value;}}
        public String description{get{return
_description;}set{_description = value;}}
        public String Location{get{return _location;}set{_location =
value;}}
        public String ServiceKey{get{return _SKey;} set{_SKey = value;}}
        public int SValue {get{return _value;} set{_value = value;}}
        public DateTime UpdateTime
        {
              get{ return _updateTime;}
              set{_updateTime = value;}
        }

        /*************************************************************/
        #endregion properties

        #region public methods
        /*************************************************************/
        public ServiceDescriptionImporter CheckForImports(string WSDLUrl,
ServiceDescriptionImporter sdi)
        {
              DiscoveryClientProtocol dcp = new
DiscoveryClientProtocol();
              dcp.DiscoverAny(WSDLUrl);
              dcp.ResolveAll();

              foreach (object osd in dcp.Documents.Values)
              {
                    if (osd is ServiceDescription)
sdi.AddServiceDescription((ServiceDescription)osd, null, null);;
                    if (osd is XmlSchema)
sdi.Schemas.Add((XmlSchema)osd);
              }
              return sdi;
        }
        public short Save_Uddi()
        {
              try
              {
                    return dba.exec_cmd("exec sp_insWSProfiler " +
_providerID + "','" + Service + "','" + Location + "','" + description +
"','" + _version + "'");
              }
              catch(Exception ex)
              {
```

```
                    throw new Exception("Can't Save Uddi" + _SKey +
ex.Message);
              }
        }
        //find replacement uddi from exsiting DB
        public ArrayList search_uddi_DB(string Service)
        {
              ArrayList marrlst = new ArrayList();
              uddi_item _uddi = new uddi_item();

              if(Service != "")
              {
                    SqlDataReader rd = dba.exec_sql("exc
sp_SearchUddiBySrv '" + Service + "'");
                    while(rd.Read())
                    {
                          _uddi.PID = rd.GetInt32(0);
                          _uddi.PName = rd.GetString(1);
                          marrlst.Add(_uddi);
                    }
                    rd.Close();
              }
              return marrlst;
        }
        public bool get_uddi(int wsid)
        {
              try
              {
                    if(wsid != 0)
                    {
                          SqlDataReader rd = dba.exec_sql("exc
sp_getUDDIByProvider " + wsid);

                          rd.Read();
                          _providerID = rd.GetInt32(2);
                          _location = rd.GetString(3);
                          //_SKey = rd.GetInt32(0);
                          _sname = rd.GetString(1);
                    }
                    return true;
              }
              catch(Exception ex)
              {
                    throw new Exception(ex.Message);
              }
        }
        public bool search_uddi_net(string sUri)
        {
              Uri mUri = new Uri(sUri);

              string HostName = mUri.Host;
              string Descrb = mUri.AbsolutePath;
              if(HostName != "")
              {
                    _providerID = dba.exec_cmd("exec sp_insProvider '" +
HostName + "','" + Descrb + "'");
              }
```

```
            return true;

        }

        /*************************************************************/
        #endregion public methods

        #region private methods
        /*************************************************************/

        /*************************************************************/
        #endregion private methods
    }
}
```

APPENDIX IV

Travel Client Web application source code

WebForm1.aspx

```
<%@ Page language="c#" Codebehind="WebForm1.aspx.cs" AutoEventWireup="false"
Inherits="Travel.WebForm1" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <title>WebForm1</title>
        <meta content="Microsoft Visual Studio .NET 7.1" name="GENERATOR">
        <meta content="C#" name="CODE_LANGUAGE">
        <meta content="JavaScript" name="vs_defaultClientScript">
        <meta content="http://schemas.microsoft.com/intellisense/ie5"
name="vs_targetSchema">
    </HEAD>
    <body>
        <form id="Form1" method="post" runat="server">
            <div align="center"><asp:panel id="Porders" BackColor="#66ffff"
Runat="server">
                <TABLE width="66%">
                    <TR>
                        <TD align="center" colSpan="2"><B>Find
a Package</B></TD>
                    </TR>
                    <TR>
                        <TD style="WIDTH: 196px"
align="right">Destination:</TD>
                        <TD>
                            <asp:textbox id="txt_date"
runat="server"></asp:textbox></TD>
                    </TR>
                </TABLE>
                <TABLE width="66%">
                    <TR>
                        <TD align="center">
                            <asp:button id="ButSinvoke"
runat="server" Text="SInvoke"></asp:button></TD>
                        <TD align="center">
```

```
                                                    <asp:button id="ButDinvoke"
runat="server" Text="DInvoke"></asp:button></TD>
                                        </TR>
                                    </TABLE>
                                    <TABLE>
                                        <TR>
                                            <TD>
                                                <asp:label id="lblResult"
runat="server" Font-Size="X-Small"></asp:label></TD>
                                        </TR>
                                    </TABLE>
                                </asp:panel><br>
                                <asp:panel id="Panel2" BackColor="#66ffff" Runat="server">
                                    <TABLE width="66%">
                                        <TR>
                                            <TD align="center" colSpan="2"><B>Get
Order Status</B></TD>
                                        </TR>
                                        <TR>
                                            <TD style="WIDTH: 196px"
align="right">Order Number:</TD>
                                            <TD>
                                                <asp:textbox id="txtOrdNbr"
runat="server"></asp:textbox></TD>
                                        </TR>
                                    </TABLE>
                                    <TABLE width="66%">
                                        <TR>
                                            <TD align="center">
                                                <asp:button id="ButSinv"
runat="server" Text="SInvoke"></asp:button></TD>
                                            <TD align="center">
                                                <asp:button id="ButDinv"
runat="server" Text="DInvoke"></asp:button></TD>
                                        </TR>
                                    </TABLE>
                                    <TABLE>
                                        <TR>
                                            <TD>
                                                <asp:label id="lblResult1"
runat="server" Font-Size="X-Small"></asp:label></TD>
                                        </TR>
                                    </TABLE>
                                </asp:panel><br>
                                <asp:panel id="Panel3" BackColor="#66ffff" Runat="server">
                                    <TABLE width="66%">
```

```
                                        <TR>
                                            <TD align="center"
colSpan="2"><B>Cancel Order </B>
                                            </TD>
                                        </TR>
                                        <TR>
                                            <TD style="WIDTH: 196px"
align="right">Order Number:</TD>
                                            <TD>
                                                <asp:textbox id="txtOrdNbr1"
runat="server"></asp:textbox></TD>
                                        </TR>
                                    </TABLE>
                                    <TABLE width="66%">
                                        <TR>
                                            <TD align="center">
                                                <asp:button id="ButSinvk"
runat="server" Text="SInvoke"></asp:button></TD>
                                            <TD align="center">
                                                <asp:button id="ButDinvk"
runat="server" Text="DInvoke"></asp:button></TD>
                                        </TR>
                                    </TABLE>
                                    <TABLE>
                                        <TR>
                                            <TD>
                                                <asp:label id="lblResult2"
runat="server" Font-Size="X-Small"></asp:label></TD>
                                        </TR>
                                    </TABLE>
                                </asp:panel><br>
                                <asp:panel id="Panel1" BackColor="#66ffff" Runat="server">
                                    <TABLE width="66%">
                                        <TR>
                                            <TD align="center"
colSpan="2"><B><B>Get </B> Orders </B>
                                            </TD>
                                        </TR>
                                        <TR>
                                            <TD style="WIDTH: 196px"
align="right">Order Date:</TD>
                                            <TD>
                                                <asp:textbox id="txt_orddate"
runat="server"></asp:textbox></TD>
                                        </TR>
                                    </TABLE>
```

```
                                    <TABLE width="66%">
                                        <TR>
                                            <TD align="center">
                                                <asp:button id="ButSinvO"
runat="server" Text="SInvoke"></asp:button></TD>
                                            <TD align="center">
                                                <asp:button id="ButDinvO"
runat="server" Text="DInvoke"></asp:button></TD>
                                        </TR>
                                    </TABLE>
                                    <TABLE>
                                        <TR>
                                            <TD>
                                                <asp:label id="lblResult3"
runat="server" Font-Size="X-Small"></asp:label></TD>
                                        </TR>
                                    </TABLE>
                                </asp:panel></div>
            </form>
    </body>
</HTML>
```

WebForm1.aspx.cs

```
using System;
using System.Reflection;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
//using DYInvoker;

namespace Travel
{
   /// <summary>
   /// Summary description for WebForm1.
   /// </summary>
   public class WebForm1 : System.Web.UI.Page
   {
         protected System.Web.UI.WebControls.Label lblResult;
         protected System.Web.UI.WebControls.Button ButSinvoke;
         protected System.Web.UI.WebControls.TextBox txt_date;
         protected System.Web.UI.WebControls.Panel Porders;
         protected System.Web.UI.WebControls.TextBox txtOrdNbr;
         protected System.Web.UI.WebControls.Label lblResult1;
         protected System.Web.UI.WebControls.Panel Panel3;
         protected System.Web.UI.WebControls.Panel Panel2;
         protected System.Web.UI.WebControls.TextBox txtOrdNbr1;
         protected System.Web.UI.WebControls.Button ButSinvk;
         protected System.Web.UI.WebControls.Button ButDinvk;
         protected System.Web.UI.WebControls.Label lblResult2;
         protected System.Web.UI.WebControls.Button ButSinvO;
         protected System.Web.UI.WebControls.Button ButDinvO;
         protected System.Web.UI.WebControls.Button ButSinv;
         protected System.Web.UI.WebControls.Button ButDinv;
         protected System.Web.UI.WebControls.TextBox txt_orddate;
         protected System.Web.UI.WebControls.Panel Panel1;
         protected System.Web.UI.WebControls.Label lblResult3;
         protected System.Web.UI.WebControls.Button ButDinvoke;

         private void Page_Load(object sender, System.EventArgs e)
         {
               // Put user code to initialize the page here
         }

         #region Web Form Designer generated code
         override protected void OnInit(EventArgs e)
         {
               //
               // CODEGEN: This call is required by the ASP.NET Web Form
Designer.
               //
               InitializeComponent();
```

```
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.ButSinvoke.Click += new
System.EventHandler(this.ButSinvoke_Click);
            this.ButDinvoke.Click += new
System.EventHandler(this.ButDinvoke_Click);
            this.ButSinv.Click += new
System.EventHandler(this.ButSinv_Click);
            this.ButDinv.Click += new
System.EventHandler(this.ButDinv_Click);
            this.ButSinvk.Click += new
System.EventHandler(this.ButSinvk_Click);
            this.ButDinvk.Click += new
System.EventHandler(this.ButDinvk_Click);
            this.ButSinvO.Click += new
System.EventHandler(this.ButSinvO_Click);
            this.ButDinvO.Click += new
System.EventHandler(this.ButDinvO_Click);
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion

        private void ButSinvoke_Click(object sender, System.EventArgs e)
        {
            VTAEnt.Provider mpd = new VTAEnt.Provider();
            lblResult.Text = mpd.GetOrders(txt_date.Text);
        }

        private void ButDinvoke_Click(object sender, System.EventArgs e)
        {
            try
            {
                lblResult.Text = get_result("str", "GetOrders",
"Provider", txt_date.Text).ToString();
            }
            catch(Exception ex)
            {
                lblResult.Text = ex.Message;
            }

        }
        private void ButSinv_Click(object sender, System.EventArgs e)
        {
            VTAEnt.Provider mpd = new VTAEnt.Provider();
            lblResult1.Text =
mpd.orderStatus(Convert.ToInt32(txtOrdNbr.Text));
        }

        private void ButDinv_Click(object sender, System.EventArgs e)
```

```
        {
            try
            {
                lblResult1.Text = get_result("int", "orderStatus",
"Provider", txtOrdNbr.Text).ToString();
            }
            catch(Exception ex)
            {
                lblResult.Text = ex.Message;
            }
        }

        private void ButSinvO_Click(object sender, System.EventArgs e)
        {
            VTAEnt.Provider mpd = new VTAEnt.Provider();
            lblResult3.Text =
mpd.allOrders(txt_orddate.Text)[0].custName;
        }

        private void ButDinvO_Click(object sender, System.EventArgs e)
        {
            try
            {
                lblResult3.Text = get_result("str", "allOrders",
"Provider", txt_orddate.Text).ToString();
            }
            catch(Exception ex)
            {
                lblResult3.Text = ex.Message;
            }
        }

        private void ButSinvk_Click(object sender, System.EventArgs e)
        {
            VTAEnt.Provider mpd = new VTAEnt.Provider();
            lblResult2.Text =
mpd.cancelOrder(Convert.ToInt32(txtOrdNbr1.Text)).ToString();
        }

        private void ButDinvk_Click(object sender, System.EventArgs e)
        {
            try
            {
                lblResult2.Text = get_result("int", "cancelOrder",
"Provider", txtOrdNbr1.Text).ToString();
            }
            catch(Exception ex)
            {
                lblResult2.Text = ex.Message;
            }
        }

        private object get_result(string ptype, string methodname, string
typename, string parm)
        {
            string wsdlUrl = "http://esszz/VTA_Ent/Provider.asmx?wsdl";
            object obj;
```

```
                DateTime dt = new DateTime(2005, 4, 12);
                DYInvoker.Invocation Inv = new DYInvoker.Invocation();
                DYInvoker.Stub stub = new DYInvoker.Stub(1, dt, wsdlUrl,
typename);

                Inv.MethodName = methodname;
                if (ptype == "str")
                {
                      Inv.AddParameter(Convert.ToString(parm));
                }
                else if (ptype == "int")
                {
                      Inv.AddParameter(Convert.ToInt32(parm));
                }
                Inv.WsdlUrl = wsdlUrl;
                obj = ((object[])Inv.InvokeCall(stub, methodname))[0];
                Type tp = obj.GetType();
                string str1 =
(string)tp.GetField("custName").GetValue(obj);
                //FieldInfo[] flds;
                //flds = tp.GetFields();
                return obj;
          }


    }
}
```

VITA

Zhitong Zhao was born in Tianjin, China on March 19, 1971. After completing his work at Tianjin NO.51 High School, Tianjin, China in 1990, he came to the United States. He earned a computer science Bachelor of Science degree from Southwest Texas State University, San Marcos, Texas in December 1998. In June 2001, he entered the Graduate School of Texas State University – San Marcos and worked on his degree of Master of Science, San Marcos, Texas.

Permanent Address: 1229 Vincent PL,

Pflugerville, TX, 78660

This thesis was typed by Zhitong Zhao.