CHAPTER I

INTRODUCTION

The emerging Service-Oriented Computing (SOC) paradigm promises to enable businesses and organizations to collaborate in an unprecedented way by means of standard web services. To support rapid and dynamic composition of services in this paradigm, web services that meet requesters' functional requirements must be able to be located and bounded dynamically from a large and constantly changing number of service providers based on their Quality of Service (QoS). In order to enable quality-driven web service selection, we need an open, fair, dynamic and secure framework to evaluate the QoS of a vast number of web services. The fair computation and enforcing of QoS of web services should have minimal overhead but yet able to achieve sufficient trust by both service requesters and providers. In this thesis, we presented our open, fair and dynamic QoS computation model for web services selection through implementation of and experimentation with a QoS registry in a hypothetical phone service provisioning market place application.

1.1 Background

Web services [17] is becoming a very hot topic in today's IT industry. Microsoft, IBM and BEA have invested enormously in this technology. Industry

standards about web services are becoming available for businesses and organizations to plan their IT projects to take advantage of this technology. Web service is moving at a very fast pace, many new concepts and industry standards are proposed to make the technology to be more capable to carry on business transactions efficiently. Recently, Microsoft, IBM and BEA have developed a new standard of Web Services Transaction, which supports the transactions among web services. This will give the technology of web services the ability to implement atomic transaction business processes among various web service applications across different platforms. Other new development of web service is the introducing of Web Service Flow Language (WSFL) [22], which is an XML language to composite multiple web services. By using this standard language to construct new processes based on existed web service applications, you eliminate the need to build new web service interfaces every time when you can use those which are working well for many years. WSFL is good for B2B integration because you can easily to construct a new application based on web services across many organizations and businesses.

To help us to understand why the web service becomes so important in IT industry, we need to take a look at today's Business-to-Business integration (B2B). Business-to-Business integration has become more important than ever before in today's IT industry. The development of Internet, the availability of broad bandwidth, new industry standards such as XML-based Web Service and a new wave of innovations have made B2B integration more challenging, and the darling of the IT industry. The force behind the current popularity of

B2B integration is the new way of doing business—automating business transactions among business organizations through the Internet and the needs of cutting the cost of developing, maintaining and integrating enterprise applications. The new evolved technology and industry standards, the new approach of application design, the difference between various developing tools and platforms have added huge complexity to the B2B Integration. The center of all attentions about B2B today is the XML-based web service. It's the key for future web-oriented business-to-business model and automation of business integration among business partners. Web services are fundamentally about interoperability, a key element in B2B, especially when those applications are developed using different languages, tools and platforms. Web services offer a widely adopted mechanism for making application work together.

Web service is not only a technology for tomorrow, but also for today. The search engine Google [15, 23] has a web service which you can easily to integrate into your application. Instantly, you application has world-class searching ability by integrating that web service interface. Many businesses in financial service offer web service interfaces for their service which can be consumed by other businesses for a fee. Microsoft has already offered many of its service to its business customers in web service. The number of potential users of this technology is enormous, for example, the health industry and travel industry are all talking about using web service in their future development or convert their existed ones into web services. The benefits of

this approach can be enjoyed greatly by the IT industry for many years to come. The list of those early web service applications can be very long, the point is that web service is the real thing.

Behind all those wonderful words about this technology, we need to investigate all the aspects of this technology such as performance, security, availability and reliability etc. How to measure the web service is very important for us to understand and improve this new technology. Furthermore, this will help us to come up with a better way to use and integrate web services. In this thesis, we will conduct our research on the QoS modeling in web service.

1.2 Definition

In this section we will give some definitions for the terms which are used in this thesis.

1.2.1 What's Web Service

Although web service is so widely discussed and used in the IT industry, however, there isn't a single universal definition for web service. The only universal thing is the term of "web service". Major software vendors who just happened to be the creators of the industry standards have many different descriptions of the web service.

IBM gives its definition of web service at http://www4.ibm.com/software/solutions/Webservices/pdf/WSCA.pdf: "A Web service is an interface that describes a collection of operations that are network accessible through

standardized XML messaging. Web services fulfill a specific task or a set of tasks. A web service is described using a standard, formal XML notion, called its service description, that provides all of the details necessary to interact with the service, including message formats (that detail the operations), transport protocols, and location.

The nature of the interface hides the implementation details of the service so that it can be used independently of the hardware or software platform on which it is implemented and independently of the programming language in which it is written. This allows and encourages web services based applications to be loosely coupled, component-oriented, cross-technology implementations. Web services can be used alone or in conjunction with other web services to carry out a complex aggregation or a business transaction."

Sun also gives its version of definition at http://www.sun.com/software/suneone/faq.html#2: "Web services are software components that can be spontaneously discovered, combined, and recombined to provide a solution to the user's problem/request. The Java language and XML are the prominent technologies for Web services."

Microsoft offers its definition of web service at http://msdn.microsoft.com/library/default..asp?url=url=/nhp/Default.asp?contentid=28000442:

"A web service is a unit of application logic providing data and services to other applications. Applications access Web service via ubiquitous Web protocols and data formats such as HTTP, XML, and SOAP, with no need to worry about how each Web service is implemented. Web services combine the best aspects

of component-based development and the Web, and are a cornerstone of the Microsoft .NET programming model."

We can see that there is a universal agreement on what the web service can do or might be, although Sun and Microsoft have added the connections between their products and the web services into the definition of web service. However, there is no uniformed definition of the web services from what we have read from their descriptions. Here, by combining what they have said about web service, we can give some more general description of web service:

1. Platform independent.
2. Described using an XML-based service description language.
3. Published to a service registry.
4. Can be composed with other web service into a new application or another web service.
5. Can be invoked through interface.
6. User doesn't need the implementation details to invoke the service.
7. Exchanged message should be XML-based.

1.2.2 What's QoS

Quality of service (QoS) is a term that has been used for a long time in computer science. The term has been used to describe many subjects such as QoS of communication network and QoS of web page [25] etc. Let's take a look at how it is used in the field of communication network. The term of QoS refers the ability of a network to deliver predictable results. Measured Metrics in

QoS of network performance often include availability, reliability, bandwidth and latency etc. QoS is monitored to ensure that network is performing at the desired level. QoS can be used in a contract for network service provider to guarantee its customer the level of service will be delivered [24].

From what we have found from the usage of the term of QoS in other fields, we can see that QoS is a set of metrics to describe a certain process or subject which reflects the state or characteristics of the described process or subject.

1.2.3 What's XML

XML stands for Extensible Markup Language. It was standardized by World Wide Web Consortium (W3C). XML is a tag-based language like HTML. It is extensible, users can describe content of their documents which are designed by their own way without the anticipation of the original language designer. It is platform independent, and used to exchange data across many businesses or organizations. It becomes the de facto standard for information exchange on the internet.

1.2.4 What's XML Schema

XML schema is an XML file used to describe the structure of a type of XML file. It is very important for message exchanging between different applications. It is a meta-language to describe the structure of XML document and map the syntax and data type in XML documents. It's the building block of the web service technology [14].

1.3 Motivation

As we discussed previously, web service is a very important technology, so we need to study how to use it in various applications. QoS in web service can be such a tool to help us to use the web service in a much efficient manner. Some research has done on how to compose web service work flow based on QoS [12]. Without our understanding of QoS in web service, it will be very difficult for us to accomplish what we intend to do. Research of quality of service in web service is an active research area which has not been full investigated. Most research in this area lacks the implementation details of the QoS metrics, which means few QoS model gives practical method to measure their proposed QoS metrics. Therefore, no previous proposed model has been implemented and there isn't enough data to be collected to conduct meaningful experiments. The needs of dynamic and business related QoS modeling is not given enough attention in prior research.

In this thesis, some researches on an extensible and business-oriental QoS model which is dynamic in nature, will be conducted from many aspects. The research will focus on the computation method of QoS and the mechanism to maintain an open, fair and secure QoS model. The extensible QoS computation method should have the ability to add new QoS metrics easily without changing underlying algorithm. In other words, the computation model should be as universal as possible in every business domain. The method should also allow business-related QoS metrics to be composed into QoS computation. The new proposed model should overcome short-coming of prior

models. The implementation of the proposed QoS model and other applications should maintain a high level of standard in Business-to-Business integration. The implementation should also have user-friendly interfaces to conduct analysis and experiments.

1.4 Methods

In this thesis, our proposed QoS model will be compared with previous QoS models. Analysis and experiments will be conducted from many aspects of the QoS model. The advantage and disadvantage of our QoS model will be discussed in details in our thesis. The QoS model and some applications will be implemented to support the analysis and experiments. Simulation of human usage of the implemented system for the experiments will be available as well.

1.5 Significance of the Problem

Web services are self-describing software applications that can be advertised, located, and used across the Internet using a set of standards such as SOAP [18], WSDL [17], and UDDI [16]. Web services encapsulate application functionality and information resources, and make them available through standard programmatic interfaces. Web services are viewed as one of the promising technologies that could help business entities to automate their operations on the web on a large scale by automatic discovery and consumption of services. Business-to-Business (B2B) integration can be achieved on a demand basis by aggregating multiple services from different

providers into a value-added composite service. In the last two years, the process-based approach to web service composition has gained considerable momentum and standardization [1]. However, with the ever increasing number of functional similar web services being made available on the Internet, there is a need to be able to distinguish them using a set of well-defined Quality of Service (QoS) criteria. A service composition system that can leverage, aggregate and make use of individual component's QoS information to derive the optimal QoS of the composite service is still an ongoing research problem. This is partly due to the lack of an extensible QoS model and a reliable mechanism to compute and police QoS that is fair and transparent to both service requesters and providers.

Currently, most approaches that deal with QoS of web services only address some generic dimensions such as price, execution duration, availability and reliability [12, 6]. In some domains, such generic criteria might not be sufficient. QoS model should also include domain specific criteria and be extensible. Moreover, most of the current approaches rely on service providers to advertise their QoS information or provide an interface to access the QoS values, which is subject to manipulation by the providers. Obviously, service providers may not advertise their QoS information in a ``neutral'' manner, for example, execution duration, reliability, etc. In approaches where QoS values are solely collected through active monitoring, there is a high overhead since QoS must be checked constantly for a large number of web services. On the other hand, an approach that relies on a third party to rate or endorse a particular service

provider is expensive and static in nature. Other problems with prior models are that no business related metrics are included into the computation of QoS value, and prior models don't have the method to meet the dynamic needs of service providers and requesters.

1.6 Solutions to the Problem

In our framework, the QoS model is extensible, and QoS information can be provided by providers, computed based on execution monitoring by the users, or collected via requester's feedback, depending on the characteristics of each QoS criterion. In a nutshell, we will propose a framework that aims at advancing the current state of the art in QoS modeling, computation and policing.

1.7 Structure of the Thesis

The thesis is organized as follows. In Chapter 2, we discuss related work. In Chapter 3, we give the details of a service broker which supports of an extensible QoS model and its computation. In Chapter 4, we describe the implementation of QoS registry and other applications, and explain how QoS information can be collected based on active monitoring and active user feedback. Chapter 5 discusses the experiments that we conducted to understand the relationships between the QoS and the business criteria and study the effectiveness of our proposed QoS model. Finally, we conclude in Chapter 6.

CHAPTER II

RELATED WORK

Multiple web services may provide similar functionality, but with different non-functional properties. In the selection of a web service, it is important to consider both functional and non-functional properties in order to satisfy the constraints or needs of users. While it is common to have a QoS model for a particular domain, an extensible QoS model that allows addition of new quality criteria without affecting the formula used for the overall computation of QoS values has not been proposed before. Moreover, very little research has been done on how to ensure that service quality criteria can be collected in a fair, dynamic and open manner. Most previous work is centered on the collection of networking level criteria such as execution time, reliability and availability etc [12]. No model gives details on how business criteria such as compensation and penalty rates can be included in QoS computation and enforced in an objective manner.

In a dynamic environment, service providers can appear and disappear around the clock. Service providers can change their services at any time in order to remain competitive. Previous work has not addressed the dynamic aspects of QoS computation. For example, there is no guarantee that the QoS obtained at run time for a particular provider is indeed the most up to date

value. Our proposed QoS computation which is based on active monitoring and consumers' feedback ensures that QoS value of a particular provider is always up to date.

2.1 QoS Metrics in Web Service

In [27], the author gives a long list of QoS metrics which covers most quality criteria in web service. They are listed as following:

1. Availability: Availability is the quality aspect of whether the web service is present or ready for immediate use.

2. Accessibility: Accessibility is the quality aspect of a service that represents the degree it is capable of serving a web service request.

3. Integrity: Integrity is the quality aspect of how the web service maintains the correctness of the interaction in respect to the source.

4. Performance: Performance is the quality aspect of web service, which is measured in terms of throughput and latency.

5. Regulatory: Regulatory is the quality aspect of web service in conformance with the rules, the law, compliance with standards, and the established service level agreement.

6. Security: Security is the quality aspect of the web service of providing confidentiality and non-repudiations by authenticating the parties involved, encrypting messages, and providing assess control.

In [27], the author also discusses the transactional QoS which refers to the quality of transactions are executed. The author doesn't give details of how to

measure his proposed QoS metrics. If it is not impossible, then it will be extremely difficult to measure some QoS metrics in this paper in the real world.

In [6,9,12], the same list of quality criteria are discussed in details. They are facing the same problem, that their researches lack the method to measure the QoS metrics. Therefore, nothing can be implemented based on their work. In [12], the author gives a web service quality model which is a general model to represent QoS in web service from multiple dimensions. However, this paper focuses mainly on web services composition based on QoS.

From what we have surveyed about QoS in web service, no business related quality criterion is included in previous work. This is one of the areas we want to improve in our QoS model.

2.2 Prior QoS Models

In [9], the author proposed a QoS model which has a QoS certifier to verify published QoS criteria. It requires all web service providers to advertise their services with the QoS certifier. The certifier has to check the service provider's QoS claims. The results with a QoS certification ID will be sent back to the service provider. This approach lacks the ability to meet the dynamics of a market place where the needs of both consumers and providers are constantly changing. For example, it does not provide methods for providers to update their QoS dynamically, and does not give the most updated QoS to service consumers as well. There are no details on how to verify the QoS with the service providers in an automatic and cost effective way.

In [10], the authors proposed a QoS middleware infrastructure which required a build-in tool to monitor metrics of QoS automatically. If such a tool can be built, it needs to poll all web services to collect metrics of their QoS. Such an approach requires the willingness of service providers to surrender some of their autonomy. Moreover, if the polling interval is set too long, the QoS will not be up to date. If the polling interval is set too short, it might incur a high performance overhead. A similar approach which emphasizes on service reputation is proposed in [4, 5]. A web service agent proxy is set up to collect reputation rating from previous usage of web services. The major problem with this approach is that, there is no mechanism in place to prevent false ratings being collected. We have a mechanism to ensure that only the true user of the service is allowed to provide feedback.

In [3], the author discusses a model with service level agreement (SLA) which is used as a bridge between service providers and consumers. The penalty concept is given in its definition of SLA. The main emphasis about this concept is what should be done if the service provider can not deliver the service under the defined SLA, and the options for consumers to terminate the service under the SLA. Penalty is not included as a service quality criterion which can be used in QoS computation. A SLA simulation and analysis tool was implemented. This tool can be used to conduct experiments about the proposed SLA model. This paper discusses the “Dynamic service ranking” which is a concept of ranking services based on customer’s real time needs. The paper doesn’t give a practical method to compute the ranking of services.

In [2], a language-based QoS model is proposed. Here, QoS of a web service is defined using a Web Service QoS Extension Language. This XML-based language can define components of QoS in a schema file which can be used across different platforms and is human readable.

In [11], the author discusses Web related metrics. It's very interesting to see that this paper introduced "Quality of Experience" (QoE) and "Quality of Business" (QoBiz). As they are proposed, QoE is related to metrics such as response times and reliability of a service which are experienced by users. QoBiz is expressed as money related metrics such as execution price etc. Here, QoS metrics are metrics such as availability and performance. A QoBiz evaluation framework is proposed in this paper as well. The framework gives the relationships among the QoS, QoE and QoBiz. In short, by combining the QoS and QoE, the customer of the service produces the QoBiz inside the framework. In most cases, the QoE and QoBiz are simply put into the category of QoS. However, this paper offers a different point of view about QoS metrics about web related services.

CHAPTER III

QOS-BASED SERVICE SELECTION

Currently, in most SOC frameworks, there is a service broker which is responsible for the brokering of functional properties of web services. In our framework, we extend capability of the service broker to support non-functional properties. This is done by introducing an extensible and multiple dimensions QoS model in the service broker. We aim to evaluate QoS of web services using an open, fair and transparent mechanism. In the following subsections, we first introduce the extensible QoS model, and then we give the details on how to evaluate web service based on our model.

3.1 Extensible QoS Model

In this section, we propose an extensible QoS model, which includes generic and domain or business specific criteria. The generic criteria are applicable to all web services, for example, their pricing and execution duration. Although the number of QoS criteria discussed in this paper is limited (for the sake of illustration), our model is extensible. New criteria (either generic or domain specific) can be added without fundamentally altering the underlying computation mechanism as shown in Section 3.2. In particular, it is possible to extend the quality model to integrate non-functional service characteristics such

as those proposed in [7], or to integrate service QoS metrics such as those proposed by [11].

In the presence of multiple web services with overlapping or identical functionality, service requesters need objective QoS criteria to distinguish one service from another. We argue that it is not practical to come up with a standard QoS model that can be used for all web services in all domains. This is because QoS is a broad concept that can encompass a number of context-dependent non-functional properties such as privacy, reputation and usability. Moreover, when evaluating QoS of web services, we should also take into consideration of domain specific criteria. For example, in the domain of phone service provisioning, penalty rate for early termination of a contract and compensation for non-service, offered in the service level agreement are important QoS criteria in that domain. Therefore, we propose an extensible QoS model that includes both the generic and domain specific criteria. In our approach, new domain specific criteria can be added and used to evaluate the QoS of web services without changing the underlying computation model.

3.1.1 Preference-oriented Service Ranking

Different users may have different preferences or requirements on QoS. It is important to be able to represent QoS from the perspective of service requesters' preference. For example, service selection may be driven completely by price, regardless of the time it takes to execute the service. A different requester may be very service sensitive. This means that criteria such

as penalty rate or the ability to return the goods after the purchase are viewed as more important than the price and the time. Another service selection may be driven completely by time because of tight deadlines. A QoS model should provide means for users to accurately express their preferences without resorting to complex coding of user profiles.

3.1.2 Fair and Open QoS Computation

Once a set of QoS criteria have been defined for a particular domain, we must ensure that QoS information is collected in a fair manner. In our framework, QoS information can be collected from service properties that are published by providers, execution monitoring, and requesters' feedback based on the characteristic of quality criterion. For example, execution price can be provided by service providers, execution duration can be computed based on service invocation instances, while service reputation is based on service requesters' feedback. We also build a policing mechanism that will prevent the manipulation of QoS value from a single service requester by requiring each requester to have a valid pair of user id and password to update the QoS registry. Furthermore, this pair of ID and password must be verified by the service providers at the consumption of the service to ensure that only the actual consumer of the service is allowed to give feedback. On the other hand, to be completely fair, providers can review those criteria and can improve their QoS if they desire to. Moreover, providers can update their QoS information

(e.g., execution price, penalty rate) at any time. Providers can also check the QoS registry to see how their QoS is ranked among other service providers.

We believe that an extensible, transparent, open and fair QoS model is necessary for the selection of web services. Such a model can benefit all participants. Although this model requires all requesters to update their usage experiences with a particular type of service in a registry, this overhead on the user is not large. In return, QoS for a particular service is actively being policed by all requesters. Each requester can search the registry to get the most-updated QoS of listed providers. Service providers can view and change their services at any time. With such an open and dynamic definition of QoS, a provider can operate its web services to give its end-users the best user experience.

3.1.3 Generic Quality Criteria

We consider three generic quality criteria which can be measured objectively for elementary services: (1) *execution price* (2) *execution duration*, and (3) *reputation*. Criteria such as availability and reliability are not required in our model due to the use of active user feedback and execution monitoring.

1. Execution price

This is the amount of money which a service requester has to pay to the service provider to use a web service such as checking a credit, or the amount of money the service requester has to pay to the service provider to get a

commodity like an entertainment ticket or a monthly phone service. Web service providers either directly advertise the execution price of their services, or they provide means for potential requesters to inquire about it. Let *S* be one web service, then $q_{pr}(s)$ is the execution price for using that service *S*.

2. Execution duration

The execution duration $q_{du}(s)$ measures the expected delay in seconds between the moment when a request is sent and the moment when the service is rendered. The execution duration is computed using the expression $q_{du}(s)$ = $T_{process}(s)$ + $T_{trans}(s)$, meaning that the execution duration is the sum of the processing time $T_{process}(s)$ and the transmission time $T_{trans}(s)$. Execution time is obtained via active monitoring.

3. Reputation

The reputation $q_{rep}(s)$ of a service S is a measure of its trustworthiness. It mainly depends on end user's experiences of using the service S. Different end users may have different opinions on the same service. The value of the reputation is defined as the average ranking given to the service by end users, i.e., $q_{rep} = \frac{1}{n}\sum_{i=1}^{n} R_i$ where R_i is the end user's ranking on a service's reputation, n is the number of times the service has been graded. Usually, end users are given a range to rank Web services. For example, in Amazon.com, the range is [0, 5].

3.1.4 Business Related Criteria

The number of business related criteria can vary in different domains. For example, in phone service provisioning domain, the penalty rate for the early termination and the fixed monthly charge are important factors for users to consider when selecting a particular service provider. We use the generic term ***usability*** to group all business related criteria. In our chosen application, we measure usability from three aspects, transaction, compensation rate and the penalty rate.

1. Transaction

Transaction is very important in today's computer environment to maintain data consistency. It is widely used in mission critical fields such as trading and banking systems etc. We can also see that it becomes an important issue for web services composition as well. There is a need for atomic transaction among web services across organizations. A transaction standard for web services was proposed by IBM, Microsoft and BEA in their "Specification: Web Services Transaction" in August 2002 [21].

Transaction support is used for maintaining data consistency. In prior QoS models, no transactional criteria are being used in the computation of QoS value. However, from the perspective of a requester, whether a web service provides an undo procedure to roll back the service execution in certain period without any charges is an important factor that will affect his/her choice. Transactional property can be evaluated by two dimensions: whether undo

procedure is supported $q_{tx}(s)$ and what's the time constraints $q_{cons}(s)$ on undo procedure. It should be noted that $q_{tx}(s)$ = 0/1, where 1 indicate the web service supports transaction and 0 otherwise; $q_{cons}(s)$ indicates the duration for which the undo procedure is allowed.

2. Compensation Rate

Compensation rate $q_{comp}(s)$ of a web service indicates percentage of the original execution price that will be refunded when the service provider can not honor the committed service or deliver the ordered commodity.

3. Penalty Rate

Penalty rate $q_{pen}(s)$ a web service indicates what percentage of the original price service requesters need to pay to the provider when he/she wants to cancel the committed service or ordered commodity after the time out period for transaction to roll back is expired.

3.2 Web Service QoS Computation

The QoS registry is responsible for the computation of QoS value for each service provider. Assuming that there is a set of web services that have the same functional properties, where $S(S = \{s_1, s_2, ..., s_n\})$, web service computation algorithm determines which service s_i is selected based on end user's constraints. Using *m* criteria to evaluate web service, we can obtain the

following matrix **Q**. Each row in **Q** represents a web service s_i, while each column represents one of the QoS criteria.

$$Q = \begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,m} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ q_{n,1} & q_{n,2} & \cdots & q_{n,m} \end{pmatrix} \quad (1)$$

In order to rank the web services, the matrix **Q** needs to be normalized. There are two phases of normalization before we can compute the final **QoS** value.

1. First Normalization

Before normalizing matrix Q, we need to define two arrays. The first array is N = $\{n_1, n_2, \ldots, n_m\}$ with $1 \leq n_j \leq m$. The value of n_j can be 0 or 1. n_j = 1 is for the case where the increase of $q_{i,j}$ benefits the service requester while n_j = 0 is for the case where the decrease of $q_{i,j}$ benefits the service requester. The second array is C = $\{c_1, c_2, \ldots, c_m\}$. Here c_j is a constant which sets the maximum normalized value for j*th* criterion in matrix Q. Each element in matrix Q will be normalized using the following equation (2) and equation (3).

$$v_{i,j} = \begin{cases} \dfrac{q_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} q_{i,j}} & if \ \frac{1}{n}\sum_{i=1}^{n} q_{i,j} \neq 0, \dfrac{q_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} q_{i,j}} \leq c_j, n_j = 1 \\ c_j \quad if \ \frac{1}{n}\sum_{i=1}^{n} q_{i,j} = 0, n_j = 1, or \dfrac{q_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} q_{i,j}} \geq c_j \end{cases} \tag{2}$$

$$v_{i,j} = \begin{cases} \dfrac{\frac{1}{n}\sum_{i=1}^{n} q_{i,j}}{q_{i,j}} & if \ \dfrac{\frac{1}{n}\sum_{i=1}^{n} q_{i,j}}{q_{i,j}} \leq c_j, q_{i,j} \neq 0, n_j = 0 \\ c_j \quad if \quad q_{i,j} = 0, n_j = 0, or \dfrac{\frac{1}{n}\sum_{i=1}^{n} q_{i,j}}{q_{i,j}} \geq c_j \end{cases} \tag{3}$$

In the above equations, $\frac{1}{n}\sum_{i=1}^{n} q_{i,j}$ is the average value of j*th* quality criterion in matrix Q. Applying these two equations to Q, we get matrix Q' which is shown below:

$$Q' = \begin{pmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,m} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ v_{n,1} & v_{n,2} & \cdots & v_{n,m} \end{pmatrix} \tag{4}$$

Example 1

Data in the following example is taken from our QoS registry implementation. We assume that there are two web services in S, and that their values of quality criteria are: Q= ($q_{1,1}$, $q_{1,2}$, $q_{1,3}$, $q_{1,4}$, $q_{1,5}$, $q_{1,6}$, $q_{1,7}$, $q_{2,1}$, $q_{2,2}$, $q_{2,3}$, $q_{2,4}$,

$q_{2,5}$, $q_{2,6}$ $q_{2,7}$) =(25, 1, 60, 0.5, 0.5, 100, 2.0, 40, 1, 200, 0.8, 0.1, 40, 2.5). The quality criteria are in the order of Price, Transaction, Time Out, Compensation Rate, Penalty Rate, Execution Duration and Reputation. For array N, since the increase of price, penalty rate and execution duration doesn't benefit the service requester, and the increase of the remaining quality criteria benefits the service requester, so the value for each element in array N is {0,1,1,1,0,0,1}. For array C, each element in the array is set to 5 which is considered as a reasonable maximum normalized value in this implemented domain. Using equation (2) and equation (3), we have the normalized matrix: Q'= ($v_{1,1}$, $v_{1,2}$, $v_{1,3}$, $v_{1,4}$, $v_{1,5}$, $v_{1,6}$, $v_{1,7}$, $v_{2,1}$, $v_{2,2}$, $v_{2,3}$, $v_{2,4}$, $v_{2,5}$, $v_{2,6}$ $v_{2,7}$) = (1.3, 1.0, 0.462, 0.769, 0.64, 0.7, 0.8894, 0.8134, 1.0, 1.538, 1.23, 3.0, 1.75, 1.111).

2. Second Normalization

In our QoS model, each quality criterion belongs to a certain group such as price, reputation and usability etc. Each group can contain multiple criteria. For example, both compensation and penalty rates belong to the usability group. To compute the final QoS value for each web service, we introduce Matrix D and Matrix G. Matrix D is used to define the relationship between quality criteria and quality groups. Each row in Matrix D represents a quality criterion, and each column in Matrix D represents one quality group value. Matrix G represents QoS information based on the values of quality groups of web services. Each row in Matrix G represents a web service, and each column in Matrix G represents one quality group value. Matrix D and Matrix G are shown below:

$$D = \begin{pmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,l} \\ d_{2,1} & d_{2,2} & \dots & d_{2,l} \\ \vdots & \vdots & \vdots & \vdots \\ d_{m,1} & d_{m,2} & \dots & d_{m,l} \end{pmatrix} \quad (5)$$

$$G = \begin{pmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,l} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,l} \\ \vdots & \vdots & \vdots & \vdots \\ g_{n,1} & g_{n,2} & \cdots & g_{n,l} \end{pmatrix} \quad (6)$$

Here, l is the total number of groups of quality criteria. For the value of each element in matrix D, $d_{i,j} = 1$ if the i*th* quality criterion in $Q^{'}$ is included in j*th* group in G. By applying Matrix D to $Q^{'}$, we have matrix G. The equation is shown below:

$$G = Q^{'} * D \quad (7)$$

To normalize matrix G, two arrays are needed. In the first array T= {t_1 , t_2 , …, t_l }, t_j is a constant which sets the maximum normalized value for the group j. In the second array F={ f_1 , f_2 , …, f_l }, f_j is a weight for group j such as price sensitivity and service sensitivity etc. This is used to express users' preferences over j*th* group. Each element in matrix G will be normalized using equation 8.

$$h_{i,j} = \begin{cases} \dfrac{g_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} g_{i,j}} & if\ \frac{1}{n}\sum_{i=1}^{n} g_{i,j} \neq 0, \dfrac{g_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} g_{i,j}} \leq t_j \\ t_j & if\ \frac{1}{n}\sum_{i=1}^{n} g_{i,j} = 0, or \dfrac{g_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} g_{i,j}} \geq t_j \end{cases} \quad (8)$$

In the above equations, $\frac{1}{n}\sum_{i=1}^{n} g_{i,j}$ is the average value of group criteria j in matrix G. Applying equation 8 to G, we get matrix G' which is shown below:

$$G' = \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,l} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,l} \\ \vdots & \vdots & \vdots & \vdots \\ h_{n,1} & h_{n,2} & \cdots & h_{n,l} \end{pmatrix} \quad (9)$$

Finally, we can compute the QoS value for each web service by applying array F to matrix G'. The formula of QoS is shown below:

$$Qos(s_i) = \sum_{j=1}^{l} (h_{i,j} * f_j) \quad (10)$$

Example 2

This example continues the computation of QoS from Example 1. For array T, each element in the array is set to 5 which is considered as a reasonable maximum normalized value in this implemented domain. For array F, each weight is given as 1. This means user gives the same preference to every

group. The defined D, which determines the grouping of the quality criteria in this example, is shown below:

$$G' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The column in D is in the order of Price, Time, Usability and Reputation. First, by applying D to matrix Q', we obtained G = (0.813, 1.750, 7.068, 1.111, 1.300, 0.700, 3.531, 0.889). Second, we apply equation 8 to G to have a normalized G'=($h_{1,1}$, $h_{1,2}$, $h_{1,3}$, $h_{1,4}$, $h_{2,1}$, $h_{2,2}$, $h_{2,3}$, $h_{2,4}$) = (0.769, 1.429, 1.334, 1.111, 0.946, 0.571, 0.666, 0.889). Finally, using equation 10, we have $Qos(s_1) = 4.643$ and $Qos(s_2) = 3.072$.

CHAPTER IV

IMPLEMENTATIONS
OF QOS REGISTRY, UPS PORTAL
AND SERVICE PROVIDERS

To demonstrate our proposed QoS model, we implemented a QoS registry as shown in Figure 1 within a hypothetical phone service (UPS) provisioning market place. The UPS market place is implemented using BEA Weblogic Workshop web service toolkit. It consists of various service providers who can register to provide various types of phone services such as long distance, local phone service, wireless and broadband. The market place also has a few credit checking agencies which offer credit checking service for a fee. The UPS market place has web interfaces which allow a customer to login and search for phone services based on his/her preferences. For example, the customer can specify whether the search for a particular type of service should be price or service sensitive. A price sensitive search will return a service provider who offers the lowest price. A service sensitive search will return a service provider with the best rated services.

The market place also has web interfaces that allow service providers to register their web services with the QoS registry, update their existing web

services in the registry or view their QoS ranking in the market place. The registry's other interfaces are available for requesters/end-users to give feedback on the QoS of the web services which they just consumed. This section only gives the functional overview of the system; the implementation details are available in the other following sections.

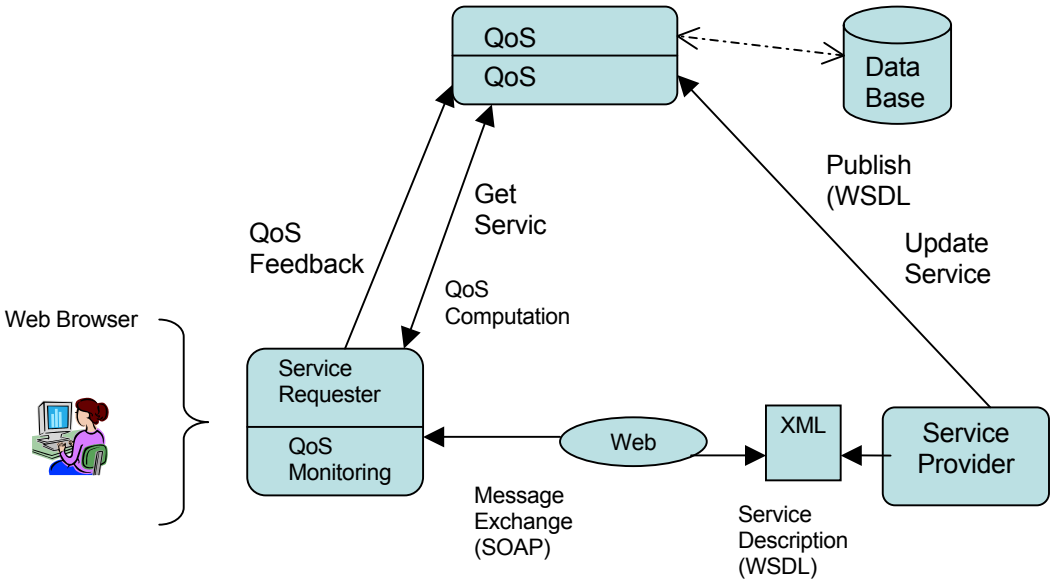


Figure 1: Architecture Diagram of QoS Registry

4.1 Collecting Service Quality Information

In our framework, we distinguish two types of criterion: *deterministic* and *non-deterministic*. Here, deterministic indicates that the value of QoS quality criterion is known or certain when a service is invoked, for example, the execution price and the penalty rate. The non-deterministic is for QoS quality criterion that is uncertain when web service is invoked, for example execution duration. For deterministic criterion, we assume that service providers have

mechanisms to advertise those values through means such as web service quality-based XML language as described in [2]. How all service providers come to an agreement of deterministic set of QoS criteria to advertise is beyond the scope of this thesis. In the next two sub-sections, we will discuss how all QoS criteria in our model can be collected in a fair, open and objective manner via active monitoring and active user's feedback.

1. Collecting Quality Information from Active Execution Monitoring.

The actual service execution duration is collected by the service requester. This requires interfaces used by all service requesters to implement some mechanisms to log the actual execution time. Although this approach puts more burden on the service requester, it has the following advantages over approaches where the service broker or QoS registry is required to perform the monitoring: 1) it lowers the overhead of QoS registry and simplifies the implementation of QoS registry, 2) data is collected from actual consumption of the service which is up to date and objective, 3) it avoids the necessity to poll a large number of service providers constantly.

Through active execution monitoring, when a service requester gets a different set of values for those deterministic criteria advertised by the service provider, this difference can be logged. If this phenomenon is common in a particular domain, we can expand the criteria for **Reputation** to record the difference between the actual deterministic quality criteria and the advertised

deterministic quality criteria from the service provider. The bigger the difference, the lower the QoS will be for that service provider.

2. Collecting Quality Information from Users Feedback

Each end-user is required to update QoS of service he/she has just consumed. This ensures that the QoS registry is fair to all end-users since QoS values can be computed based on real user experience with up to date runtime execution data. To prevent the manipulation of QoS by a single party, for each feedback, the end-user is given a pair of keys. This pair of keys must be authenticated by the service provider before the user is allowed to update the QoS value. The update must take place in a limited time-frame to prevent the system from being overloaded with un-consumed service requests.

4.2 Dynamics of QoS Model

To argue that the proposed model is open and transparent, we can not ignore the needs for the service providers to understand and use the information from QoS registry dynamically. It's very important for service providers to have the ranking of its service in the QoS registry instantly, since the providers can decide whether to improve their QoS value to attract more customers or even lower their QoS value to cut cost but still be able to maintain competitive based on the QoS information. The ability of our QoS model to let service providers query their QoS ranking among listed providers will help to

build a very competitive and live business community around the QoS registry, which will benefit all participants eventually.

4.3 Scenarios of QoS Model

To help us understand the proposed QoS model better, some scenarios are given in details in this section. The main goal of QoS computation is on providing an efficient and flexible way for web service consumers to find the right service provider from a large pool of similar providers in a well-defined and well-understood domain. We can view it as a web service broker which lets web service providers register themselves, advertise their services and update QoS information dynamically. On the other hand, it lets web consumers to have the ability to fine-tune their search to find the right web service and update the QoS of the web services that they have used.

4.3.1 Scenario 1

One customer is visiting UPS marketplace web site, after he/she sign up with the UPS, she/he wants to search for a long distance phone service provider. This is a kind of service which lasts for a period of time, so some providers will have options for user to cancel the service before the termination of the contracted term. The UPS marketplace will query QoS registry to get a list of providers. This search passes a list of parameters with, type of web service, number of providers and option of price sensitive or service sensitive to the registry. In this scenario, the type of service will be “Telecom”, number of

provider will be the number pre-defined by the UPS marketplace and the option of price sensitive or service sensitive is set to 1. The registry will compute the QoS of its registered providers using the QoS computation component, and return the list of providers in an XML file with information about those providers such as name, address, end point URL etc.

4.3.2 Scenario 2

A customer is visiting UPS marketplace web site, he wants to search for a long distance phone service provider. He is looking for the lowest price for such a service. This search will have the option of price sensitivity being checked. The registry will use a high price sensitivity factor for computing the QoS of providers. The information of a provider, which has the highest QoS value of this search, will be returned to this customer.

4.3.3 Scenario 3

When one customer is trying to get a service from the UPS, the UPS wants to check the customer's credit rating. There are many service providers offer credit rating for a fee. In this case, the UPS portal needs to access the registry to request a list of service providers. It dependents on the requirements of the UPS portal management, it might be very price sensitive, service sensitive or balanced with both. The registry will return the requested list according to UPS's constrains using its QoS composition. Then the UPS will select one provider to check its customer's credit. This kind of integration between the

UPS and credit agent is short term. At the end, the UPS portal will use the pair of keys to update the QoS of the provider with the registry. In this scenario, service like credit rating doesn't require a long term relationship such as a 3 month leasing contract. So customers are normally more concerned with price, not with service such as penalty rate and compensation rate. A price sensitivity search serves customers well in this case to find the best deal from all listed service providers. This shows the needs for offering sensitivity factors.

4.4 What has Implemented

To demonstrate the QoS model and conduct experiments on it, we implemented one QoS registry, one UPS portal, two telecom service providers and two credit rating agencies. In the following sections, we will show details of those implementations.

4.5 Choice of Design

The BEA Weblogic Workshop 8.1 is used to develop all the systems in our implementation. The reason for us to choose this tool is that we can develop a 3 tiers application in Java very easily. Other reasons are those: the Workshop 8.1 has an advantage to create web services and test them in an integrated environment; it also relieves the burden on the developers to create JSP pages connecting to back-end databases; and Workshop 8.1 provides programmers the ability to manipulate Extensible Markup Language (XML) files.

The J2EE multi-tier distributed application model is adopted in the design of the implementations of QoS registry and UPS portal. Figure 2 shows a typical architecture of such a multi-tier system. This approach separates the client interface, business logic and the back-end database. This architecture has been proved to be efficient to build complicated distributed applications which can survive many years. The client tier handles user interface. It doesn't deal with complex business logics such as querying databases and connecting to legacy systems. The business tier encapsulates all business logics from the client tier. It's also a client to the back-end databases. It supports clients such as web browsers, web services, java application and other computing devices. This tier helps Business-to-Business integration in a very efficient way. Business logics are put in a centralized place which can be modified and maintained uniformly. They can be re-used saving precious development resources. The database tier has the same kind of advantages like the middle tier [13].

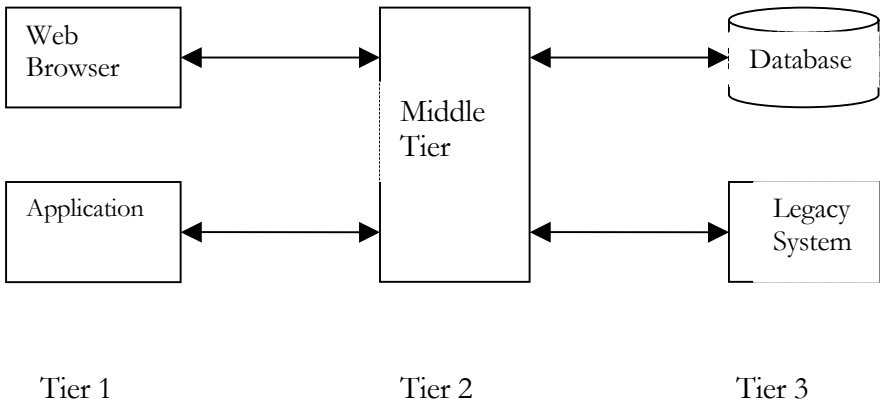


Figure 2 Three-Tier Architecture

Web services are widely used in our implementation. Web services are interfaces which can be accessed across different platforms through XML messaging. A web service represents a set of business logic which can be consumed by various users. Users can consume web services via standard protocol such as HTTP and SOAP etc. The web service interfaces are defined using Web Services Description Language (WSDL) [14].

The future of web services depends on its ability to ease the difficulty of B2B integration. This technology certainly is the right one for the next generation of B2B integration. Moreover, web services can be published through Universal Description Discovery and Integration (UDDI) registry, and be discovered and consumed automatically. In our implementation, all the interfaces which are available to third party are implemented in Web services. We just take the advantage of this new technology to improve our ability to integrate different applications. This gives the clients some interfaces to integrate their system with ours easily.

To improve the ability for B2B integration, all the applications in this implementation are designed to be loosely coupled. Information is exchanged between applications through XML files. A few XML schema files are created for those exchanges. This approach doesn't require the applications to have a large number of interfaces between them.

4.6 Implementation of QoS Registry

The details of QoS registry is shown in Figure 3. It has a set of Web services for providers and end-users to search services, update QoS and check ranking etc. It also has an internal timer to removed expired pairs of ID and password to prevent the overload of the system from used passwords and IDs.

4.6.1 Interfaces of QoS Registry

1. String getServices(String type, int num, int ps, int ss)

This web service interface takes six parameters and returns an XML file with a list of providers. The first parameter is the service type. The service type must be pre-defined and agreed by each participant in the domain. The second parameter is provider number which means returning the requested number of providers. The third parameter is about the price sensitivity which means the degree of the price sensitivity of this search. A high value of price sensitivity search will return a list of providers which has a very low price. The fourth parameter is about the service sensitivity which means the degree of the service sensitivity. A high value of service sensitivity search will return a list of providers which has a QoS with high values of service related quality criteria.

2. public boolean updateQoS(String xmlfile)

This web service interface takes an XML file and updates the QoS of a provider. If it succeeds, it will return true, or false if it fails.

3. public boolean updatePrice(double price, String user, String pwd,)

This web service interface allows provider to update it price dynamically.

4. public Boolean updateCompensationRate(double crate, String user, String pwd)

This interface allows providers update its compensation rates.

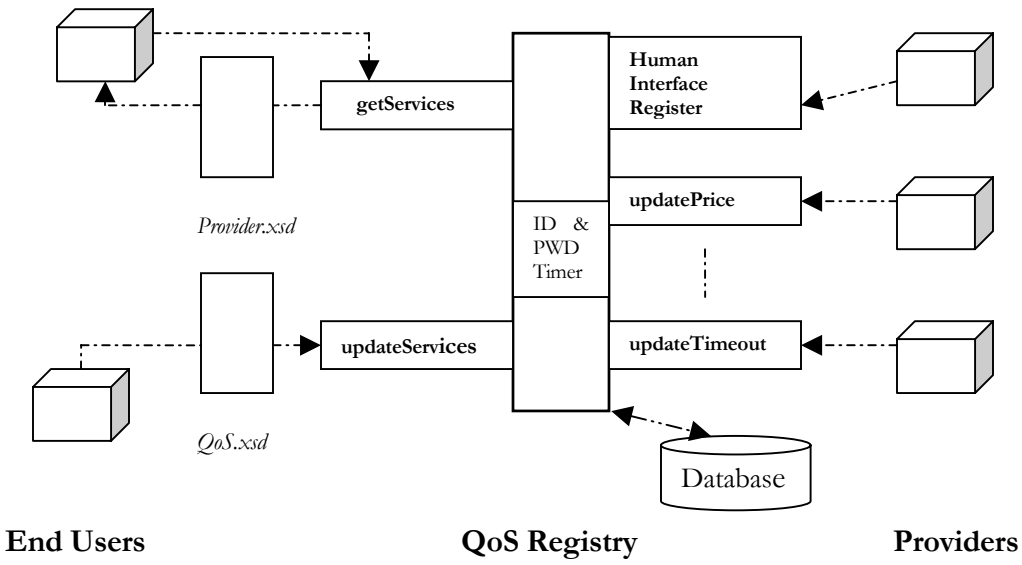


Figure 3 Architectural diagram of QoS Registry

5. public Boolean updatePenaltyRate(double prate, String user, String pwd)

This interface allows providers update its penalty rates.

4.6.2 Programming Code of QoS Registry

All data entries about providers are stored in the database. There is a timer which can remove any expired pair of id and key. It is implemented using a Workshop timer. Details are available in the appendix I.

4.6.3 Graphic Interfaces of QoS Registry

In Figure 4, we can see that the QoS registry has a web page for provider administrator to login to create providers' accounts. Other pages include the login page and the page for updating an account.

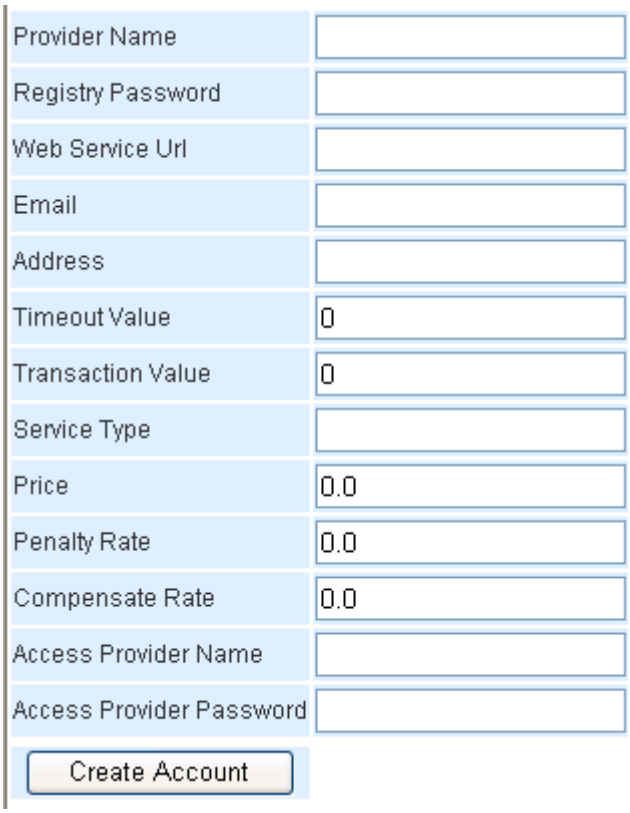


Figure 4 Interface for administrator to create provider account

4.6.4 Schema of QoS Registry

```
<?xml version="1.0"?>
<xs:schema targetNamespace="http://openuri.org/bea/samples/workshop"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ws="http://openuri.org/bea/samples/workshop"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
  <xs:element name="provider-all-data">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="provider" type="ws:ProviderType" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="ProviderType">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="email"    type="xs:string"/>
      <xs:element name="selfpwd"    type="xs:string"/>
      <xs:element name="address"   type="xs:string"/>
      <xs:element name="url"   type="xs:string"/>
      <xs:element name="servicetype"   type="xs:string"/>
      <xs:element name="qos" type="xs:float" use="required" />
      <xs:element name="accessname"   type="xs:string"/>
      <xs:element name="accesspwd"   type="xs:string"/>
      <xs:element name="serviceid"   type="xs:int"/>
      <xs:element name="servicepwd"   type="xs:int"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"/>
  </xs:complexType>
</xs:schema>
```

Figure 5 Provider.xsd

Figure 5 and Figure 6 are two XML schema files which are used to exchange data between the UPS portal and QoS registry. The Provider.xsd is used to construct a message about a service provider with name, address, web service URL etc. It can also be used to parse an XML file built using the same schema file of Provider.xsd. The QoS.xsd is used for service consumer to send feedback to QoS registry about the service it just consumed.

```
<?xml version="1.0"?>
<xs:schema targetNamespace="http://openuri.org/bea/samples/workshop"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ws="http://openuri.org/bea/samples/workshop" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
    <xs:element name="qos-data">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="providername" type="xs:string" use="required" />
          <xs:element name="reputation" type="xs:int" use="required" />
          <xs:element name="time" type="xs:int" use="required" />
          <xs:element name="id" type="xs:int" use="required" />
          <xs:element name="pwd" type="xs:int" use="required" />
        </xs:sequence>
      </xs:complexType>
   </xs:element>

</xs:schema>
```

Figure 6 QoS.xsd

4.7 Implementation of UPS Portal

The UPS portal is designed as a web site offering telecom services such as local phone service and long distance service. It lets its customers to search for a service and order that service later. The portal has many web pages serving its customers and its own administrators. In Figure 7, it shows that UPS portal uses a 3-tier architecture which has a client tier, a business logic tier and back-end database tier.

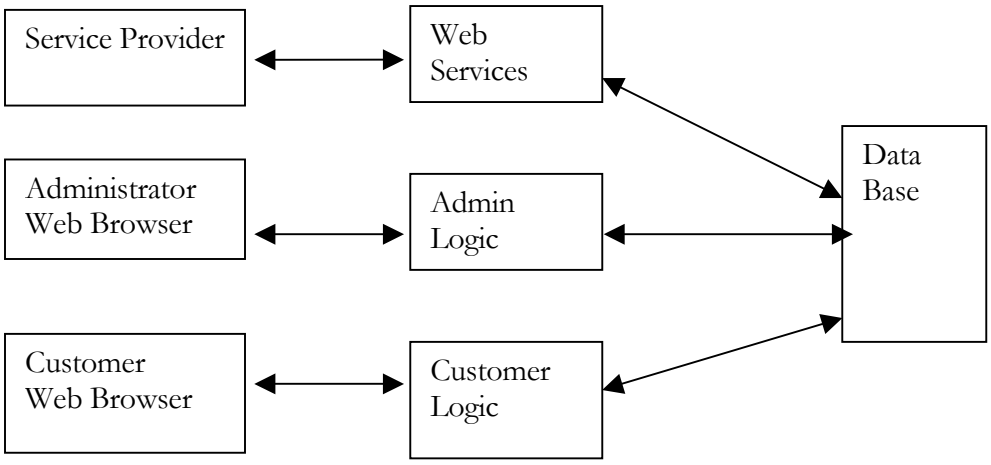


Figure 7 Architecture of UPS Portal

4.7.1 Database of UPS Portal

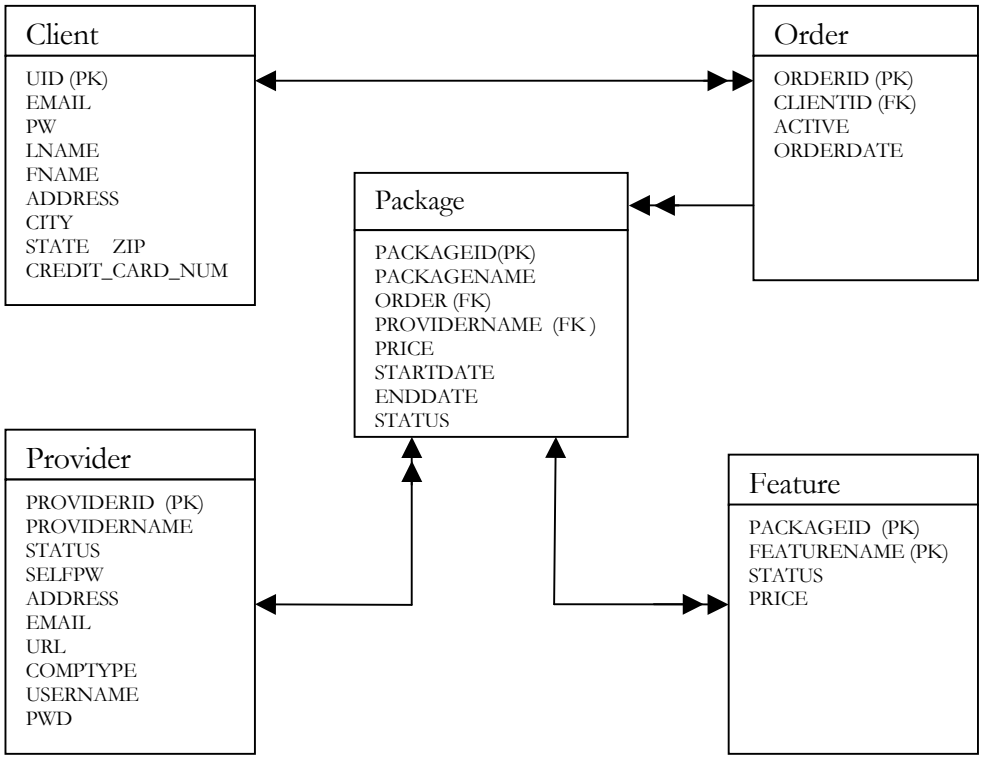


Figure 8 Database Entity Relationships of UPS

4.7.2 Exchange Data between UPS and Providers

UPS portal have one web service interface for provider to remove its service from the UPS portal. Every telecom service provider has three web service interfaces: the first one is “search service” which returns a list of service packages matching users’ query; the second one is “place order”, it lets its client place the order with packages from the results of “search service”; and the third one is “cancel service” which lets its user to cancel ordered services. For security reason, each web service requires an ID and password to invoke. All data exchanged between the UPS and telecom service providers is in the form of XML file. A set of XML schema is available on the UPS portal web site.

4.7.3 Web Pages of UPS Portal

Figure 9 Web page for client to create an account

Please enter the service option to search:

Key Word: Caller ID

Very Service Sensitive ☐

Very Price Sensitive ☐

Search

Figure 10 Web page for client to search service with choice of sensitivities

Universal Phone Service

Home | Logout | Cart | Check Out | Search | Account | Order History | Search Results |

Package	Provider	Features	Cart
Economic Phone	SBC	Detail	Add
Homeline Plus	SBC	Detail	Add
Homeline Select	SBC	Detail	Add
Homeline	SBC	Detail	Add

Figure 11 Result of search service

4.8 Implementation of Telecom Service Providers

There are two identical telecom service providers implemented to demonstrate the QoS model and conduct experiments about the QoS model. The two providers are registered with the QoS registry, so end-user like UPS portal can search the QoS registry to find the service provider and access its service dynamically.

4.8.1 Programming of Telecom Service Providers

The web services of telecom service providers include four interfaces: “Search Service”, “Cancel Service”, “Place Order” and “Test”. The provider also has a set of web pages to let administrator to login and update its service including changing price, adding new packages and features or removing them.

4.8.2 JSP Web Pages of Telecom Service Providers

List of All Packages

Package ID	Package Name	Price	Remove
1	Economic Phone	19.99	Remove
2	Middle Level	29.99	Remove
3	Deluxe Package	39.99	Remove
4	Homeline Plus	39.99	Remove
5	Homeline Select	39.99	Remove
6	Homeline	9.99	Remove

Figure 12 List of all packages from one provider

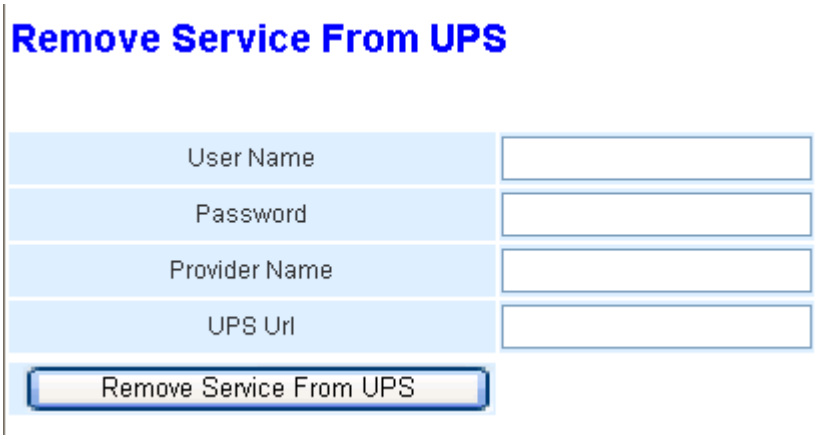


Figure 13 Web page for provider to remove its service from the UPS portal

4.8.3 Database of Telecom Service Providers

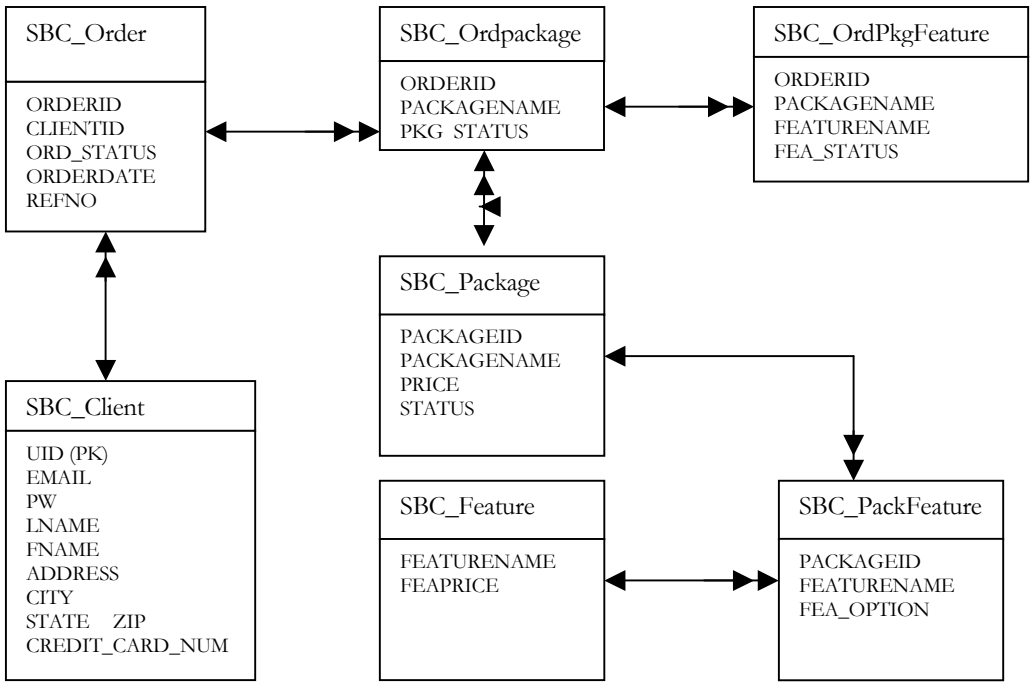


Figure 14 Database Entity Relationship of Service Provider

4.9 Implementation of Credit Agent

Two identical credit rating agents are created in our implementation. The credit agent only has one web service interface: “check credit”. This interface returns an integer number 1 indicating good credit and 0 for a bad credit. For any query about one client, it simply generates a random number between 0 and 1.

The credit agents are registered with the QoS registry. So end-user such as UPS can search QoS registry to find a credit agent and access that credit agent to get customer’s credit rating dynamically.

CHAPTER V

EXPERIMENTS AND ANALYSIS

We conducted a series of experiments to: 1) investigate the relationship between QoS value and the business criteria, 2) study the effectiveness of price and the service sensitivity factors in our QoS computation. The experiments are conducted on a Pentium computer with a 750 MHz CPU and 640MB RAM. We first simulated 600 users searching for services, consuming the services and providing feedbacks to update the QoS registry in the UPS application. This will generate a set of test data for those non-deterministic quality criteria such as reputation and execution duration for each service provider in the QoS registry. The deterministic quality criteria (price, penalty rate) have been advertised and stored in the QoS registry for each service provider prior to the simulation.

One difficulty we have to overcome is that QoS modeling in web service is a very new concept, therefore, we lack of other QoS models in web sevice to be compared with our model by conducting experiments. However, we have carefully selected what should be studied very closely in our experiments. The experiments on the QoS model help us to deeply understand the proposed QoS model and give us ideas about how to improve the QoS model in the future.

5.1 Comparing QoS Data of Two Providers

Providers	Price	Tran	Timeout	Compensation Rate	Penalty Rate	Execution Duration	Reputation
ABC	25	Yes	60	0.5	0.5	100	2.0
BTT	40	Yes	200	0.8	0.1	40	2.5

Table 1: Comparing Service Quality Data of Two Providers

Table 1 shows the values of various quality criteria from two phone service providers with respect to local phone service. From this table, we can see that provider ABC has a better price but the various criteria that are related to services are not very good. Its time out value is small; this means it is not so flexible for end-users. Its compensation rate is lower than BTT, and its penalty rate is higher than BTT. Its execution time is longer than BTT and reputation is lower than BTT as well. Using our QoS computation algorithm, can we ensure that ABC will win for a price sensitive search or BTT will win for a service sensitive search?

Providers	Price Sensitivity	Service Sensitivity	QoS Value
ABC	1	2	4.675
BTT	1	2	5.437
ABC	2	1	4.281
BTT	2	1	3.938

Table 2: Results of Sensitivity Experiments

Table 2 shows that BTT has a QoS value of 3.938 with a price sensitivity search and a QoS value of 5.437 with a service sensitivity search. On the other hand, ABC has a QoS value of 4.281 with a price sensitivity search and a QoS value of 3.938 with a service sensitivity search. ABC is winning for a price sensitive search (4.281 > 3.938), BTT is winning for a service sensitive search (5.437 > 4.281) which verifies our original hypothesis.

5.2 Relationships between QoS and Quality Criteria

The following three figures show the relationships between the QoS value and the price, the compensation rate, the penalty rate, and the sensitivities factors. From Figure 15, we can see that the QoS increases exponentially as the price approaches zero. As the price reaches 9x20 = 180$, the QoS becomes very stable. This gives us the confidence that price can be used very effectively to increase QoS within certain range. It also shows that in our QoS computation model, QoS cannot be dominated by price component indefinitely.

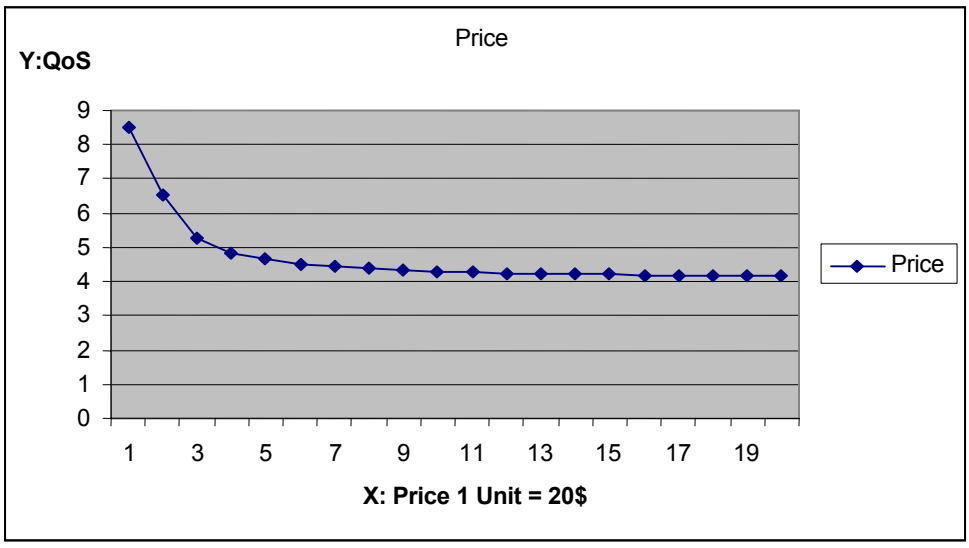


Figure 15: Relationship between QoS and Price

Figure 16 shows the relationships between QoS and services rates. For the penalty, it shows that the QoS value decreases as the penalty rate increases. For the compensation, it shows that the QoS value increases as the compensation rate increases. The limited range of QoS in Figure 16 shows that QoS computation cannot be dominated by these two components indefinitely.

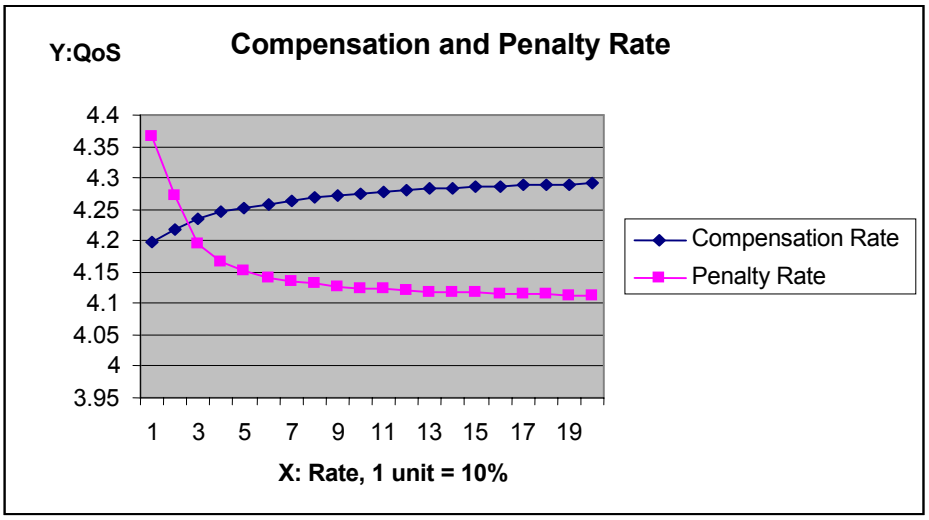


Figure 16: Relationship between QoS, Compensation and Penalty Rate

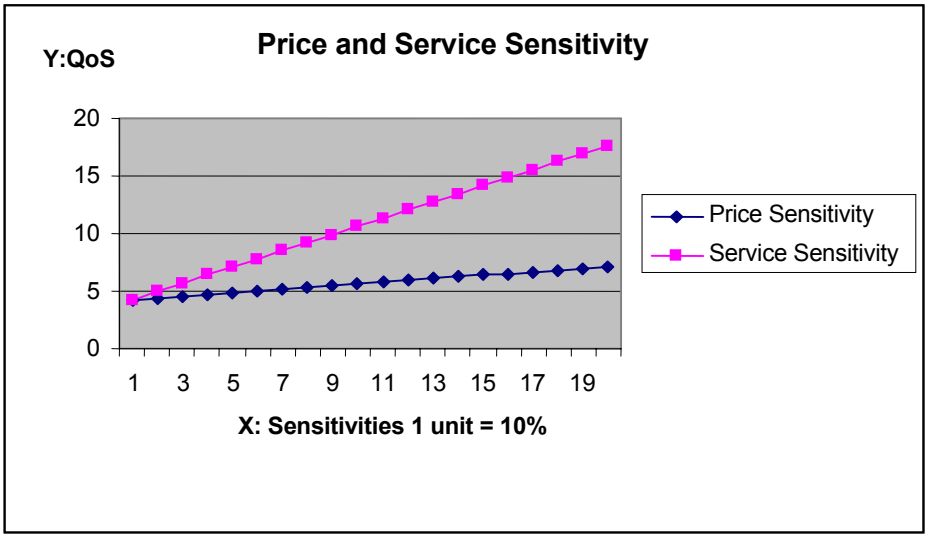


Figure 17: Relationship between QoS, Price and Service Sensitivity Factor

Figure 17 indicates that QoS value increases almost four times faster for the service sensitivity than the price sensitivity. This shows that using the same sensitivity value for both price and service factors will not be effective for the price sensitivity search in our QoS registry. To get an effective price or service sensitivity factor for a domain, we need to analyze the sensitivity factors after the QoS registry has been used for a while and re-adjust the values by performing experiment mentioned in Figure 4.

5.3 Simulation in Experiments

To conduct our experiments effectively in such a short time, we need a method for us to collect a large amount of data automatically. It will be impossible and un-necessary for us to manually access the web pages of UPS, search service and place order with it to collect such amount of data. The solution to this problem is implementing a component in the UPS to simulate many users to do the search, access QoS registry, place orders with service providers and update QoS information with the registry. This component can be accessed through a web interface, which can let administrator to set the number of human users in the simulation.

The implementation of this simulation is very straight. First, it accesses the web service of QoS registry to search for a telecom service provider. Once it gets the XML file from the registry, it parses the file to extract all the information about the provider; this should include a pair of ID and password keys for updating the QoS of the service provider later. Second, the simulation

component uses the information about the provider to access service provider's web service to search for a service with "caller ID" feature. Once it gets the XML file with the search results from the provider, it parses the XML file to construct a Java class. Third, the simulation component will place an order with the service provider using the parsed data in the Java class. Only the execution duration of placing an order with the service provider is recorded. Fourth, once the order has been placed and the duration time becomes available, the simulation component will construct an XML file with all the QoS information about this service provider, and this XML file will be sent to the QoS registry through the registry's web service interface. The user's feedback about the reputation of this service provider is generated using a Java random number generator. Inside the XML file, a pair of ID and password keys which obtained from the previous search results from the QoS registry is included.

CHAPTER VI

CONCLUSION

The quality-based selection of web services is an active research topic in the dynamic composition of services. The main drawback of current work in dynamic web service selection is the inability to ensure that the QoS of published web services remain open, trustworthy and fair. We have presented an extensible QoS model that is open, fair and dynamic for both service requesters and service providers. We achieved the dynamic and fair computation of QoS values of web services through secure active users' feedback and active monitoring. Our QoS model concentrates on criteria that can be collected and enforced objectively. In particular, we showed how business related criteria (compensation, penalty policies and transaction) can be measured and included in QoS computation. Our QoS model is extensible and thus new domain specific criteria can be added without changing the underlying computation model. We provided an implementation of a QoS registry based on our extensible QoS model in the context of a hypothetical phone service provisioning market place. We also conducted experiments which validate the formula we used in the computation of QoS values in the phone service provisioning domain.

The formula for computing the QoS values can be varied for different domains by using different sensitivity factors or different criteria with its

associated groupings. Additional factors, different groupings, or criteria might serve certain domains better. We provided a mechanism for service providers to query their QoS computed by the registry, and update their published services to become more competitive at anytime.

The success of this model depends on the willingness of end-users to give feedback on the quality of the services that they consume. Our future work includes ways to automate the collection of feedback data. To ensure that all users will provide feedback, we need to make providing feedback very straight forward. We also want to see that the implemented QoS registry becomes intelligent internally and externally. Internal intelligence refers to the ability to infer the appropriate sensitivity factors to use to get the list of providers which meets end-users requirements. External intelligence refers to the ability of QoS registry to be able to predict the trend of QoS from certain providers at runtime. For example, we want to have a system which knows how much the QoS value decreases in the morning and increases in the evening or during different time of the year, and takes action to notify its users. For the service provider, we want to have the ability to notify the provider when the QoS of its service is below a certain threshold.

APPENDICES

APPENDIX I

PROGRAMMING SOURCE CODE OF QOS REGISTRY

The following material is the Java source code of the implementation of QoS registry which is created in BEA Weblogic Workshop.

1. AccessWebService.jws
2. RegDatabaseControl.jcx
3. provider.java
4. RegJavaControl.jcs
5. RegistryWebFlowController.jpf
6. login.jsp
7. updateAccount.jsp

1. AccessWebService.jws

```
package AccessWebServiceFile;

import weblogic.jws.control.JwsContext;
import org.openuri.bea.samples.workshop.ProviderbAllData;
import org.openuri.bea.samples.workshop.ProviderbType;
import JavaClasses.provider;
import java.lang.Math;
import java.util.Calendar;
import java.util.Random;
import JavaClasses.providerb;
import org.openuri.bea.samples.workshop.QosData;
import com.bea.wlw.netui.util.XMLString;
import org.openuri.bea.samples.workshop.ProviderbAllData;
import org.openuri.bea.samples.workshop.ProviderbType;

public class AccessWebServices
{
   /**
    * @common:control
    * @jc:timer repeats-every="600 minutes" timeout="600 minutes"
    */
   private weblogic.jws.control.TimerControl removeOldKeyTimer;

   /**
    * @common:control
    */
   private AccessWebServiceFile.RegDatabaseControl RegDatabaseControl;

   /** @common:context */
   JwsContext context;

   /**
    * @common:operation
    */
   public  String  getServices(String serviceType, int ProviderNumber,
                        int pricesensitive, int servicesensitive)
   {
      ProviderbAllData ResultDataXML;
      ProviderbType provider;

      try
      {
        ResultDataXML = ProviderbAllData.Factory.newInstance();
        provider[] providerArrayA =
              RegDatabaseControl.getAllActiveProvider(serviceType);

         // catch some exceptions
          if( ProviderNumber <= 0 || providerArrayA == null ||
                providerArrayA.length == 0 )
           return null;
         if( pricesensitive <0 || pricesensitive >2 ||
                servicesensitive < 0
            || servicesensitive > 2)
           return null;

         providerb[] providerArray = new
                        providerb[providerArrayA.length];

         //calculate Qos needed average price, time, reputation
```

```
double totalPrice = 0;
double totalTime = 0;
double totalUsability = 0;
double totalReputation = 0;
long   totalTimeout = 0;
double totalCompensaterate = 0;
double totalPenaltyrate = 0;
int    totalTransactionProvider = 0;

for( int j=0; j < providerArrayA.length; j++)
{
    totalPrice += providerArrayA[j].getPrice();
    totalTime += providerArrayA[j].getAvgexetime();
    totalReputation += providerArrayA[j].getAprap();
    if( providerArrayA[j].getTransaction() == 5 )
    {
        totalTransactionProvider++;
        totalTimeout += providerArrayA[j].getTimeout();
    }
    totalCompensaterate +=
            providerArrayA[j].getCompensaterate();
    totalPenaltyrate += providerArrayA[j].getPenaltyrate();
}

// calculate average data to calculate usability
double avgTimeout = (double)
          totalTimeout/totalTransactionProvider;
double avgCompensaterate =
           totalCompensaterate/providerArrayA.length;
double avgPenaltyrate =
            totalPenaltyrate/providerArrayA.length;

for( int j=0; j < providerArrayA.length; j++)
{
    if( providerArrayA[j].getTransaction() == 5 )
    {
       double TimeFraction;
       if( avgTimeout != 0 )
       {
          TimeFraction =
            providerArrayA[j].getTimeout()/avgTimeout;
       }
       else
       {
         TimeFraction = 2.5;
       }
       if( TimeFraction > 5 )
            TimeFraction = 5;

       double CompensaterateFraction;
       if( avgCompensaterate != 0 )
       {
         CompensaterateFraction =
  providerArrayA[j].getCompensaterate()/
                        avgCompensaterate;
       }
       else
         CompensaterateFraction = 2.5;

       if( CompensaterateFraction > 5 )
          CompensaterateFraction = 5;
```

```
        double PenaltyrateFraction;
        if( avgPenaltyrate != 0 )
        {
           PenaltyrateFraction = avgPenaltyrate/
                  providerArrayA[j].getPenaltyrate();
        }
        else
           PenaltyrateFraction = 5;

        if( PenaltyrateFraction > 5)
           PenaltyrateFraction = 5;

        totalUsability +=
              2.5+TimeFraction+CompensaterateFraction
                    +PenaltyrateFraction;
      }
      else
      {

        double CompensaterateFraction;
        if( avgCompensaterate != 0 )
        {
          CompensaterateFraction =
              providerArrayA[j].getCompensaterate()/
                           avgCompensaterate;
        }
        else
          CompensaterateFraction = 2.5;

        if( CompensaterateFraction > 5 )
           CompensaterateFraction = 5;

        double PenaltyrateFraction;
        if( avgPenaltyrate != 0 )
        {
           PenaltyrateFraction = avgPenaltyrate/
                 providerArrayA[j].getPenaltyrate();
        }
        else
           PenaltyrateFraction = 0;

        if( PenaltyrateFraction > 5)
           PenaltyrateFraction = 5;

        totalUsability +=
              CompensaterateFraction+PenaltyrateFraction;
      }
   }

//calculate average price, time, usability, reputation and
//evaluation values
double avgPrice =totalPrice/providerArrayA.length;
double avgTime =totalTime/providerArrayA.length;
double avgUsability =totalUsability/providerArrayA.length;
double avgReputation =totalReputation/providerArrayA.length;

//create a new array of providerb type with
//calculated QoS values
for( int i=0; i<providerArrayA.length; i++)
{
   providerArray[i] = new providerb();
   providerArray[i].setProvidername(providerArrayA[i].
```

```
                    getProvidername());
providerArray[i].setEmail(providerArrayA[i].getEmail());
providerArray[i].setSelfpw(providerArrayA[i].getSelfpw());
providerArray[i].setAddress(providerArrayA[i].getAddress());
providerArray[i].setUrl(providerArrayA[i].getUrl());
providerArray[i].setServicetype(providerArrayA[i].getServicetype());
providerArray[i].setAccessproviderpw(providerArrayA[i].getAccessproviderpw());
 providerArray[i].setAccessproviderusername(providerArrayA[i].getAccessproviderusername());

// calculate the price fraction
double PRICELIMIT = 5;
double p = 0;
if( providerArrayA[i].getPrice() == 0 )
  p = PRICELIMIT;
else
{
   if( avgPrice/providerArrayA[i].getPrice() > PRICELIMIT )
      p = PRICELIMIT;
   else
      p = avgPrice/providerArrayA[i].getPrice();
}

// calculate the time fration
double TIMELIMIT = 5;
double t = 0;
if( avgTime == 0 || providerArrayA[i].getAvgexetime() == 0 )
  t = TIMELIMIT / 5;
else
{
  t = avgTime/providerArrayA[i].getAvgexetime();
  if( t > TIMELIMIT )
    t = TIMELIMIT;
}

//calculate usability
double u = 0;
if( avgUsability == 0 )
 u = 0;
else
{
 double oneProviderUsability;

 if( providerArrayA[i].getTransaction() == 5 )
 {
   double TimeFraction;
   if( avgTimeout != 0 )
   {
     TimeFraction =  providerArrayA[i].getTimeout()/avgTimeout;
   }
   else
   {
     TimeFraction = 2.5;
   }
   if( TimeFraction > 5 )
        TimeFraction = 5;

   double CompensaterateFraction;
   if( avgCompensaterate != 0 )
   {
     CompensaterateFraction = providerArrayA[i].getCompensaterate()/
                              avgCompensaterate;
   }
```

```
    else
      CompensaterateFraction = 2.5;

    if( CompensaterateFraction > 5 )
       CompensaterateFraction = 5;

    double PenaltyrateFraction;
    if( providerArrayA[i].getPenaltyrate() != 0 )
    {
       PenaltyrateFraction = avgPenaltyrate/
                        providerArrayA[i].getPenaltyrate();
    }
    else
       PenaltyrateFraction = 5;

    if( PenaltyrateFraction > 5)
       PenaltyrateFraction = 5;

    oneProviderUsability = 2.5+TimeFraction+CompensaterateFraction
                +PenaltyrateFraction;
  }
  else
  {

    double CompensaterateFraction;
    if( avgCompensaterate != 0 )
    {
      CompensaterateFraction = providerArrayA[i].getCompensaterate()/
                         avgCompensaterate;
    }
    else
      CompensaterateFraction = 2.5;

    if( CompensaterateFraction > 5 )
       CompensaterateFraction = 5;

    double PenaltyrateFraction;
    if( providerArrayA[i].getPenaltyrate() != 0 )
    {
       PenaltyrateFraction = avgPenaltyrate/
                        providerArrayA[i].getPenaltyrate();
    }
    else
       PenaltyrateFraction = 5;

    if( PenaltyrateFraction > 5)
       PenaltyrateFraction = 5;

    oneProviderUsability = CompensaterateFraction+PenaltyrateFraction;
  }
  u = oneProviderUsability/avgUsability;
}
//calculate evaluation
double e = 0;
if( avgReputation == 0 ||
    providerArrayA[i].getAprap() == 0 )
    e = 0;
else
{
    e = providerArrayA[i].getAprap()/avgReputation ;
}
```

```
    //set the p, t, u and e to their up limit of 5
    if( p >5 )
      p =5;
    if( t >5 )
      t =5;
    if( e >5 )
      e =5;
     if( u >5 )
      u =5;
    double qos = pricesensitive*p + servicesensitive*t
                + servicesensitive*u + servicesensitive*e;

    providerArray[i].setQos(qos);

}

//the rest of code will find those provider with QoS values
//which are the top ProviderNumber
float MARKIT = 1000;  //used mark those QoS values are the tops
boolean moreprovider = false;

if( ProviderNumber < providerArray.length )
{
  moreprovider = true;
  for( int p=0; p<ProviderNumber; p++)
  {
    double highest = 0;
    int index = 0;
    boolean found = false;
    for( int i=0; i < providerArray.length; i++)
    {
      double qos = providerArray[i].getQos();
      if( providerArray[i].getQos() < MARKIT & providerArray[i].getQos() > highest )
      {
          highest =  providerArray[i].getQos();
          found = true;
          index = i;
      }
    }
    if( found == true ) //mark the select provider
       providerArray[index].setQos(providerArray[index].getQos()+MARKIT);
  }
}

//add all the packages
for( int j=0; j < providerArray.length; j++)
{

   if( moreprovider==true && providerArray[j].getQos() >MARKIT )
   {
     ProviderbType prov = ResultDataXML.addNewProviderb();
     prov.setName(providerArray[j].getProvidername());
     prov.setEmail(providerArray[j].getEmail());
     prov.setSelfpwd(providerArray[j].getSelfpw());
     prov.setAddress(providerArray[j].getAddress());
     prov.setUrl(providerArray[j].getUrl());
     prov.setServicetype(providerArray[j].getServicetype());
     prov.setQos((float) providerArray[j].getQos() - MARKIT );
     prov.setAccessname(providerArray[j].getAccessproviderusername());
     prov.setAccesspwd(providerArray[j].getAccessproviderpw());
     prov.setId(providerArray[j].getProvidername());
```

```
            // setup the id and pwd for updating QoS
            int id, pwd;
            Random RandomGen = new Random();
            pwd = RandomGen.nextInt();

            Calendar rightNow = Calendar.getInstance();
            long ctime = rightNow.getTimeInMillis();
            int  ct = (int) (ctime /( 1000*60 ));
            RegDatabaseControl.addPairKeys(pwd,providerArray[j].getProvidername(), ct);
            id = RegDatabaseControl.getNewKeyId();
            prov.setServiceid(id);
            prov.setServicepwd(pwd);
          }
          else  if( moreprovider == false )
          {
            ProviderbType prov = ResultDataXML.addNewProviderb();
            prov.setName(providerArray[j].getProvidername());
            prov.setEmail(providerArray[j].getEmail());
            prov.setSelfpwd(providerArray[j].getSelfpw());
            prov.setAddress(providerArray[j].getAddress());
            prov.setUrl(providerArray[j].getUrl());
            prov.setServicetype(providerArray[j].getServicetype());
            prov.setQos( (float) providerArray[j].getQos() );
            prov.setAccessname(providerArray[j].getAccessproviderusername());
            prov.setAccesspwd(providerArray[j].getAccessproviderpw());
            prov.setId(providerArray[j].getProvidername());

            // setup the id and pwd for updating QoS
            int id, pwd;
            Random RandomGen = new Random();
            pwd = RandomGen.nextInt();
            //get the current time
            Calendar rightNow = Calendar.getInstance();
            long ctime = rightNow.getTimeInMillis();
            int  ct = (int) (ctime /( 1000*60 ));
            RegDatabaseControl.addPairKeys( pwd, providerArray[j].getProvidername(), ct);
            id = RegDatabaseControl.getNewKeyId();
            prov.setServiceid(id);
            prov.setServicepwd(pwd);
          }
        }
        return ResultDataXML.xmlText();

    }catch( Exception e){ return null;}
}

/**
 * @common:operation
 */
public boolean updateQoS(String xmlfile)
{
    try{
      /*create the test xmlfile*/
      /*QosData ResultDataXML;
      ResultDataXML = QosData.Factory.newInstance();

      String prvname = "CITI";
      ResultDataXML.setProvidername(prvname);
      ResultDataXML.setReputation(3);
      ResultDataXML.setTime(9729929);
      ResultDataXML.setId(3);
      ResultDataXML.setPwd(-2039625279);
```

```
      // read data from the xml file
      QosData QosInfo = QosData.Factory.parse( ResultDataXML.xmlText());
      */
      QosData QosInfo = QosData.Factory.parse(xmlfile);
      String providername = QosInfo.getProvidername();
      int reputation = QosInfo.getReputation();
      int etime = QosInfo.getTime();
      int id = QosInfo.getId();
      int pwd = QosInfo.getPwd();

      java.sql.ResultSet keypair = RegDatabaseControl.getPairKeys(id);
      if( keypair == null )
        return false;

       while( keypair.next() )
       {
          int idIntable = keypair.getInt(1);
          int pwdIntable = keypair.getInt(2);
          String pvdnameIntable = keypair.getString(3);
          if( idIntable == id && pwdIntable == pwd &&
          providername.equals(pvdnameIntable) )
          {
            //password, id and name matched, update qos
            int pvdId = RegDatabaseControl.findProviderId(pvdnameIntable);
            provider pvd = RegDatabaseControl.getProvider(pvdId);

            double newReputation = (reputation+pvd.getTotalaccessed()*pvd.getAprap())
                          /(1+pvd.getTotalaccessed());
            double newAvgexetime = (etime+pvd.getAvgexetime()*pvd.getTotalaccessed())
                          /(1+pvd.getTotalaccessed());
            RegDatabaseControl.updateProvider(newReputation,newAvgexetime,
                                  pvd.getTotalaccessed()+1,pvdId);
            RegDatabaseControl.removePairKeys(id);
            return true;
          }
          else
             return false;
       }

      return false;
    }
    catch( Exception e )
    {
      return false;
    }
}

public void removeOldKeyTimer_onTimeout(long arg0)
{
    Calendar rightNow = Calendar.getInstance();
    long ctime = rightNow.getTimeInMillis();
    int  ct = (int) (ctime /( 1000*60 ));
    int  ctToRemove = ct - 600;
    int[] keyidArray = RegDatabaseControl.findTimeoutKeys(ctToRemove);

    //remove timeout key pairs, 10 hours ago
    for( int i=0; i<keyidArray.length; i++)
    {
       RegDatabaseControl.removePairKeys(keyidArray[i]);
    }
}
```

```
}
```

2.RegDatabaseControl.jcx

```
package AccessWebServiceFile;

import weblogic.jws.control.*;
import java.sql.SQLException;
import JavaClasses.*;

/**
 * Defines a new database control.
 *
 * The @jc:connection tag indicates which WebLogic data source will be used by
 * this database control. Please change this to suit your needs. You can see a
 * list of available data sources by going to the WebLogic console in a browser
 * (typically http://localhost:7001/console) and clicking Services, JDBC,
 * Data Sources.
 *
 * @jc:connection data-source-jndi-name="UPSDataSource"
 */
public interface RegDatabaseControl extends DatabaseControl
{
    // Sample database function.  Uncomment to use

    // static public class Customer
    // {
    //   public int id;
    //   public String name;
    // }
    //
    // /**
    //  * @jc:sql statement="SELECT ID, NAME FROM CUSTOMERS WHERE ID = {id}"
    //  */
    // Customer findCustomer(int id);

    // Add "throws SQLException" to request that SQLExeptions be thrown on errors.

    /**
     * @jc:sql statement::
     * SELECT * FROM REGISTRYPROVIDER
     * WHERE SERVICETYPE={servicetype}::
     */
    provider[] getAllProvider(String servicetype);

   /**
     * @jc:sql statement::
     * SELECT * FROM REGISTRYPROVIDER WHERE STATUS='active'
     * AND SERVICETYPE={servicetype}::
     */
    provider[] getAllActiveProvider(String servicetype);

    /**
     * @jc:sql statement::
     * INSERT INTO REGISTRYPROVIDER( PROVIDERNAME, STATUS,
     * SELFPW, ADDRESS, EMAIL, URL, SERVICETYPE, TOTALACCESSED,
     * PRICE, AVGEXETIME,  TRANSACTION,APRAP,TIMEOUT,
     * COMPENSATERATE, PENALTYRATE,
     * ACCESSPROVIDERUSERNAME, ACCESSPROVIDERPW)
     * VALUES({providername}, {status},
     * {selfpw}, {address}, {email},{url},
     * {servicetype},{totalaccessed},{price},
```

```
 * {avgexetime},{transaction},{aprap},
 * {timeout},{compensaterate},{penaltyrate},
 * {accessproviderusername},
 * {accessproviderpw})::

 */
void addProvider(String providername,String status, String selfpw ,
    String address,String email, String url, String servicetype,
    int totalaccessed, double price, double avgexetime,
    int transaction, double aprap, int timeout,
    double compensaterate, double penaltyrate,
    String accessproviderusername, String accessproviderpw );


/**
 * @jc:sql statement::
 * INSERT INTO REGISTRYKEYS(PASSWORD,PROVIDERNAME, WHENENTERED)
 * VALUES({password},{providername},{ctime})::

 */
void addPairKeys(int password, String providername, int ctime);

/**
 * @jc:sql statement::
 * SELECT * FROM REGISTRYKEYS
 * WHERE KEYID={keyid}::

 */
java.sql.ResultSet getPairKeys(int keyid);

/**
 * @jc:sql statement::
 * DELETE FROM REGISTRYKEYS
 * WHERE keyid={keyid}::

 */
void removePairKeys(int keyid);

/**
 * @jc:sql statement::
 * UPDATE REGISTRYPROVIDER SET STATUS={'unactive'}
 * WHERE PROVIDERID={providerid}::

 */
void removeProvider(int providerid);

/**
 * @jc:sql statement::
 * SELECT * FROM REGISTRYPROVIDER
 * WHERE PROVIDERID={providerid}::

 */
provider getProvider(int providerid);

/**
 * @jc:sql statement="SELECT MAX(KEYID) FROM REGISTRYKEYS"
 */
int getNewKeyId();

/**
 * @jc:sql statement::
 * SELECT PROVIDERID FROM REGISTRYPROVIDER
```

```
     * WHERE PROVIDERNAME={name}::

     */
    int findProviderId(String name);

    /**
     * @jc:sql statement::
     * UPDATE REGISTRYPROVIDER SET
     * APRAP={naprap},
     * AVGEXETIME={navgexetime},
     * TOTALACCESSED={ntotalaccessed}
     * WHERE PROVIDERID={pid}::

     */
    void updateProvider(double naprap, double navgexetime, int ntotalaccessed, int pid);

    /**
     * @jc:sql statement::
     * SELECT KEYID FROM REGISTRYKEYS
     * WHERE WHENENTERED < timeOut::

     */
    int[] findTimeoutKeys(int timeOut);

    /**
     * @jc:sql statement::
     * SELECT PROVIDERID FROM REGISTRYPROVIDER WHERE
     * PROVIDERNAME={providername} AND SELFPW={selfpw}::

     */
    int matchProviderPassword(String providername, String selfpw);
}
```

3. provider.java

```
package JavaClasses;

import java.io.Serializable;

public class provider implements Serializable
{

public int providerid;
public String providername;
public String status;
public String selfpw;
public String address;
public String email;
public String url;
public String servicetype;
public double price;
public double avgexetime;
public int transaction;
public double aprap;
public int timeout;
public double compensaterate;
public double penaltyrate;
public int totalaccessed;
public String accessproviderusername;
public String accessproviderpw;
```

```
public provider() {}

public provider(String pname, String pstatus,String pselfpw,
String padd,String pemail,String purl,String pservicetype,
double pprice, double pavgexetime, int ptransaction,double paprap,
int ptimeout, double pcompensaterate, double ppenaltyrate,
int ptotalaccessed, String paname,String papw)
{

   providername=pname;
   status=pstatus;
   selfpw=pselfpw;
   address=padd;
   email=pemail;
   url=purl;
   servicetype=pservicetype;
   price = pprice;
   avgexetime = pavgexetime;

   aprap = paprap;
   transaction = ptransaction;
   timeout = ptimeout;
   compensaterate = pcompensaterate;
   penaltyrate = ppenaltyrate;
   totalaccessed = ptotalaccessed;
   accessproviderusername=paname;
   accessproviderpw=papw;
}

   public int getProviderid() { return providerid;}
   public String getProvidername() { return providername;}
public String getStatus() { return status;}
public String getSelfpw() { return selfpw;}
public String getAddress() { return address;}
public String getEmail() { return email;}
public String getUrl() { return url;}
public String getServicetype() { return servicetype;}
public String getAccessproviderusername() { return accessproviderusername;}
public String getAccessproviderpw() { return accessproviderpw;}
public double getPrice() {return price; }
public double getAvgexetime() {return avgexetime; }
public double getAprap() { return aprap; }
public int getTransaction() { return transaction;}
public int getTimeout(){return timeout;}
public double getCompensaterate(){ return compensaterate;}
public double getPenaltyrate(){ return penaltyrate;}
public int getTotalaccessed() {return totalaccessed; }

public void setProviderid(int i) { providerid=i;}
public void setProvidername(String n) {  providername=n;}
public void setStatus(String s) {  status=s;}
public void setSelfpw(String self) {  selfpw=self;}
public void setAddress(String a) { address=a;}
public void setEmail(String e) { email=e;}
public void setUrl(String u) { url=u;}
public void setServicetype(String d) { servicetype=d;}
public void setTotalaccessed( int ta) {totalaccessed=ta; }
public void setPrice( double ap ) { price = ap; }
public void setAvgexetime(double ap ) { avgexetime=ap; }
public void setAprap(double ap) { aprap=ap; }
public void setTransaction(int pt) { transaction=pt;}
```

```
public void setTimeout(int pt){ timeout=pt;}
public void setCompensaterate(double cr){ compensaterate=cr;}
public void setPenaltyrate(double pr){ penaltyrate=pr;}
public void setAccessproviderusername(String n) { accessproviderusername=n;}
public void setAccessproviderpw(String p) { accessproviderpw=p;}

}
```

4.RegJavaControl.jcs

```
package resources;

import weblogic.jws.util.Logger;
import weblogic.jws.control.JbcContext;
import JavaClasses.provider;

/**
* @jcs:jc-jar label="RegJavaControl"
*/
public class RegJavaControl
{
    /**
     * @common:control
     */
    private resources.RegDatabaseControl1 regDatabaseControl1;

    /**
     * @common:context
     */
    JbcContext context;

    /**
     * @common:operation
     */
  public boolean createAccount(String providername,String selfpw ,
        String address,String email, String url, String servicetype,
        double price,int transaction, int timeout,
        double compensaterate, double penaltyrate,
        String accessproviderusername, String accessproviderpw
    )
    {
        int pvdid = regDatabaseControl1.findProviderId(providername);
        if( pvdid !=0 ) // the name is not their
          return false;

        regDatabaseControl1.addProvider(providername,"active", selfpw ,
        address, email, url, servicetype,
        0, price, 0,transaction, 0, timeout,
        compensaterate, penaltyrate,
        accessproviderusername, accessproviderpw  );

        return true;
    }


    /**
     * @common:operation
     */
    public int login(String user, String pwd)
    {
       return regDatabaseControl1.matchProviderPassword(user, pwd);
```

```
    }

    /**
     * @common:operation
     */
    public void updateAccount(String providername, String status, String selfpw ,
        String address,String email, String url, String servicetype,
        double price, int transaction, int timeout,
        double compensaterate, double penaltyrate,
        String accessproviderusername, String accessproviderpw, int providerid )
    {
        regDatabaseControl1.updateAccount(providername, status, selfpw ,
        address,email, url, servicetype,
        price, transaction, timeout,
        compensaterate, penaltyrate,
        accessproviderusername, accessproviderpw, providerid );

        return;
    }

    /**
     * @common:operation
     */
    public provider displayAccount(int providerid)
    {
       return regDatabaseControl1.getProvider(providerid);
    }
}
```

5. RegistryWebFlowController.jpf

```
// ---------------------------------------------------------------------------
//RegistryWebFlowController.jpf
//  Generated by Weblogic Workshop
//
//  Created on: Fri Oct 17 16:41:11 CDT 2003
//  By: Yutu
// ---------------------------------------------------------------------------
package RegistryWebFlow;

import com.bea.wlw.netui.pageflow.FormData;
import com.bea.wlw.netui.pageflow.Forward;

import resources.RegJavaControl;
import JavaClasses.provider;

/**
 * PageFlow class generated from control RegJavaControl
 *
 */
public class RegistryWebFlowController extends com.bea.wlw.netui.pageflow.PageFlowController
{
    /**
     *  This is the control used to generate this pageflow
     *  @common:control
     */
    private RegJavaControl myControl;

    //    Uncomment this declaration to access Global.app.
    //
    //    protected global.Global globalApp;
```

```
//

int providerid = 0;
/**
 * This method represents the point of entry into the pageflow
 *
 * @jpf:action
 * @jpf:forward name="success" path="index.jsp"
 */
protected Forward begin()
{
    return new Forward( "success" );
}

/**
 * Action encapsulating the control method : createAccount
 *
 * @jpf:action
 * @jpf:forward name="success" path="index.jsp"
 */
public Forward createAccount( CreateAccountForm aForm )
{
    try
    {
        myControl.createAccount(aForm.providername,
        aForm.selfpw, aForm.address, aForm.email, aForm.url,
        aForm.servicetype, aForm.price, aForm.transaction,
        aForm.timeout, aForm.compensaterate, aForm.penaltyrate,
        aForm.accessproviderusername, aForm.accessproviderpw);
        getRequest().setAttribute( "results", "   succeed to create account" );

        return new Forward( "success" );

    }
    catch( Throwable ex )
    {
        ex.printStackTrace();
    }
    return null;
}

/**
 * Action encapsulating the control method : login
 *
 * @jpf:action
 * @jpf:forward name="success" path="updateAccount.jsp"
 * @jpf:forward name="failed" path="index.jsp"
 */
public Forward login( LoginForm aForm )
{
    try
    {
        int var = myControl.login(aForm.user, aForm.pwd);
        //getRequest().setAttribute( "results", new Integer ( var ) );
        providerid = var;

        if( var == 0 )
        {
          getRequest().setAttribute( "results", "    failed to login");
          return new Forward( "failed");
        }
        else
```

```
        {
          getRequest().setAttribute( "results", "    succeed to log in");
          provider pvd = myControl.displayAccount(var);
          CreateAccountForm aform = new CreateAccountForm();
          aform.setProvidername( pvd.getProvidername() );
          aform.setSelfpw(pvd.getSelfpw());
          aform.setAddress(pvd.getSelfpw());
          aform.setEmail(pvd.getEmail());
          aform.setUrl(pvd.getUrl() );
          aform.setServicetype( pvd.getServicetype());
          aform.setPrice( pvd.getPrice());
          aform.setTransaction( pvd.getTransaction());
          aform.setTimeout( pvd.getTimeout());
          aform.setStatus(pvd.getStatus());
          aform.setCompensaterate(pvd.getCompensaterate());
          aform.setPenaltyrate( pvd.getPenaltyrate());
          aform.setAccessproviderusername( pvd.getAccessproviderusername());
          aform.setAccessproviderpw( pvd.getAccessproviderpw());
          aform.setProviderid(var);
           return new Forward( "success", aform );
        }

    }
    catch( Throwable ex )
    {
        ex.printStackTrace();
    }
    return null;
}

/**
 * Action encapsulating the control method : login
 *
 * @jpf:action
 * @jpf:forward name="success" path="index.jsp"
 * @jpf:forward name="failed" path="index.jsp"
 */
public Forward updateAccount( CreateAccountForm aForm )
{
    try
    {
        myControl.updateAccount(aForm.getProvidername(), aForm.getStatus(),
          aForm.getSelfpw(),
        aForm.getAddress(),aForm.getEmail(), aForm.getUrl(), aForm.getServicetype(),
        aForm.getPrice(), aForm.getTransaction(), aForm.getTimeout(),
        aForm.getCompensaterate(), aForm.getPenaltyrate(),
        aForm.getAccessproviderusername(),aForm.getAccessproviderpw(),providerid);
        getRequest().setAttribute( "results", "succeed to update account" );

        return new Forward( "success" );

    }
    catch( Throwable ex )
    {
        ex.printStackTrace();
    }
    return null;
}

/**
 * FormData class CreateAccountForm
 *
```

```
 */
public static class CreateAccountForm extends FormData
{
   private java.lang.String providername;
   private java.lang.String selfpw;
   private java.lang.String address;
   private java.lang.String email;
   private java.lang.String url;
   private java.lang.String servicetype;
   private double price;
   private int transaction;
   private int timeout;
   private double compensaterate;
   private double penaltyrate;
   private java.lang.String accessproviderusername;
   private java.lang.String accessproviderpw;
   private java.lang.String message;
   private int providerid;
   private java.lang.String status;

   public void setProvidername( java.lang.String providername )
   {
      this.providername = providername;
   }

   public java.lang.String getProvidername()
   {
      return providername;
   }

   public void setSelfpw( java.lang.String selfpw )
   {
      this.selfpw = selfpw;
   }

   public java.lang.String getSelfpw()
   {
      return selfpw;
   }

   public void setAddress( java.lang.String address )
   {
      this.address = address;
   }

   public java.lang.String getAddress()
   {
      return address;
   }

   public void setEmail( java.lang.String email )
   {
      this.email = email;
   }

   public void setProviderid( int providerid )
   {
      this.providerid = providerid;
   }
    public void setStatus( java.lang.String status )
   {
      this.status = status;
```

```
}

public java.lang.String getEmail()
{
   return email;
}

public void setUrl( java.lang.String url )
{
   this.url = url;
}

public java.lang.String getUrl()
{
   return url;
}

public void setServicetype( java.lang.String servicetype )
{
   this.servicetype = servicetype;
}

public java.lang.String getServicetype()
{
   return servicetype;
}

public void setPrice( double price )
{
   this.price = price;
}

public double getPrice()
{
   return price;
}

public void setTransaction( int transaction )
{
   this.transaction = transaction;
}

public int getTransaction()
{
   return transaction;
}

public void setTimeout( int timeout )
{
   this.timeout = timeout;
}

public int getTimeout()
{
   return timeout;
}

public void setCompensaterate( double compensaterate )
{
   this.compensaterate = compensaterate;
}
```

```
    public void setMessage( java.lang.String message )
    {
        this.message = message;
    }

    public double getCompensaterate()
    {
        return compensaterate;
    }

    public void setPenaltyrate( double penaltyrate )
    {
        this.penaltyrate = penaltyrate;
    }

    public double getPenaltyrate()
    {
        return penaltyrate;
    }

    public void setAccessproviderusername( java.lang.String accessproviderusername )
    {
        this.accessproviderusername = accessproviderusername;
    }

    public java.lang.String getAccessproviderusername()
    {
        return accessproviderusername;
    }

    public void setAccessproviderpw( java.lang.String accessproviderpw )
    {
        this.accessproviderpw = accessproviderpw;
    }

    public java.lang.String getAccessproviderpw()
    {
        return accessproviderpw;
    }

    public java.lang.String getMessage()
    {
        return message;
    }

    public java.lang.String getStatus()
    {
        return status;
    }
    public int getProviderid()
    {
        return providerid;
    }
}

/**
 * FormData class LoginForm
 *
 */
public static class LoginForm extends FormData
{
    private java.lang.String user;
```

```
    private java.lang.String pwd;

    public void setUser( java.lang.String user )
    {
       this.user = user;
    }

    public java.lang.String getUser()
    {
       return user;
    }

    public void setPwd( java.lang.String pwd )
    {
       this.pwd = pwd;
    }

    public java.lang.String getPwd()
    {
       return pwd;
    }
  }
}
```

6. Login.jsp

Provider Name

Password

login

7. updateAccount.jsp

Provider Name	ATT
Registry Password	ups
Account Status	active
Web Service Url	http://localhost:7001/Busi
Email	att@att.com
Address	ups
Timeout Value	200
Transaction Value	5
Service Type	telcom
Price	40.0
Penalty Rate	0.1
Compensate Rate	0.8
Access Provider Name	ups
Access Provider Password	ups

Update Account

APPENDIX II

PROGRAMMING SOURCE CODE OF UPS PORTAL

The following material is the Java source code of the implementation of UPS portal which is created in BEA Weblogic Workshop.

A. Business Logics

1.CustomerControl.jws
2. myDatabaseControl.jcs
3. myDatabaseController.jcx
4. UPSWebService.jws

B. Web Pages

5. checkoutpage.jsp
6. CustomerPageFlowController.jpf
7. DisplayOrderHistory.jsp
8 . index.jsp
9. Login.jsp
10. LoginResult.jsp
11. Registry.jsp
12. RegistryResult.jsp
13. SearchService.jsp
14. showFeature.jsp
15. ShowSearchResult.jsp
16. ShowSuccessOrder.jsp

17. UpdateAccount.jsp

C. Java containers for XML

18. dclient.java
19. feature.java
20. order.java
21. package.java
22 provider.java

D. Schema files

23 Order.xsd
24. Package.xsd
25. Provider.xsd

1. CustomerControl.jws

```
package CustomerControlFiles;

import weblogic.jws.control.JwsContext;
import weblogic.jws.proxies.BusinessProcessor_Impl;
import weblogic.jws.proxies.BusinessProcessorSoap;
import java.util.Vector;
import java.util.Random;
import org.openuri.bea.samples.workshop.PackageType;
import org.openuri.bea.samples.workshop.FeatureType;
import org.openuri.bea.samples.workshop.ProviderType;
import org.openuri.bea.samples.workshop.OrderType;
import org.openuri.bea.samples.workshop.CustomerType;

import org.openuri.bea.samples.workshop.OrderAllData;
import org.openuri.bea.samples.workshop.OrderAllDataDocument;
import org.openuri.bea.samples.workshop.CancelOrderData;
import org.openuri.bea.samples.workshop.CancelOrderDataDocument;
import org.openuri.bea.samples.workshop.ConfirmOrderData;
import org.openuri.bea.samples.workshop.ConfirmOrderDataDocument;
import org.openuri.bea.samples.workshop.PackageAllData;
import org.openuri.bea.samples.workshop.PackageAllDataDocument;
import org.openuri.bea.samples.workshop.ProviderAllData;
import org.openuri.bea.samples.workshop.ProviderAllDataDocument;
import org.openuri.bea.samples.workshop.TestData;
import org.openuri.bea.samples.workshop.TestDataDocument;
import org.openuri.bea.samples.workshop.RemoveServiceData;
import org.openuri.bea.samples.workshop.RemoveServiceDataDocument;
import com.bea.xml.XmlException;

import databaseJavaClass.*;
import java.lang.reflect.Array;
import org.openuri.bea.samples.workshop.SearchResultData;
import weblogic.jws.proxies.AccessWebServices_Impl;
import weblogic.jws.proxies.AccessWebServicesSoap;
import org.openuri.bea.samples.workshop.ProviderbType;
import org.openuri.bea.samples.workshop.ProviderbAllData;
import weblogic.jws.proxies.CreditAgent_Impl;
import weblogic.jws.proxies.CreditAgentSoap;
import org.openuri.bea.samples.workshop.QosData;
import java.util.Calendar;
//import CustomerPageFlow.CustomerPageFlowController.accountForm;


public class CustomerControl
{

   /**
    * @common:control
    */
   private CustomerResource.myDatabaseControl1 myDatabase;

   /** @common:context */
   JwsContext context;



    /**
    * @common:operation
    */
```

```
public databaseJavaClass.packages[] SearchService( String searchText,
      boolean pricesensitive, boolean servicesensitive)
// if it is empty, return null
{

   String WebServiceResult;


   // packages[] EJBResultArray = searchProviderWithEJB(searchText);
   // put two together

   // MDBResult = searchProviderWithMDB(searchText);
   // return WebServiceResult;

    //  WebServiceResult
   Vector resVector = new Vector(searchProviderWithWebService(searchText,
                        pricesensitive, servicesensitive));

   if( resVector != null && resVector.size() != 0 )
   {
      int vsize = resVector.size();
      databaseJavaClass.packages[] resPack = new databaseJavaClass.packages[vsize];
      resVector.copyInto(resPack);
      return  resPack;
   }else
   {
      return null;
   }

 }


private Vector searchProviderWithWebService(String searchText,
            boolean pricesensitive, boolean servicesensitive)
// return the list of all the available packages
{
  provider[] pl = new provider[1];
  int serviceid;
  int servicepwd;

  try{
    // Setup the global JAXM message factory
    System.setProperty("javax.xml.soap.MessageFactory",
    "weblogic.webservice.core.soap.MessageFactoryImpl");
    // Setup the global JAX-RPC service factory
    System.setProperty( "javax.xml.rpc.ServiceFactory",
    "weblogic.webservice.core.rpc.ServiceFactoryImpl");

    String url1=
"http://localhost:7001/AccessWebServices/AccessWebServiceFile/AccessWebServices.jws?WSDL";
    AccessWebServices_Impl bpstub1 = new AccessWebServices_Impl(url1);
    AccessWebServicesSoap businessSoap1 = bpstub1.getAccessWebServicesSoap();

    // set the price and service sensitive factors
    int ps, ss;
    if( pricesensitive == true )
      ps = 2;
    else
      ps = 1;
    if( servicesensitive == true )
      ss = 2;
    else
```

```
   ss = 1;

int numberOfProvider = 1;
String providerType = "telcom";
String xmlfile = businessSoap1.getServices(providerType, numberOfProvider, ps, ss);

if( xmlfile != null )
{
   ProviderbAllData ProviderInfo = ProviderbAllData.Factory.parse(xmlfile);
   ProviderbType[] providerArray = ProviderInfo.getProviderbArray();

   pl[0] = new provider();
   pl[0].setProvidername(providerArray[0].getName());
   pl[0].accessproviderpw  = providerArray[0].getAccesspwd();
   pl[0].accessproviderusername = providerArray[0].getAccessname();
   pl[0].url = providerArray[0].getUrl();
   serviceid = providerArray[0].getServiceid();
   servicepwd = providerArray[0].getServicepwd();


   //in this case, only one provider is selected,
   //this can also be changed into multi-providers
   for( int j=0; j<=0; j++)
   {
     provider pd = new provider();
     pd.providerid = 0;
     pd.providername = providerArray[0].getName();
     pd.status = "active";
     pd.selfpw = providerArray[0].getSelfpwd();
     pd.address = providerArray[0].getAddress();
     pd.email = providerArray[0].getEmail();
     pd.url = providerArray[0].getUrl();
     pd.comtype = "SOAP";
     pd.accessproviderpw = providerArray[0].getAccesspwd();
     pd.accessproviderusername = providerArray[0].getAccessname();

     provider[] providerArrayB = myDatabase.ShowAllproviderInfo();

    if( providerArrayB == null )
      return null;
     //found if this provider is already in the database
     boolean found = false;
     for( int id=0; id < providerArrayB.length; id++)
     {
       if((providerArrayB[id].getProvidername()).equals(providerArray[0].getName()))
       {
          found = true;
          break;
       }
     }
     if( found == false ) //this provider is not in the database
       myDatabase.AddProvider(pd);

  }
}else
   return null;


//provider[] pl = myDatabase.ShowAllActiveProviderInfo();
String WebServiceResult = null;
Vector pklist = new Vector();//container for all the packages
```

```
 try{

  for( int i=0; i < pl.length; i++ )
  {
     try{
      // Setup the global JAXM message factory
      System.setProperty("javax.xml.soap.MessageFactory",
       "weblogic.webservice.core.soap.MessageFactoryImpl");
       // Setup the global JAX-RPC service factory
       System.setProperty( "javax.xml.rpc.ServiceFactory",
        "weblogic.webservice.core.rpc.ServiceFactoryImpl");

       String user = pl[i].accessproviderusername;
       String password =pl[i].accessproviderpw;
       String url= pl[i].url;

       BusinessProcessor_Impl bpstub = new BusinessProcessor_Impl(url);
       BusinessProcessorSoap businessSoap = bpstub.getBusinessProcessorSoap();
       WebServiceResult = businessSoap.searchService(user,password,searchText);

     }
     catch( Exception e){ ;}//if encounter problem with one provider, ignore it.

     if( WebServiceResult != null )
     {
        PackageAllData PackageInfo = PackageAllData.Factory.parse(WebServiceResult);
        PackageType[] packageArray = PackageInfo.getPackageArray();

        for( int j=0; j<packageArray.length; j++)
        {
           packages resPack = new packages();

           ProviderType prov = packageArray[j].getProvider();
           FeatureType[] featureArray = packageArray[j].getFeatureArray();
           resPack.price = packageArray[j].getPrice();
           resPack.packagename = packageArray[j].getId();
           resPack.providername = prov.getName();
           resPack.serviceid = serviceid;
           resPack.servicepwd = servicepwd;

           resPack.featureList = new feature[featureArray.length];

           for( int k=0; k<featureArray.length; k++)
           {
             resPack.featureList[k] = new feature();
             resPack.featureList[k].featurename  = new String(featureArray[k].getId());
             resPack.featureList[k].price = featureArray[k].getPrice();
             resPack.featureList[k].option = featureArray[k].getOption();
           }
           pklist.add(resPack);
         }
       }else
          continue;
 }
if( pklist != null && pklist.size() != 0 )
  return pklist;
else
  return null;

 }
 catch( Exception e )
 {
```

```
    return null;
  }
 }catch( Exception e){return null;}//if encounter problem with one p

}


// sub-funtion for placeOrder()
private boolean PlaceOrderOfOneProvider( dclient aclient, String url, String password,
                                String user, int orderid, packages[] pkArray )
{

   try{

   OrderAllData OrderDataXML;
   OrderDataXML = OrderAllData.Factory.newInstance();

   OrderType anOrder = OrderDataXML.addNewOrder();
   CustomerType ac = anOrder.addNewCustomer();
   ac.setId(99); //not used, but need to construct the data
   ac.setAddress(aclient.address);
   ac.setEmail(aclient.email);
   ac.setFirstname(aclient.fname);
   ac.setLastname(aclient.lname);
   anOrder.setId(orderid)   ;

   //add all the packages
  for( int j=0; j < pkArray.length; j++)
   {

      PackageType pkg = anOrder.addNewPackage();
      ProviderType prov = pkg.addNewProvider();
      prov.setName("SBC");
      prov.setEmail("alinux@netzero.net");
      prov.setAddress("200 SBC Street,Austin");
      prov.setId("SBC");
      pkg.setId( pkArray[j].packagename );

      for( int k=0; k<pkArray[j].featureList.length; k++)
      {
         FeatureType af = pkg.addNewFeature();

         af.setId(pkArray[j].featureList[k].featurename);
         af.setOption(pkArray[j].featureList[k].option);
         af.setPrice( (float)pkArray[j].featureList[k].price );
      }
    }

    // Setup the global JAXM message factory
   System.setProperty("javax.xml.soap.MessageFactory",
     "weblogic.webservice.core.soap.MessageFactoryImpl");
   // Setup the global JAX-RPC service factory
   System.setProperty( "javax.xml.rpc.ServiceFactory",
     "weblogic.webservice.core.rpc.ServiceFactoryImpl");

   BusinessProcessor_Impl bpstub = new BusinessProcessor_Impl(url);
   BusinessProcessorSoap businessSoap = bpstub.getBusinessProcessorSoap();
   businessSoap.placeOrder( user, password, OrderDataXML.xmlText() );

   // set up to access registry web services
   String url1= "http://localhost:7001/AccessWebServices
            /AccessWebServiceFile/AccessWebServices.jws?WSDL";
```

```
    AccessWebServices_Impl bpstub1 = new AccessWebServices_Impl(url1);
    AccessWebServicesSoap businessSoap1 = bpstub1.getAccessWebServicesSoap();

    //get the execution time
    Calendar rightNow = Calendar.getInstance();
    long beforetime = rightNow.getTimeInMillis();
    businessSoap.placeOrder( user, password, OrderDataXML.xmlText() );
    long aftertime = rightNow.getTimeInMillis();
    int exectime = (int) ( aftertime - beforetime );

    //create the QoS xml file
    QosData ResultDataXML = QosData.Factory.newInstance();
    ResultDataXML.setProvidername(pkArray[0].getProvidername());
    // the reputation is given as 3
    ResultDataXML.setReputation(3);
    ResultDataXML.setTime(exectime+1);
    ResultDataXML.setId(pkArray[0].getServiceid());
    ResultDataXML.setPwd(pkArray[0].getServicepwd());

    businessSoap1.updateQoS(ResultDataXML.xmlText());

    return true;

    }catch( Exception e )
    {
        return false;
    }
}

  /**
  * @common:operation
  */

  public boolean PlaceOrder(int clientid, dclient ac, packages[] pckArray )
  {
     // one order can have mult-packages and need to place order with mult-providers
      int orderid;
      // get the reference number from database to pass to provider
      orderid = myDatabase.PlaceOrder(clientid, pckArray);

      Vector provList = new Vector();

      for( int i=0; i<pckArray.length; i++)
      {
         // if the provider is not included, then include it
         if( !provList.contains( pckArray[i].getProvidername()) )
         {
             provList.add( pckArray[i].getProvidername() );
         }
      }

      int provsize[] = new int[provList.size()];

      for( int j=0; j<provList.size(); j++)
      {
         provsize[j]=0;
         for( int i=0; i<pckArray.length; i++)
         {
            // if the provider is not included, then include it
            if( ((String) provList.get(j)).equals(pckArray[i].getProvidername()) )
               provsize[j]++;
          }
```

```
    }

    for( int j = 0; j < provList.size(); j++)
    {
       databaseJavaClass.packages[] pkArray = new databaseJavaClass.packages[provsize[j]];

       int idx = 0;
       for( int i=0; i<pckArray.length; i++)
       {
          if( ((String) provList.get(j)).equals(pckArray[i].getProvidername()) )
          {
             //put pckArray[i] into orderPack[idx];
             pkArray[idx] = new databaseJavaClass.packages();

             pkArray[idx].packageid = pckArray[i].packageid;
             pkArray[idx].packagename = pckArray[i].packagename;
             pkArray[idx].price = pckArray[i].price;
             pkArray[idx].providername = pckArray[i].providername;
             pkArray[idx].status = pckArray[i].status;
             pkArray[idx].serviceid = pckArray[idx].serviceid;
             pkArray[idx].servicepwd = pckArray[idx].servicepwd;

             int featlengd = pckArray[i].featureList.length;
             databaseJavaClass.feature[] featArrayd =
                      new  databaseJavaClass.feature[featlengd] ;

             for( int k=0; k<featlengd; k++)
             {
                featArrayd[k] = new databaseJavaClass.feature();

                featArrayd[k].featureid = pckArray[i].featureList[k].featureid;
                featArrayd[k].featurename = pckArray[i].featureList[k].featurename;
                featArrayd[k].option = pckArray[i].featureList[k].option;
                featArrayd[k].price = pckArray[i].featureList[k].price;
             }
                pkArray[idx].featureList = featArrayd;
              idx++;
           }
       }//gone through the whole pckArray list for one provider

        // place the order
       // get the list of providers
       provider[] pl = myDatabase.ShowAllActiveProviderInfo();

       for( int q=0; q < pl.length; q++)
       {

          if( (pl[q].getProvidername()).equals( (String) provList.get(j) ) )
          {
             String user = pl[q].accessproviderusername;
             String password =pl[q].accessproviderpw;
             String url= pl[q].url;

             PlaceOrderOfOneProvider( ac, url, password, user, orderid, pkArray );
             break;
          }
        }
    }
    return true;

}
```

```
/**
 * @common:operation
 */
public boolean CancelService(int refnumber, packages ap )
{
  // update the database
  myDatabase.CancelService(ap.packageid);
 // get the list of providers
 provider[] pl = myDatabase.ShowAllActiveProviderInfo();

  try{  // Setup the global JAXM message factory
    System.setProperty("javax.xml.soap.MessageFactory",
      "weblogic.webservice.core.soap.MessageFactoryImpl");
     // Setup the global JAX-RPC service factory
    System.setProperty( "javax.xml.rpc.ServiceFactory",
     "weblogic.webservice.core.rpc.ServiceFactoryImpl");

    for( int q=0; q < pl.length; q++ )
    {

       if( (pl[q].getProvidername()).equals( (String) ap.getProvidername()) )
        {
          String user = pl[q].accessproviderusername;
          String password =pl[q].accessproviderpw;
          String url= pl[q].url;
          BusinessProcessor_Impl bpstub = new BusinessProcessor_Impl(url);
          BusinessProcessorSoap businessSoap = bpstub.getBusinessProcessorSoap();

          businessSoap.cancelService(user, password, ap.getPackagename(), refnumber);
          return true;
        }
     }
     return true;
   }catch( Exception e){ return false;}
}


/**
 * @common:operation
 */
public int Login(String user, String password )
{
    int clientid = myDatabase.MatchUserPassword(user,password);

    return clientid;
}

/**
 * @common:operation
 */
//need client id
public databaseJavaClass.packages[] DisplayOrderHistory( int customID )
{
    Vector var = myDatabase.ShowClientOrderHistory(customID);

    if( var != null && var.size() != 0 )
    {
          int vsize = var.size()/6;
          databaseJavaClass.packages[] orderHs = new databaseJavaClass.packages[vsize];
```

```
            for( int i=0; i< var.size()/6; i++)
            {
              orderHs[i]= new packages();
              orderHs[i].setPackagename( (String) var.get(i*6+1) );
              orderHs[i].setProvidername( (String) var.get(i*6) );
              orderHs[i].setStartdate( (var.get(i*6+2)).toString() );
              orderHs[i].setPackageid(((Integer) var.get(i*6+3)).intValue());
              orderHs[i].setPrice( ((Double) var.get(i*6+4)).doubleValue() );
              orderHs[i].setStatus( (String) var.get(i*6+5) );
              orderHs[i].setEnddate( Integer.toString(i) );
            }

             return orderHs;
        }
        else
            return null;
    }


    /**
     * @common:operation
     */
    // need client id
    public dclient DisplayAccount( int customID )
    {
         return myDatabase.ShowClientAccount(customID);
    }

    /**
     * @common:operation
     */
    public boolean AddOneClient(databaseJavaClass.dclient ac)
    {
        try{
        // Setup the global JAXM message factory
        System.setProperty("javax.xml.soap.MessageFactory",
        "weblogic.webservice.core.soap.MessageFactoryImpl");
        // Setup the global JAX-RPC service factory
        System.setProperty( "javax.xml.rpc.ServiceFactory",
        "weblogic.webservice.core.rpc.ServiceFactoryImpl");
          String url1= "http://localhost:7001/AccessWebServices/AccessWebServiceFile
                                /AccessWebServices.jws?WSDL";
        AccessWebServices_Impl bpstub1 = new AccessWebServices_Impl(url1);
        AccessWebServicesSoap businessSoap1 = bpstub1.getAccessWebServicesSoap();

        int pricesensitive = 1;
        int servicesensitive = 1;
        int numberOfProvider = 1;
        String providerType = "credit";
        String xmlfile = businessSoap1.getServices(providerType, numberOfProvider,
                            pricesensitive, servicesensitive);

        if( xmlfile != null )
        {
          //parse data from the xml file
          ProviderbAllData ProviderInfo = ProviderbAllData.Factory.parse(xmlfile);
          ProviderbType[] providerArray = ProviderInfo.getProviderbArray();

          String providername=providerArray[0].getName();
          String accessproviderpw  = providerArray[0].getAccesspwd();
          String accessproviderusername = providerArray[0].getAccessname();
          String url = providerArray[0].getUrl();
```

```
      int serviceid = providerArray[0].getServiceid();
      int servicepwd = providerArray[0].getServicepwd();
      String customerName = ac.lname;

      CreditAgent_Impl bpstub = new CreditAgent_Impl(url);
      CreditAgentSoap businessSoap = bpstub.getCreditAgentSoap();

      //get the execution time
      Calendar rightNow = Calendar.getInstance();
      long beforetime = rightNow.getTimeInMillis();
      int result = businessSoap.checkCredit( customerName,
                  accessproviderusername, accessproviderpw );
      long aftertime = rightNow.getTimeInMillis();
      int exectime = (int) ( aftertime - beforetime );
      QosData ResultDataXML = QosData.Factory.newInstance();

      ResultDataXML.setProvidername(providername);
      // the reputation is given as 3
      ResultDataXML.setReputation(3);
      ResultDataXML.setTime(exectime+1);
      ResultDataXML.setId(serviceid);
      ResultDataXML.setPwd(servicepwd);

      businessSoap1.updateQoS(ResultDataXML.xmlText());

      if( result == 5 )
      {
         myDatabase.AddClientAccount(ac);
         return true;
      }
      else
         return false;
   }else
      return false;
  }catch( Exception e){ return false;}
}

/**
 * @common:operation
 */
public void UpdataClientAccount(databaseJavaClass.dclient ac)
{
   myDatabase.UpdateClientAccount(ac);
}

/**
 * @common:operation
 */
public boolean Simulation(int num)
{

  try{
    // Setup the global JAXM message factory
    System.setProperty("javax.xml.soap.MessageFactory",
    "weblogic.webservice.core.soap.MessageFactoryImpl");
    // Setup the global JAX-RPC service factory
    System.setProperty( "javax.xml.rpc.ServiceFactory",
    "weblogic.webservice.core.rpc.ServiceFactoryImpl");

    String url1= "http://localhost:7001/AccessWebServices
                /AccessWebServiceFile/AccessWebServices.jws?WSDL";
    AccessWebServices_Impl bpstub1 = new AccessWebServices_Impl(url1);
```

```
AccessWebServicesSoap businessSoap1 = bpstub1.getAccessWebServicesSoap();

// set the price and service sensitive factors
int ps =1;
int ss =1;

int numberOfProvider = 1;
String providerType = "telcom";

for( int j=0; j<num; j++)
{
  // let the price sensitive and service sensitive
  // search have equal opportunities
  if( j%2 == 1)
  {
    ps = 1;
    ss = 2;
  }
  else
  {
    ps = 2;
    ss = 1;
  }

  String xmlfile = businessSoap1.getServices(providerType, numberOfProvider, ps, ss);
  // read date from xml of selected provider
  ProviderbAllData ProviderInfo = ProviderbAllData.Factory.parse(xmlfile);
  ProviderbType[] providerArray = ProviderInfo.getProviderbArray();

  String providername = providerArray[0].getName();
  String password  = providerArray[0].getAccesspwd();
  String user = providerArray[0].getAccessname();
  String url2 = providerArray[0].getUrl();
  String searchText = "caller ID";
  int serviceid = providerArray[0].getServiceid();
  int servicepwd = providerArray[0].getServicepwd();

  // search service from selected provider
  BusinessProcessor_Impl bpstub2 = new BusinessProcessor_Impl(url2);
  BusinessProcessorSoap businessSoap2 = bpstub2.getBusinessProcessorSoap();
  String WebServiceResult = businessSoap2.searchService(user,password,searchText);

  // get data from xml file about services
  PackageAllData PackageInfo = PackageAllData.Factory.parse(WebServiceResult);
  PackageType[] packageArray = PackageInfo.getPackageArray();

  packages resPack = new packages();

  ProviderType prov = packageArray[0].getProvider();
  FeatureType[] featureArray = packageArray[0].getFeatureArray();
  resPack.price = packageArray[0].getPrice();
  resPack.packagename = packageArray[0].getId();
  resPack.providername = prov.getName();
  resPack.serviceid = serviceid;
  resPack.servicepwd = servicepwd;

  resPack.featureList = new feature[featureArray.length];

  for( int k=0; k<featureArray.length; k++)
  {
     resPack.featureList[k] = new feature();
     resPack.featureList[k].featurename  = new String(featureArray[k].getId());
```

```
    resPack.featureList[k].price = featureArray[k].getPrice();
    resPack.featureList[k].option = featureArray[k].getOption();
}

// construct the xml file of order
OrderAllData OrderDataXML;
OrderDataXML = OrderAllData.Factory.newInstance();

OrderType anOrder = OrderDataXML.addNewOrder();
CustomerType ac = anOrder.addNewCustomer();
ac.setId(99); //not used, but need to construct the data
ac.setAddress("client address");
ac.setEmail("aclientemail");
ac.setFirstname("aclientfname");
ac.setLastname("aclientlname");
int clientid = 1;
anOrder.setId(clientid)   ;

//add all the packages to the order xml
PackageType pkg = anOrder.addNewPackage();
ProviderType prov2 = pkg.addNewProvider();
prov2.setName(resPack.providername);
prov2.setEmail("eamil");
prov2.setAddress("200 SBC Street,Austin");
prov2.setId("SBC");
pkg.setId( resPack.packagename );

 for( int k=0; k<resPack.featureList.length; k++)
 {
    FeatureType af = pkg.addNewFeature();

    af.setId(resPack.featureList[k].featurename);
    af.setOption(resPack.featureList[k].option);
    af.setPrice( (float)resPack.featureList[k].price );
 }

 //get the execution time
 Calendar rightNow = Calendar.getInstance();
 long beforetime = rightNow.getTimeInMillis();
 businessSoap2.placeOrder( user, password, OrderDataXML.xmlText() );
 long aftertime = rightNow.getTimeInMillis();
 int exectime = (int) ( aftertime - beforetime );

 //create the QoS xml file
 QosData ResultDataXML = QosData.Factory.newInstance();
 ResultDataXML.setProvidername(resPack.getProvidername());
 //generate random value for the reputation
 int rep;
 Random RandomGen = new Random();
 rep = RandomGen.nextInt();
 rep = rep%3;
 if( rep >= 0 )
   ResultDataXML.setReputation(rep);
 else
   ResultDataXML.setReputation(rep+3);
 int rt = RandomGen.nextInt();
 rt = rt%2;
 if( rt >= 0)
   ResultDataXML.setTime(exectime+rt);
 else
   ResultDataXML.setTime(exectime+rt+2);
 ResultDataXML.setId(resPack.getServiceid());
```

```
            ResultDataXML.setPwd(resPack.getServicepwd());

            businessSoap1.updateQoS(ResultDataXML.xmlText());
          }
          return true;
        }catch( Exception e){return false;}//if encounter problem with one p
      }

}
```

2. myDatabaseControls.jcs

```
package myDatabaseControls;

import weblogic.jws.util.Logger;
import weblogic.jws.control.JbcContext;
import java.util.Vector;
import databaseJavaClass.*;


/**
* @jcs:jc-jar label="myDatabaseControl"
*/
public class myDatabaseControl
{
    /**
     * @common:control
     */
    private myDatabaseControls.myDatabaseController myDatabase;

    /**
     * @common:context
     */
    JbcContext context;

    /**
     * @common:operation
     */
        public int PlaceOrder( int clientid, packages[] packagesList )
      {

          myDatabase.addEntryInOrderTable( clientid );
          int id = myDatabase.findLastOrderEntryId();

          for( int i=0; i< packagesList.length; i++)
          {
            packages ap = packagesList[i];
            myDatabase.addEntryInPackageTable(id, ap.packagename, ap.providername,
              ap.price,"active");
            int id2 = myDatabase.findLastPackageEntryId();
            for( int j=0; j<ap.featureList.length; j++)
            {
                feature af = ap.featureList[j];
                myDatabase.addEntryInFeatureTable(id2,af.featurename);
            }
          }
          return id;
      }
```

```
/**
 * @common:operation
 */
public void CancelService( int packageid )
 {
    myDatabase.updataPackageStatus(packageid, "cancelled");
 }

/**
 * @common:operation
 */
 public void UpdateClientAccount(dclient ac )
 {
     myDatabase.updataClientAccountInfo(ac);
 }

/**
 * @common:operation
 */
 public void AddProvider(provider ap)
  {
    ap.setStatus("pending");
    myDatabase.addProvider(ap);
 }

/**
 * @common:operation
 */
// return a list of affected client with email address,package name, provider name
 public Vector RemoveProvider(String providername)
   {
      Vector alist = new Vector();

      myDatabase.updateProviderStatus(providername,"non_active");
      int[] idArray = myDatabase.findAllRemovedPackageId(providername);

      for( int i=0; i<idArray.length; i++)
      {
         myDatabase.updataPackageStatus(idArray[i],"non_active");
      }

      for( int i=0; i<idArray.length; i++)
      {
         try{
            java.sql.ResultSet sublist = myDatabase.findAllAffectedClientList(idArray[i]);
            while( sublist.next() )
            {
               alist.add( sublist.getString(1) );
               alist.add( sublist.getString(2) );
               alist.add( sublist.getString(3) );
            }
         }catch( Exception e){ return alist;}
      }

        return alist;
 }

/**
 * @common:operation
 */
 public void AproveProvider(String providername, String status )
 {
```

```
        myDatabase.updateProviderStatus( providername,status );
    }

   /**
    * @common:operation
    */
    public int MatchUserPassword(String user, String pw)
      {
        return myDatabase.matchUserPassword( user, pw);
    }

       /**
    * @common:operation
    */
    public int MatchProviderPassword(String user, String pw)
      {
        return myDatabase.matchProviderPassword( user, pw);
    }

   /**
    * @common:operation
    */
   public void AddClientAccount( dclient ac )
   {

      myDatabase.addClientAccount(ac);
   }

  /**
    * @common:operation
    */
   public dclient ShowClientAccount( int id )
   {
     return myDatabase.showClientAccount(id);
   }

  /**
    * @common:operation
    */
    // need client id
   public Vector ShowClientOrderHistory( int id )
   {
      java.sql.ResultSet ret = myDatabase.showAClientOrderHistory(id);
       try{
           Vector a = new Vector();
          while (ret.next())
          {
             String pname  = new String(ret.getString("providername"));
             a.add( pname);
             String pckname = new String(ret.getString("packagename"));
             a.add( pckname);
             int orderid  = ret.getInt("orderid");
             a.add(new Integer(orderid) );
             int packageid = ret.getInt("packageid");
             a.add( new Integer(packageid) );
             double price = ret.getDouble("price");
             a.add( new Double(price) );
             String status = new String(ret.getString("status"));
             a.add( status);
          }
          return a;
      }catch( Exception e) { return null;}
```

```
    }

   /**
    * @common:operation
    */
   public provider ShowOneProviderInfo(int id)
   {
      return myDatabase.showProvider(id);
   }

    /**
    * @common:operation
    */
   public void UpdateOrderStatus(int oid, String status)
   {
      myDatabase.updataOrderStatus( oid, status );
   }


   /**
    * @common:operation
    */
   public provider[] ShowAllproviderInfo()
   {
       return myDatabase.showAllProviderInfo();
   }

   /**
    * @common:operation
    */
   public provider[] ShowAllActiveProviderInfo()
   {
       return myDatabase.showAllActiveProviderInfo();
   }

   /**
    * @common:operation
    */
   public Vector ShowFeaturesUnderPackage(int orderid, String packagename, String providername)
   {
       java.sql.ResultSet res =
         myDatabase.showFeatureUnderPackage(orderid,packagename,providername);
           try{
            Vector a = new Vector();
           while (res.next())
           {
              String pname  = new String(res.getString("featurename"));
              a.add( pname);
           }
           return a;
       }catch( Exception e) { return null;}
   }
}
```

3. myDatabaseController.jcx

```
package myDatabaseControls;
import weblogic.jws.control.*;
import java.sql.SQLException;
import databaseJavaClass.*;
```

```
/**
 * Defines a new database control.
* @jc:connection data-source-jndi-name="UPSDataSource"
 */
public interface myDatabaseController extends DatabaseControl
{
    // Sample database function.  Uncomment to use

    // static public class Customer
    // {
    //   public int id;
    //   public String name;
    // }
    //
    // /**
    //  * @jc:sql statement="SELECT ID, NAME FROM CUSTOMERS WHERE ID = {id}"
    //  */
    // Customer findCustomer(int id);

    // Add "throws SQLException" to request that SQLExeptions be thrown on errors.


   /**
     * @jc:sql statement::
     *          INSERT          INTO          CLIENT          (EMAIL,PW,LNAME,FNAME,ADDRESS,CITY,STATE,
ZIP,CREDIT_CARD_NUM)
     *
VALUES({ac.email},{ac.pw},{ac.lname},{ac.fname},{ac.address},{ac.city},{ac.state},{ac.zip},{ac.credit_card
_num})::

     */
    void addClientAccount(dclient ac );

    /**
     * @jc:sql statement::
     * SELECT UID,LNAME, FNAME, PW, EMAIL, ADDRESS, CITY,STATE,ZIP,CREDIT_CARD_NUM
     * FROM CLIENT
     * WHERE UID = {id}::

     */
    dclient showClientAccount(int id );

    /**
     * @jc:sql statement::
     * SELECT * FROM PROVIDER
     * WHERE PROVIDERID = {id}::

     */
    provider showProvider( int id );

    /**
     * @jc:sql statement::
     *   INSERT INTO PROVIDER(PROVIDERNAME, STATUS, SELFPW, ADDRESS, EMAIL,
URL,COMTYPE,
     *                   ACCESSPROVIDERUSERNAME, ACCESSPROVIDERPW)
     *
VALUES({ap.providername},{ap.status},{ap.selfpw},{ap.address},{ap.email},{ap.url},{ap.comtype},{ap.acce
ssproviderusername}, {ap.accessproviderpw})
     * ::

     */
```

```
    void addProvider(provider ap);

    /**
     * @jc:sql statement::
     * UPDATE ORDER SET ACTIVE = {active}
     * WHERE ORDERID = {orderid}
     * ::

     */
    void updataOrderStatus(int orderid, String active);

    /**
     * @jc:sql statement::
     * UPDATE PROVIDER SET STATUS = {status}
     * WHERE PROVIDERNAME ={providername}::

     */
    void updateProviderStatus(String providername, String status);

    /**
     * @jc:sql statement="SELECT * FROM PROVIDER"
     */
    provider[] showAllProviderInfo();

    /**
     * @jc:sql statement="SELECT * FROM PROVIDER WHERE STATUS='active'"
     */
    provider[] showAllActiveProviderInfo();

    /**
     * @jc:sql statement::
     *    SELECT    ORDER.ORDERID,PACKAGEID,    PACKAGENAME,    PRICE,    PROVIDERNAME,
ORDER.ACTIVE, PACKAGE.STATUS
     * FROM ORDER, PACKAGE
     * WHERE CLIENTID = {clientid}
     * AND  ORDER.ORDERID = PACKAGE.ORDERID::

     */
   java.sql.ResultSet showAClientOrderHistory( int clientid );

    /**
     * @jc:sql statement="INSERT INTO ORDER VALUES({clientid},'active', CURRENT_DATE)"
     */
    void addEntryInOrderTable(int clientid);

    /**
     * @jc:sql statement::
     * SELECT DISTINCT (FEATURENAME)
     * FROM FEATURE, PACKAGE
     * WHERE  ORDERID= {orderid}
     * AND  PACKAGENAME = {packagename}
     * AND PACKAGE.PACKAGEID=FEATURE.PACKAGEID
     * AND PROVIDERNAME={providername}::

     */
    java.sql.ResultSet showFeatureUnderPackage(int orderid, String packagename, String providername);

    /**
     * @jc:sql statement="SELECT MAX(ORDERID) FROM ORDER"
     */
    int findLastOrderEntryId();
```

```
    /**
     * @jc:sql statement::
     * INSERT INTO PACKAGE(PACKAGENAME,ORDERID,PROVIDERNAME,PRICE,STATUS)
     * VALUES({packagename},{orderid},{providername},{price},{status})::

     */
    void addEntryInPackageTable(int orderid, String packagename, String providername, double
price,String status);

    /**
     * @jc:sql statement="SELECT MAX(PACKAGEID) FROM PACKAGE"
     */
    int findLastPackageEntryId();

    /**
     * @jc:sql statement::
     * INSERT INTO FEATURE(PACKAGEID,FEATURENAME)
     * VALUES({packageid},{featurename})::

     */
    void addEntryInFeatureTable(int packageid, String featurename);

    /**
     * @jc:sql statement::
     * SELECT PACKAGEID FROM PACKAGE
     * WHERE PROVIDERNAME={providername}
     *  ::

     */
    int[] findAllRemovedPackageId(String providername);

    /**
     * @jc:sql statement::
     * UPDATE PACKAGE SET STATUS={status}
     * WHERE PACKAGEID = {packageid}::

     */
    void updataPackageStatus(int packageid, String status);

    /**
     * @jc:sql statement::
     * SELECT EMAIL,PACKAGENAME,PROVIDERNAME
     * FROM CLIENT,ORDER,PACKAGE
     * WHERE PACKAGEID={packageid}
     * AND PACKAGE.ORDERID=ORDER.ORDERID
     * AND ORDER.CLIENTID=UID

     * ::

     */
    java.sql.ResultSet findAllAffectedClientList(int packageid);

    /**
     * @jc:sql statement::
     * UPDATE CLIENT
     * SET  EMAIL = {ac.email},
     *      PW = {ac.pw},
     *      LNAME={ac.lname},
     *      FNAME={ac.fname},
     *      ADDRESS={ac.address},
     *      CITY={ac.city},
     *      STATE={ac.state},
```

```
     *      ZIP={ac.zip},
     *      CREDIT_CARD_NUM={ac.credit_card_num}
     * WHERE UID={ac.uid}::

     */
    void updataClientAccountInfo(dclient ac);

    /**
     * @jc:sql statement::
     * SELECT UID FROM CLIENT WHERE
     * EMAIL={email} AND PW={pw}::

     */
    int matchUserPassword(String email, String pw);

     /**
     * @jc:sql statement::
     * SELECT PROVIDERID FROM PROVIDER WHERE
     * EMAIL={email} AND SELFPW={selfpw}::

     */
    int matchProviderPassword(String email, String selfpw);


}
```

4. UPSWebService.jws

```
package UPSWebServiceFile;

import weblogic.jws.control.JwsContext;
import org.openuri.bea.samples.workshop.ProviderType;
import org.openuri.bea.samples.workshop.ProviderAllData;


public class UPSWebService
{
    /**
     * @common:control
     */
    private WebServiceControl.myDatabaseControl1 myDatabase;

    /** @common:context */
    JwsContext context;
    /**
     * @common:operation
     */
    public boolean RemoveServiceFromUPS(String user, String password, String xmlfile)
    // for provider to remove themselves
    {
        try{
            ProviderAllData ProviderInfo = ProviderAllData.Factory.parse(xmlfile);
            ProviderType[] providerArray = ProviderInfo.getProviderArray();
            String providerName = providerArray[0].getName();
            myDatabase.RemoveProvider(providerName);

        }catch( Exception e ) {return false;}

        return true;
    }
```

```
    /**
     * @common:operation
     */
    public void ConfirmOrder(String user, String password, int refid )
    {
        myDatabase.UpdateOrderStatus(refid, "active");
    }
}
```

5. CustomerPageFlowController.jpf

```
// ----------------------------------------------------------------------------
//  Generated by Weblogic Workshop
//
//  By: Yutu
// ----------------------------------------------------------------------------
package CustomerPageFlow;

import com.bea.wlw.netui.pageflow.FormData;
import com.bea.wlw.netui.pageflow.Forward;
import java.text.NumberFormat;
import CustomerPageFlow.CustomerPageFlowController.FeatureForm;

import CustomerControls.CustomerControl1Control;

/**
* PageFlow class generated from control CustomerControl1Control
*
*/
public class CustomerPageFlowController extends com.bea.wlw.netui.pageflow.PageFlowController
{
/**
*  This is the control used to generate this pageflow
*  @common:control
*/
private CustomerControl1Control myControl;

private CustomerControls.CustomerControl1Control.packages[] searchResults;
private FeatureForm[] featurelist;

private CustomerControls.CustomerControl1Control.dclient aclient;
private CustomerControls.CustomerControl1Control.packages[] shopCart;
private CustomerControls.CustomerControl1Control.packages[] orderHistory;
private HistoryForm[] orderH;
private int clientid;
private boolean showCart;
private boolean showOrderHistory;

/**
* Getter for databinding.
*/
public CustomerControls.CustomerControl1Control.packages[] getPackages()
{
    if( showCart == true )
    {
       showCart = false;
        return shopCart;
    }
```

```
    else
    return searchResults;
}

/**
* Getter for databinding.
*/
public FeatureForm[] getFeature()
{
    return featurelist;
}

/**
* Getter for databinding.
*/
public HistoryForm[] getHistory()
{
     return orderH;
}

/**
* This method represents the point of entry into the pageflow
*
* @jpf:action
* @jpf:forward name="success" path="index.jsp"
*/
protected Forward begin()
{
clientid = 0;
shopCart = null;
aclient =  null;
showCart = false;
orderHistory = null;
featurelist = null;
searchResults = null;
return new Forward( "success" );
}

/**
* This method represents the point of entry into the pageflow
*
* @jpf:action
* @jpf:forward name="success" path="index.jsp"
*/
protected Forward logout()
{
clientid = 0;
shopCart = null;
aclient =  null;
showCart = false;
orderHistory = null;
featurelist = null;
searchResults = null;
return new Forward( "success" );
}

/**
* Action encapsulating the control method : SearchService
*
* @jpf:action
* @jpf:forward name="success" path="ShowSearchResults.jsp"
*/
```

```
public Forward SearchService( SearchServiceForm aForm )
{
try
{
CustomerControls.CustomerControl1Control.packages[] var =
myControl.SearchService(aForm.searchText, aForm.pricesensitive, aForm.servicesensitive);

getRequest().setAttribute( "results", var );
searchResults = var;

//this only pass a temp id for each package for later displaying feature
for( int i=0; i<searchResults.length; i++)
searchResults[i].packageid = i;

return new Forward( "success" );

}
catch( Throwable ex )
{
      ex.printStackTrace();
}
      return null;
}


/**
* @jpf:action
* @jpf:forward name="success" path="showFeature.jsp"
*/
public Forward ShowFeature()
{
String idstr = getRequest().getParameter( "packageid" );
Integer idobj = new Integer(idstr);
int id = idobj.intValue();
CustomerControls.CustomerControl1Control.feature[] featlist =
searchResults[id].featureList;

featurelist = new FeatureForm[featlist.length];

for(int i=0; i<featlist.length; i++)
{
featurelist[i] = new FeatureForm();
featurelist[i].setFeaturename(featlist[i].featurename);
featurelist[i].setPrice(featlist[i].price);
}

return new Forward("success");
}

/**
* @jpf:action
* @jpf:forward name="success" path="checkoutpage.jsp"
*/
public Forward ShowShopCart()
{
  showCart = true;

  return new Forward("success");
}

/**
* @jpf:action
```

```
* @jpf:forward name="success" path="checkoutpage.jsp"
* @jpf:forward name="success_2" path="index.jsp"
*/
public Forward CheckOut()
{
     if( shopCart == null )
     {
clientid = 0;
shopCart = null;
aclient =  null;
showCart = false;
orderHistory = null;
featurelist = null;
searchResults = null;
return new Forward("success_2");
     }
     else
     {
   showCart = true;
   return  new Forward("success");
}
}

/**
* @jpf:action
* @jpf:forward name="success" path="ShowSearchResults.jsp"
*/
public Forward AddPackage(messageForm mform)
{
String idstr = getRequest().getParameter( "packageid" );
Integer idobj = new Integer(idstr);
int id = idobj.intValue();

      if( shopCart == null )
      {
CustomerControls.CustomerControl1Control.packages[] pkArray =
new  CustomerControls.CustomerControl1Control.packages[1] ;
pkArray[0] = new  CustomerControls.CustomerControl1Control.packages();

pkArray[0].packagename = searchResults[id].packagename;
pkArray[0].price = searchResults[id].price;
pkArray[0].providername = searchResults[id].providername;
pkArray[0].status = "active";
pkArray[0].serviceid = searchResults[id].serviceid;
pkArray[0].servicepwd = searchResults[id].servicepwd;

int featleng = searchResults[id].featureList.length;
CustomerControls.CustomerControl1Control.feature[] featArray =
new  CustomerControls.CustomerControl1Control.feature[featleng] ;

   for( int i=0; i<featleng; i++)
   {
featArray[i] = new CustomerControls.CustomerControl1Control.feature();
featArray[i].featureid = searchResults[id].featureList[i].featureid;
featArray[i].featurename = searchResults[id].featureList[i].featurename;
featArray[i].option = searchResults[id].featureList[i].option;
featArray[i].price = searchResults[id].featureList[i].price;
 }
        pkArray[0].featureList = featArray;
         shopCart = pkArray;
}
else
```

```
{
int cartLeng = shopCart.length;

CustomerControls.CustomerControl1Control.packages[] pkArray =
new  CustomerControls.CustomerControl1Control.packages[cartLeng + 1] ;

// copy the contents of old shopping cart into the new shopping cart
for( int j=0; j<cartLeng; j++)
{
pkArray[j] = new  CustomerControls.CustomerControl1Control.packages();

pkArray[j].packagename = shopCart[j].packagename;
pkArray[j].price = shopCart[j].price;
pkArray[j].providername = shopCart[j].providername;
pkArray[j].status = "active";
pkArray[j].serviceid = shopCart[j].serviceid;
pkArray[j].servicepwd = shopCart[j].servicepwd;

int featlengd = shopCart[j].featureList.length;
CustomerControls.CustomerControl1Control.feature[] featArrayd =
new  CustomerControls.CustomerControl1Control.feature[featlengd] ;

for( int k=0; k<featlengd; k++)
{
featArrayd[k] = new CustomerControls.CustomerControl1Control.feature();

featArrayd[k].featureid = shopCart[j].featureList[k].featureid;
featArrayd[k].featurename = shopCart[j].featureList[k].featurename;
featArrayd[k].option = shopCart[j].featureList[k].option;
featArrayd[k].price = shopCart[j].featureList[k].price;
}
    pkArray[j].featureList = featArrayd;
       }


pkArray[cartLeng] = new  CustomerControls.CustomerControl1Control.packages();

pkArray[cartLeng].packagename = searchResults[id].packagename;
pkArray[cartLeng].price = searchResults[id].price;
pkArray[cartLeng].providername = searchResults[id].providername;
pkArray[cartLeng].status = "active";
pkArray[cartLeng].serviceid = searchResults[id].serviceid;
pkArray[cartLeng].servicepwd = searchResults[id].servicepwd;

int featleng = searchResults[id].featureList.length;
CustomerControls.CustomerControl1Control.feature[] featArray =
new  CustomerControls.CustomerControl1Control.feature[featleng] ;

for( int i=0; i<featleng; i++)
{
featArray[i] = new CustomerControls.CustomerControl1Control.feature();

featArray[i].featureid = searchResults[id].featureList[i].featureid;
featArray[i].featurename = searchResults[id].featureList[i].featurename;
featArray[i].option = searchResults[id].featureList[i].option;
featArray[i].price = searchResults[id].featureList[i].price;
 }
   pkArray[cartLeng].featureList = featArray;

  // the new shopping cart is ready
  shopCart = pkArray;
}
```

```
        mform.setMessage("You have added package "+ searchResults[id].packagename+" into your
        shop cart.");
        return new Forward("success",mform);
}



/**
* @jpf:action
* @jpf:forward name="success" path="RegisterResult.jsp"
*/
protected Forward CreateCustomerAccountAction(accountForm aForm)
{

CustomerControls.CustomerControl1Control.dclient bclient =
new CustomerControls.CustomerControl1Control.dclient();

databaseJavaClass.dclient ac;

bclient.email=aForm.email;
bclient.pw=aForm.pw;
bclient.lname=aForm.lname;
bclient.fname=aForm.fname;
bclient.address=aForm.address;
bclient.city=aForm.city;
bclient.state=aForm.state;
bclient.zip=aForm.zip;
bclient.credit_card_num=aForm.credit_card_num;

boolean success = myControl.AddOneClient(bclient);
if( success == false )
aForm.setMessage(aForm.fname+" "+ aForm.lname +", Your credit is not good, failed to
  create an account!");
else
{
      clientid = myControl.Login(bclient.email, bclient.pw);
     aForm.setMessage(aForm.fname+" "+ aForm.lname +", Congratulations!");
   }

      return new Forward("success", aForm);
}



/**
* Action encapsulating the control method : PlaceOrder
*
* @jpf:action
* @jpf:forward name="success" path="ShowSuccesOrder.jsp"
* @jpf:forward name="failed" path="Login.jsp"
*/
public Forward PlaceOrder()
{
   try
   {

if( clientid == 0 || aclient == null )
return new Forward("failed");
else
{
```

```
showCart = true;
if( shopCart != null )
myControl.PlaceOrder( clientid, aclient, shopCart );

shopCart = null;
return new Forward( "success" );

}
   }
   catch( Throwable ex )
   {
ex.printStackTrace();
    }
return new Forward("failed");
}

/**
* Action encapsulating the control method : CancelService
*
* @jpf:action
* @jpf:forward name = "success" path = "DisplayOrderHistory.jsp"
*/
public Forward CancelService( CancelServiceForm aForm )
{
  try
  {
String idstr = getRequest().getParameter( "tempid" );
Integer id = new Integer( idstr );
int idint = id.intValue();
int refnum = Integer.valueOf(orderHistory[idint].startdate).intValue();

boolean var = myControl.CancelService(refnum, orderHistory[idint]);
orderHistory = myControl.DisplayOrderHistory(clientid);

int sizev = orderHistory.length;
orderH = new HistoryForm[sizev];

for( int i=0; i<sizev; i++)
{
orderH[i] = new HistoryForm();
orderH[i].setPackagename(orderHistory[i].packagename );
orderH[i].setProvidername(orderHistory[i].providername );
orderH[i].setStatus(orderHistory[i].status);
orderH[i].setPrice(orderHistory[i].price );
orderH[i].setTempid( orderHistory[i].enddate );
}
showOrderHistory = true;
return new Forward( "success" );

     }catch( Throwable ex )
     {
ex.printStackTrace();
      }
     return null;
}

/**
* Action encapsulating the control method : Login
*
* @jpf:action
* @jpf:forward name="success" path="LoginResult.jsp"
* @jpf:forward name="failed" path="Login.jsp"
```

```
*/
public Forward Login( LoginForm aForm )
{
  try
  {
     int var = myControl.Login(aForm.user, aForm.password);

if( var == 0 )
{
    return new Forward("failed");
}
else
{
  getRequest().setAttribute( "results", new Integer ( var ) );
  clientid = var;
  aclient = myControl.DisplayAccount(clientid);

   return new Forward("success");
}
  }catch( Throwable ex )
   {
ex.printStackTrace();
    }
    return null;
}

/**
* Action encapsulating the control method : DisplayOrderHistory
*
* @jpf:action
* @jpf:forward name="success" path="DisplayOrderHistory.jsp"
* @jpf:forward name="failed"  path="DisplayOrderHistory.jsp"
*/
public Forward DisplayOrderHistory( DisplayOrderHistoryForm aForm )
{
  try
  {
orderHistory = myControl.DisplayOrderHistory(clientid);
int sizev = orderHistory.length;
orderH = new HistoryForm[sizev];

for( int i=0; i<sizev; i++)
{
    orderH[i] = new HistoryForm();
orderH[i].setPackagename(orderHistory[i].packagename );
orderH[i].setProvidername(orderHistory[i].providername );
orderH[i].setStatus(orderHistory[i].status);
orderH[i].setPrice(orderHistory[i].price );
orderH[i].setTempid( orderHistory[i].enddate );
}

if( orderHistory != null && orderHistory.length != 0 )
{
showOrderHistory = true;
aForm.setMessage("Order History");
return new Forward( "success", aForm);
}
else
{
showOrderHistory = true;
aForm.setMessage("No Order History");
return new Forward( "failed", aForm );
```

```
}
   }
  catch( Throwable ex )
  {
      ex.printStackTrace();
  }
   return null;
}


/**
* Action encapsulating the control method : UpdataClientAccount
*
* @jpf:action
* @jpf:forward name="success" path="checkoutpage.jsp"
*/
public Forward RemovePackCart( )
{
    String idstr = getRequest().getParameter( "packageid" );
   Integer idobj = new Integer(idstr);
   int id = idobj.intValue();

  int cartLeng = shopCart.length;

if( cartLeng > 1 )
{
    CustomerControls.CustomerControl1Control.packages[] pkArray =
    new  CustomerControls.CustomerControl1Control.packages[cartLeng - 1] ;

    // copy the contents of old shopping cart into the new shopping cart
   for( int j=0; j < cartLeng; j++)
   {
      if( j < id )
      {
pkArray[j] = new  CustomerControls.CustomerControl1Control.packages();

pkArray[j].packagename = shopCart[j].packagename;
pkArray[j].price = shopCart[j].price;
pkArray[j].providername = shopCart[j].providername;
pkArray[j].status = "active";

int featlengd = shopCart[j].featureList.length;
CustomerControls.CustomerControl1Control.feature[] featArrayd =
new  CustomerControls.CustomerControl1Control.feature[featlengd] ;

for( int k=0; k<featlengd; k++)
{
featArrayd[k] = new CustomerControls.CustomerControl1Control.feature();

featArrayd[k].featureid = shopCart[j].featureList[k].featureid;
featArrayd[k].featurename = shopCart[j].featureList[k].featurename;
featArrayd[k].option = shopCart[j].featureList[k].option;
featArrayd[k].price = shopCart[j].featureList[k].price;
}
        pkArray[j].featureList = featArrayd;
    }
    else if( j > id )
    {
pkArray[j-1] = new  CustomerControls.CustomerControl1Control.packages();

pkArray[j-1].packagename = shopCart[j].packagename;
pkArray[j-1].price = shopCart[j].price;
```

```
pkArray[j-1].providername = shopCart[j].providername;
pkArray[j-1].status = "active";

int featlengd = shopCart[j].featureList.length;
CustomerControls.CustomerControl1Control.feature[] featArrayd =
new  CustomerControls.CustomerControl1Control.feature[featlengd] ;

for( int k=0; k<featlengd; k++)
{
featArrayd[k] = new CustomerControls.CustomerControl1Control.feature();

featArrayd[k].featureid = shopCart[j].featureList[k].featureid;
featArrayd[k].featurename = shopCart[j].featureList[k].featurename;
featArrayd[k].option = shopCart[j].featureList[k].option;
          featArrayd[k].price = shopCart[j].featureList[k].price;
        }
       pkArray[j-1].featureList = featArrayd;
    }
  }
  // the new shopping cart is ready
  shopCart = pkArray;
}
  else
  shopCart = null;

  showCart = true;
  return new Forward("success");
}

/**
* Action encapsulating the control method : UpdataClientAccount
*
* @jpf:action
* @jpf:forward name="success" path="UpdateAccount.jsp"
*/
public Forward UpdataClientAccount( accountForm aForm )
{
 try
 {
CustomerControls.CustomerControl1Control.dclient ac
= new CustomerControls.CustomerControl1Control.dclient();

ac.uid = clientid;
ac.email = aForm.getEmail();
ac.pw = aForm.getPw();
ac.lname = aForm.getLast();
ac.fname = aForm.getFirst();
ac.address = aForm.getAddress();
ac.city = aForm.getCity();
ac.state = aForm.getState();
ac.zip = aForm.getZip();
ac.credit_card_num = aForm.credit_card_num;

myControl.UpdataClientAccount(ac);
aForm.setMessage("You have successfully updated your account!");

return new Forward( "success", aForm );

 }
 catch( Throwable ex )
 {
   ex.printStackTrace();
```

```
   }
   return null;
}

/**
* FormData class SearchServiceForm
*
*/
public static class SearchServiceForm extends FormData
{
private java.lang.String searchText;
private boolean pricesensitive;
private boolean servicesensitive;

public void setSearchText( java.lang.String searchText )
{
  this.searchText = searchText;
}

public java.lang.String getSearchText()
{
   return searchText;
}

public void setPricesensitive( boolean pricesensitive )
{
  this.pricesensitive = pricesensitive;
}

public boolean getPricesensitive()
{
  return pricesensitive;
}

public void setServicesensitive( boolean servicesensitive )
{
  this.servicesensitive = servicesensitive;
}

public boolean getServicesensitive()
{
   return servicesensitive;
}
}



/**
* FormData class PlaceOrderForm
*
*/
public static class PlaceOrderForm extends FormData
{
private int clientid;
private CustomerControls.CustomerControl1Control.dclient ac;
private CustomerControls.CustomerControl1Control.packages[] pckArray;

public void reset( org.apache.struts.action.ActionMapping mapping,
javax.servlet.http.HttpServletRequest request )
{
  // Todo - allocate the object <beanName> here.
  // <beanName> = new <BeanName>();
```

```
}

public void setClientid( int clientid )
{
  this.clientid = clientid;
}

public int getClientid()
{
  return clientid;
}

public void setAc( CustomerControls.CustomerControl1Control.dclient ac )
{
 this.ac = ac;
}

public CustomerControls.CustomerControl1Control.dclient getAc()
{
  return ac;
}

public void setPckArray( CustomerControls.CustomerControl1Control.packages[] pckArray )
{
   this.pckArray = pckArray;
}

public CustomerControls.CustomerControl1Control.packages[] getPckArray()
{
   return pckArray;
}
}


/**
* FormData class messageForm
*
*/
public static class messageForm extends FormData
{
private java.lang.String message;

public void setMessage( java.lang.String message )
{
  this.message = message;
}

public java.lang.String getMessage()
{
  return message;
}
}


/**
* FormData class messageForm
*
*/
public static class HistoryForm extends FormData
{
private java.lang.String packagename;
private java.lang.String providername;
```

```
private java.lang.String price;
private java.lang.String status;
private String tempid;


public void setPackagename( java.lang.String packagename )
{
  this.packagename = packagename;
}

public java.lang.String getPackagename()
{
  return packagename;
}

public void setProvidername( java.lang.String providername )
{
  this.providername = providername;
}

public java.lang.String getProvidername()
{
  return providername;
}

public void setStatus( java.lang.String status )
{
  this.status = status;
}

public java.lang.String getStatus()
{
  return status;
}

public void setPrice( double prc )
{
NumberFormat nf = NumberFormat.getNumberInstance();
nf.setMinimumFractionDigits(2);
nf.setMaximumFractionDigits(2);
String pricestr = nf.format( prc );
this.price = pricestr;
}

public java.lang.String getPrice()
{
  return price;
  }

public void setTempid( String tempid )
{
   this.tempid = tempid;
}

public String getTempid()
{
   return tempid;
}

}

/**
```

```
* FormData class featForm
*
*/
public static class FeatureForm extends FormData
{
private java.lang.String featurename;
private java.lang.String price;

public void setFeaturename( java.lang.String featurename )
{
  this.featurename = featurename;
}

public java.lang.String getFeaturename()
{
  return featurename;
}

public void setPrice( double price )
{
NumberFormat nf = NumberFormat.getNumberInstance();
nf.setMinimumFractionDigits(2);
nf.setMaximumFractionDigits(2);
String pricestr = nf.format(price);
this.price = pricestr;
}

public java.lang.String getPrice()
{
  return price;
}
}



/**
* FormData class CancelServiceForm
*
*/
public static class CancelServiceForm extends FormData
{
private int refnumber;
private CustomerControls.CustomerControl1Control.packages ap;

public void reset( org.apache.struts.action.ActionMapping mapping, javax.servlet.http.HttpServletRequest
request )
{
  // Todo - allocate the object <beanName> here.
  // <beanName> = new <BeanName>();
}

public void setRefnumber( int refnumber )
{
  this.refnumber = refnumber;
}

public int getRefnumber()
{
  return refnumber;
}
```

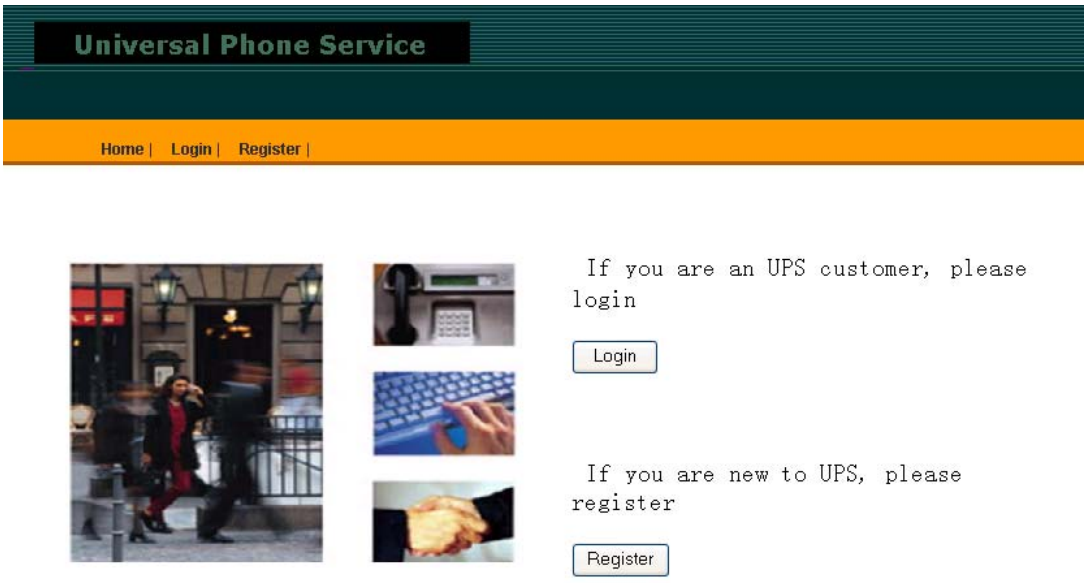
```
public void setAp( CustomerControls.CustomerControl1Control.packages ap )
{
  this.ap = ap;
}

public CustomerControls.CustomerControl1Control.packages getAp()
{
  return ap;
}
}

/**
* FormData class LoginForm
*
*/
public static class LoginForm extends FormData
{
private java.lang.String user;
private java.lang.String password;

public void setUser( java.lang.String user )
{
  this.user = user;
}

public java.lang.String getUser()
{
  return user;
}

public void setPassword( java.lang.String password )
{
   this.password = password;
}

public java.lang.String getPassword()
{
  return password;
}
}


/**
* FormData class DisplayOrderHistoryForm
*
*/
public static class DisplayOrderHistoryForm extends FormData
{
private int customID;
private String message;

public void setMessage( String message)
{
   this.message = message;
}

public String getMessage()
{
   return message;
}

public void setCustomID( int customID )
```

```
{
  this.customID = customID;
}

public int getCustomID()
{
  return customID;
}
}


/**
* FormData class SimpleSearchForm
*
*/
public static class accountForm extends FormData
{

public int uid;
public   String email;
public   String pw;
public   String lname;
public   String fname;
public   String address;
public   String city;
public   String state;
public   String zip;
public   String credit_card_num;
public   String message;

public void setMessage(String message) { this.message = message; }

public String getMessage() { return message; }

public void setUid( int id ) { this.uid = id; }

public int getUid() { return uid; }

public void setEmail(String em) { this.email = em; }

public String getEmail() { return email; }

public void setPw( String psw ) { this.pw = psw; }

public String getPw() { return pw; }

public void setLast( String last ) { this.lname = last; }

public String getLast() { return lname ; }

public void setFirst( String first ) { this.fname = first; }

public String getFirst() { return fname; }

public void setAddress( String addr ) { this.address = addr; }

public String getAddress() { return address; }

public void setCity( String ct ) { this.city = ct; }

public String getCity() { return city; }
```

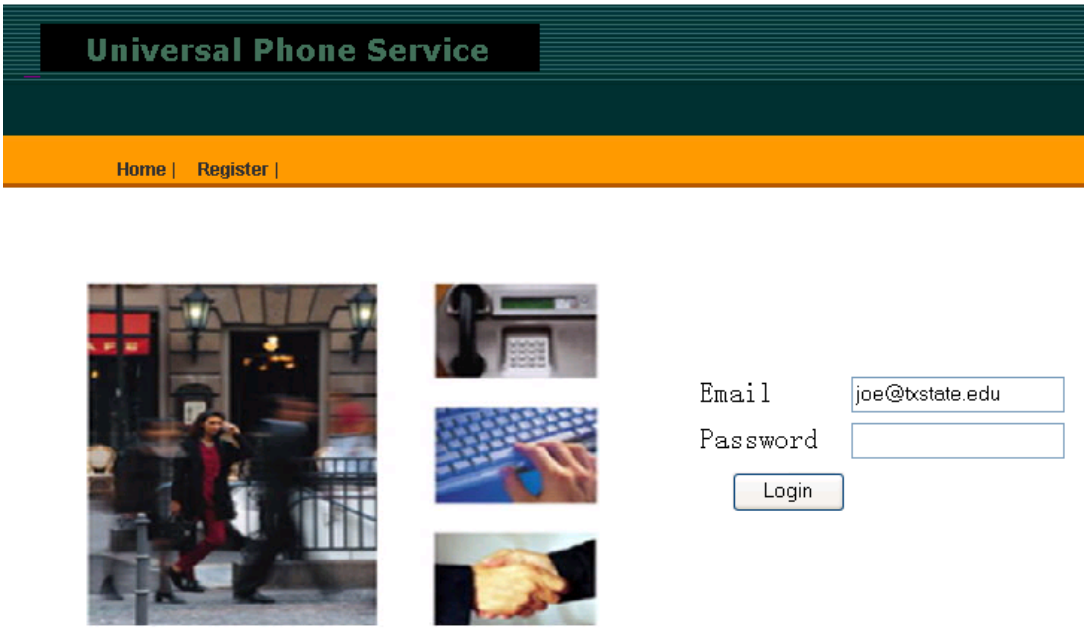
```
public void setState( String st ) { this.state = st; }

public String getState() { return state; }

public void setZip( String zp ) { this.zip = zp; }

public String getZip() { return zip; }

public void setCredit( String credit ) { this.credit_card_num = credit; }

public String getCredit() { return credit_card_num; }
}


/**
* @jpf:action
* @jpf:forward name="success" path="Login.jsp"
*/
protected Forward CustLoginPage()
{
  return new Forward("success");
}

/**
* @jpf:action
* @jpf:forward name="success" path="Register.jsp"
*/
protected Forward CustRegisterPage()
{
  return new Forward("success");

}


/**
* @jpf:action
* @jpf:forward name="success" path="SearchService.jsp"
*/
protected Forward GotoSearch()
{
  return new Forward("success");
}

/**
* @jpf:action
* @jpf:forward name="success" path="UpdateAccount.jsp"
* @jpf:forward name="failed" path="LoginResult.jsp"
*/
protected Forward GotoUpdateAccount()
{
try
{
CustomerControls.CustomerControl1Control.dclient acln
= myControl.DisplayAccount(clientid);
accountForm aForm = new accountForm();

aForm.setUid(acln.uid);
aForm.setEmail(acln.email);
aForm.setPw(acln.pw);
aForm.setLast(acln.lname);
aForm.setFirst(acln.fname);
aForm.setAddress(acln.address);
```

```
aForm.setCity(acln.city);
aForm.setState(acln.state);
aForm.setZip(acln.zip);
aForm.setCredit(acln.credit_card_num);

return new Forward( "success", aForm );

}
catch( Throwable ex )
{
  ex.printStackTrace();
}
  return new Forward("failed");
}
}
```

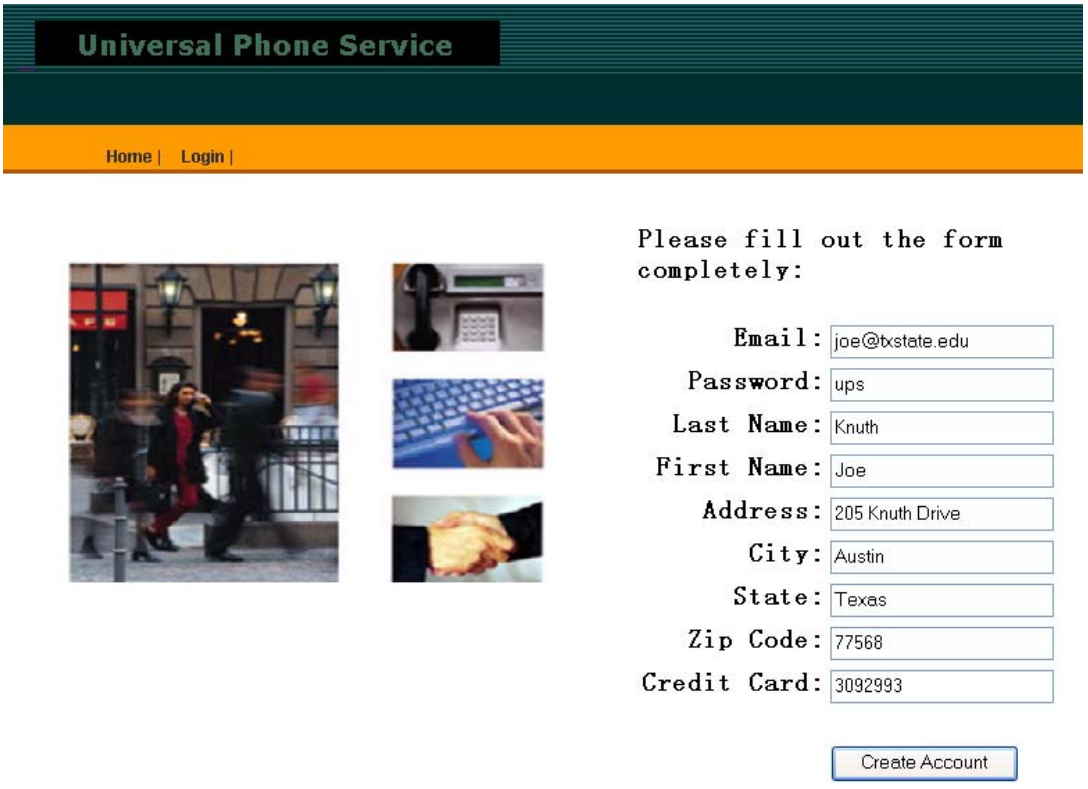
6. index.jsp



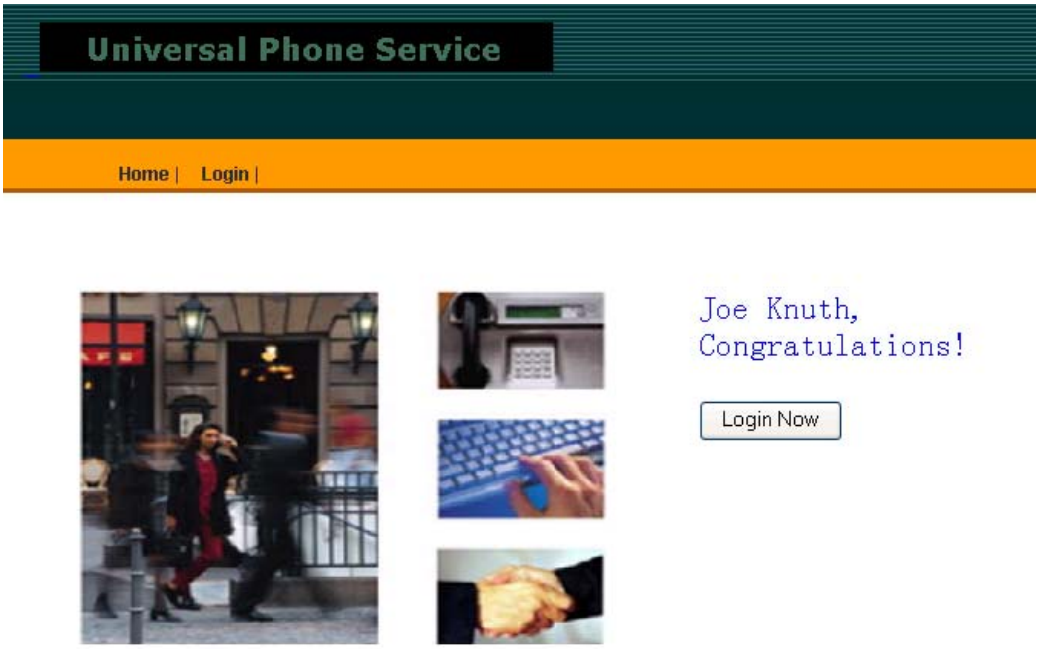
7. login.jsp



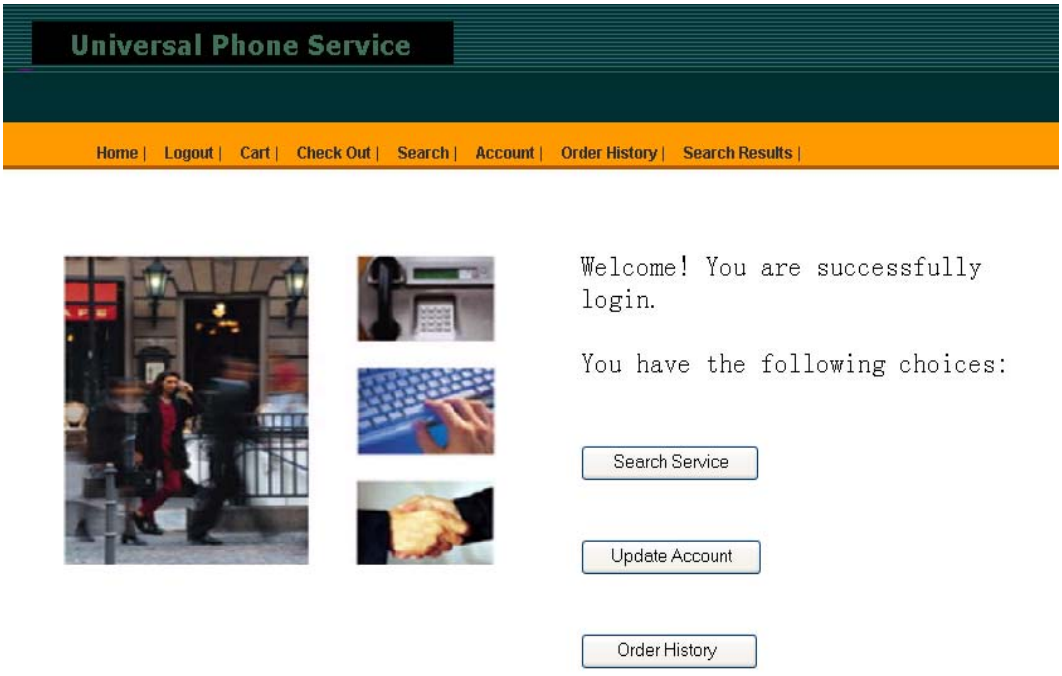
8.Register.jsp



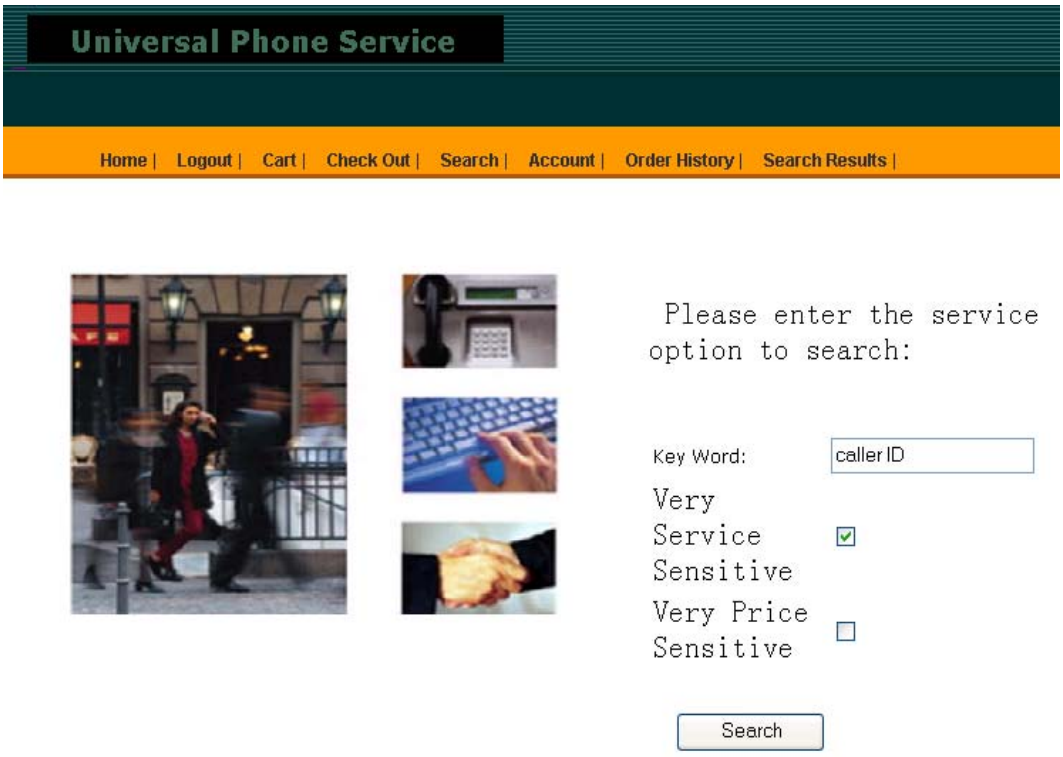
9. RegistryResult.jsp



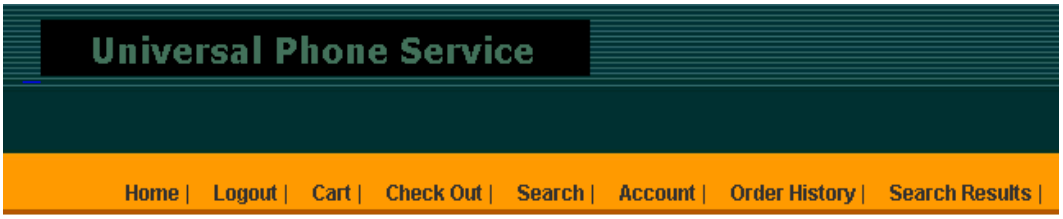
10. LoginResult.jsp



11. SearchService.jsp

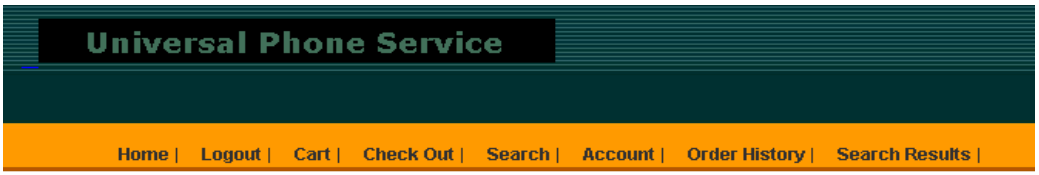


12. ShowSearchResult.jsp



Package	Provider	Features	Cart
Economic Phone ATT	ATT	Detail	Add
Homeline Plus ATT	ATT	Detail	Add
Homeline Select ATT	ATT	Detail	Add
Homeline ATT	ATT	Detail	Add

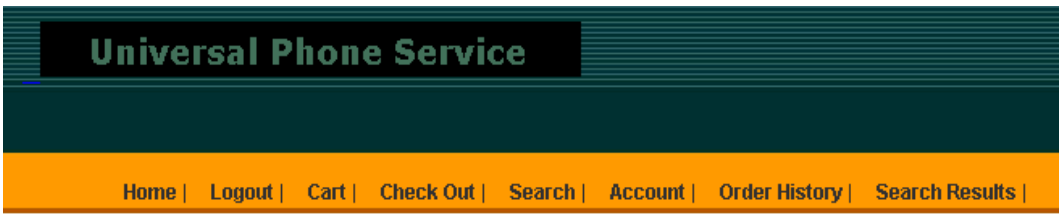
13. checkoutpage.jsp



Package	Provider	Features	Remove
Economic Phone ATT	ATT	Detail	Remove
Homeline Plus ATT	ATT	Detail	Remove

Place Order

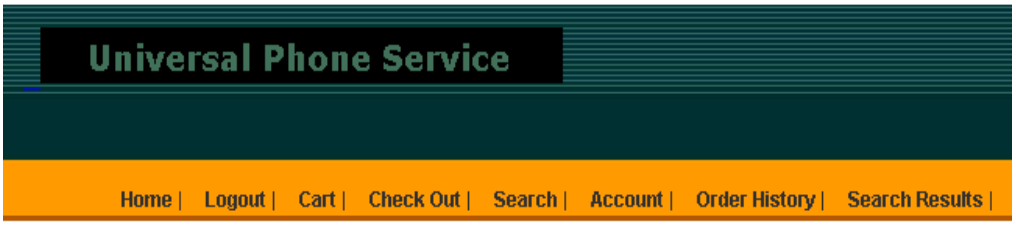
14. showFeature.jsp



Detailed Feature List

Feature	Price
speed dial	0.39
call waiting	1.99
caller ID	3.99
call return	1.39
call blocker	2.29

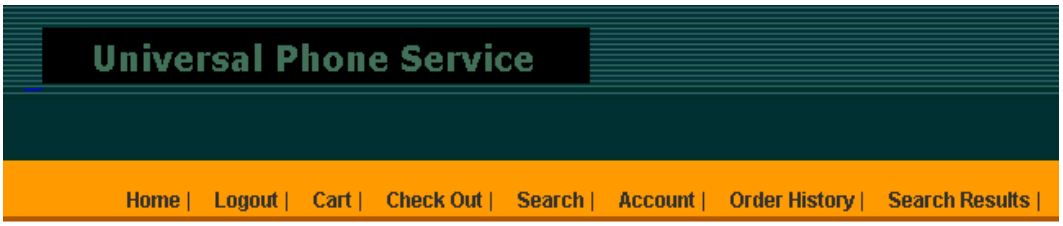
15. ShowSuccessOrder.jsp



You have successfully placed your order!

Package	Provider

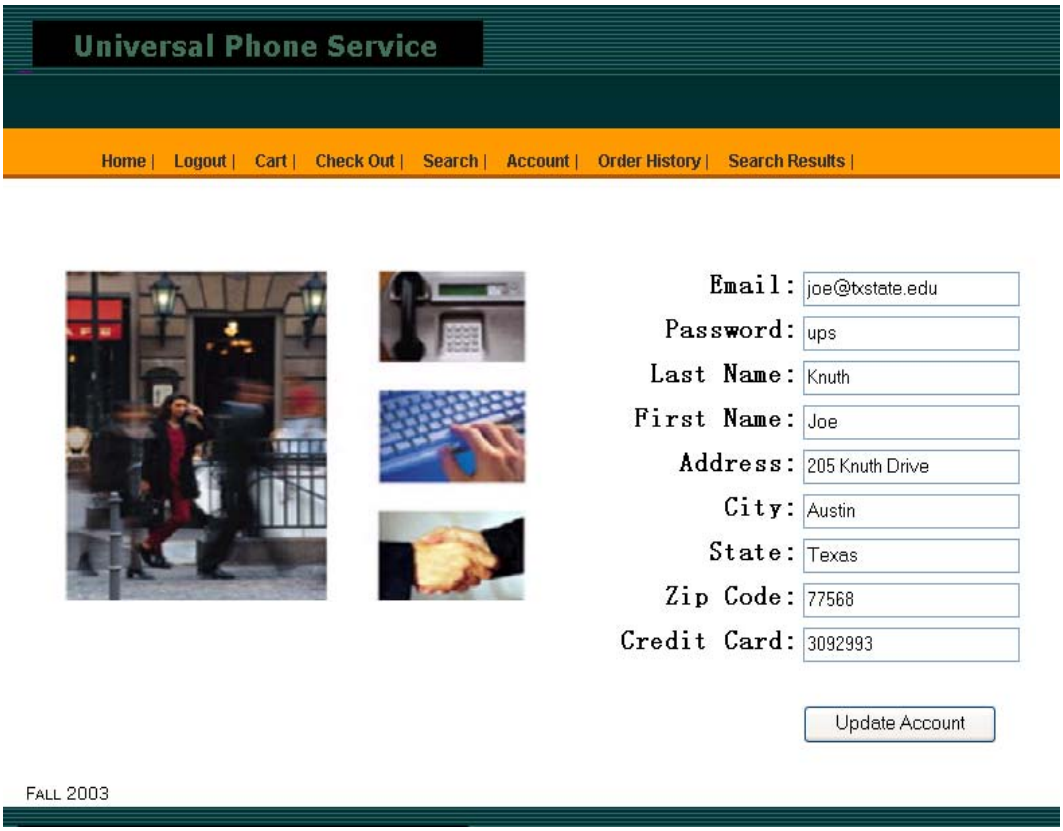
16. DisplayOrderHistory.jsp



Order History

Package	Provider	Price	Status	Remove
Economic Phone ATT	ATT	19.99	active	Remove
Homeline Plus ATT	ATT	39.99	active	Remove

17.UpdateAccount.jsp



18. dclient.java

```
package databaseJavaClass;

import java.io.Serializable;

public class dclient implements Serializable
{

public int uid;
public  String email;
public  String pw;
public  String lname;
public  String fname;
public  String address;
public  String city;
public   String state;
public   String zip;
public   String credit_card_num;


public dclient() {}

public dclient(String Email,String Pw,String Lname,String Fname,String Address,String City,String
    State,String Zip,String Credit_card_num)
```

```
{
email = new String(Email);
pw = new String(Pw);
lname = new String(Lname);
fname = new String(Fname);
address = new String(Address);
city = new String(City);
state = new String(State);
zip = new String(Zip);
credit_card_num = Credit_card_num;
}
}
```

19.feature.java

```
package databaseJavaClass;

import java.io.Serializable;

public class feature implements Serializable
{

        public int featureid;
        public String featurename;
        public boolean option;
        public double price;

        public feature() {}

        public feature(int f,String name,boolean option, double price )
        {
                featureid = f;
                featurename=name;
                this.option = option;
                this.price = price;
        }


         public int getFeatureid(){ return featureid;}
         public String getFeaturename() { return featurename;}
         public boolean getOption() { return option;}
         public double getPrice() { return price;}
         public void setPrice( double price){this.price=price;}
         public void setOption( boolean option) { this.option=option;}
         public void setFeatureid(int i) { featureid=i;}
         public void setFeaturename(String n) {featurename=n;}
}
```

20. order.java

```
package databaseJavaClass;

import java.io.Serializable;

public class order implements Serializable
{

    public int orderid;
    public int clientid;
```

```
    public String active;
    public String orderdate;
    public packages[] packagesList;

    public order() {}

public order(int cid,String act,String d)
{
   clientid=cid;
   active=act;
   orderdate=d;
}

public int getOrderid() { return orderid;}
public int getClientid() { return clientid;}
public String getActive() { return active;}
public String getOrderdata() { return orderdate;}

public void setOrderid(int i) { orderid=i;}
public void setClientid(int i) { clientid=i;}
public void setActive(String a) { active=a;}
public void setOrderdate(String d) {orderdate=d;}

}
```

21. package.java

```
package databaseJavaClass;

import java.io.Serializable;

public class packages implements Serializable
{

public int packageid;
public String packagename;
public String providername;
public double price;
public String startdate;
public String enddate;
public String status;
public int serviceid;
public int servicepwd;
public feature[] featureList;

public packages() {}


public packages(String name, String pvname, double p, String s,
                        int pserviceid, int pservicepwd, String e,String sta)
{
packagename=name;
providername=pvname;
price=p;
startdate=s;
enddate=e;
serviceid = pserviceid;
servicepwd = pservicepwd;
status=sta;
      }
public int getPackageid() { return packageid;}
```

```
public String getPackagename() { return packagename;}
public String getProvidername() { return providername;}
public double getPrice() { return price;}
public String getStartdate() { return startdate;}
public String getEnddate() { return enddate;}
public String getStatus() {return status;}
public int getServiceid() { return serviceid;}
public int getServicepwd() { return servicepwd;}

public void setPackageid(int i){ packageid=i;}
public void setPackagename(String n){ packagename=n;}
public void setProvidername(String pvname) { providername=pvname;}
public void setPrice(double p) { price=p;}
public void setStartdate(String s) { startdate=s;}
public void setEnddate(String e) { enddate=e;}
public void setStatus(String t) { status=t;}
public void setServiceid( int d ) {serviceid =d;}
public void setServicepwd( int p ) { servicepwd = p;}
}
```

22 provider.java

```
package databaseJavaClass;

import java.io.Serializable;

public class provider implements Serializable
{
public int providerid;
public String providername;
public String status;
public String selfpw;
public String address;
public String email;
public String url;
public String comtype;
public String accessproviderusername;
public String accessproviderpw;
public provider() {}

public provider(String pname, String pstatus,String pselfpw,String padd,String pemail,String purl,String
pcomtype,String paname,String papw)
{
providername=pname;
status=pstatus;
selfpw=pselfpw;
address=padd;
email=pemail;
url=purl;
comtype=pcomtype;
accessproviderusername=paname;
accessproviderpw=papw;
}

public int getProviderid() { return providerid;}
public String getProvidername() { return providername;}
public String getStatus() { return status;}
public String getSelfpw() { return selfpw;}
public String getAddress() { return address;}
public String getEmail() { return email;}
public String getUrl() { return url;}
```

```
public String getComtype() { return comtype;}
public String getAccessproviderusername() { return accessproviderusername;}
public String getAccessproviderpw() { return accessproviderpw;}

public void setProviderid(int i) { providerid=i;}
public void setProvidername(String n) {  providername=n;}
public void setStatus(String s) {  status=s;}
public void setSelfpw(String self) {  selfpw=self;}
public void setAddress(String a) { address=a;}
public void setEmail(String e) { email=e;}
public void setUrl(String u) { url=u;}
public void setComtype(String d) { comtype=d;}
public void setAccessproviderusername(String n) { accessproviderusername=n;}
public void setAccessproviderpw(String p) { accessproviderpw=p;}
}
```

23 Order.xsd

```
<?xml version="1.0"?>
<xs:schema targetNamespace="http://openuri.org/bea/samples/workshop"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ws="http://openuri.org/bea/samples/workshop"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
   <xs:element name="order-all-data">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="order" type="ws:orderType" use="required"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:complexType name="CustomerType">
      <xs:sequence>
         <xs:element name="firstname" type="xs:string"/>
         <xs:element name="lastname"  type="xs:string"/>
         <xs:element name="email"     type="xs:string"/>
         <xs:element name="address"   type="xs:string"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:int" use="required"/>
   </xs:complexType>
   <xs:complexType name="orderType">
      <xs:sequence>
         <xs:element name="package" type="ws:PackageType" maxOccurs="unbounded"/>
         <xs:element name="customer" type="ws:CustomerType" use="required" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:int" use="required"/>
   </xs:complexType>
</xs:schema>
```

24. Package.xsd

```
<?xml version="1.0"?>
<xs:schema targetNamespace="http://openuri.org/bea/samples/workshop"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ws="http://openuri.org/bea/samples/workshop"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
   <xs:element name="package-all-data">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="package" type="ws:PackageType" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
```

```
   </xs:element>
   <xs:complexType name="FeatureType">
      <xs:sequence>

         <xs:element name="option" type="xs:boolean"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
      <xs:attribute name="price" type="xs:float" use="required"/>
   </xs:complexType>
   <xs:complexType name="PackageType">
      <xs:sequence>
         <xs:element name="feature" type="ws:FeatureType" maxOccurs="unbounded"/>
         <xs:element name="provider" type="ws:ProviderType" use="required" />
         <xs:element name="price" type="xs:float" use="required" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
   </xs:complexType>
</xs:schema>
```

25. Provider.xsd

```
<?xml version="1.0"?>
<xs:schema targetNamespace="http://openuri.org/bea/samples/workshop"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ws="http://openuri.org/bea/samples/workshop"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
   <xs:element name="provider-all-data">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="provider" type="ws:ProviderType" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:complexType name="ProviderType">
      <xs:sequence>
         <xs:element name="name" type="xs:string"/>
         <xs:element name="email"     type="xs:string"/>
         <xs:element name="address"   type="xs:string"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
   </xs:complexType>

</xs:schema>
```

APPENDIX III

PROGRAMMING SOURCE CODE OF SERVICE PROVIDERS

The following material is the Java source code of the implementation of service providers which is created in BEA Weblogic Workshop.

A. Business Logics

1. BusinessProcessor.jws
2. myDatabaseControl.jcx
3. AdminAPageFlowController.jpf
4. CreditAgent.jws

B. Web Pages

5. LoginResult.jsp
6. index.jsp
7. RemoveServiceFromUPS.jsp
8. ShowAllOrders.jsp
9. ShowAllPackages.jsp

1. BusinessProcessor.jws

```
package BusinessProcessorWebService;

import weblogic.jws.control.JwsContext;
import org.openuri.bea.samples.workshop.PackageType;
import org.openuri.bea.samples.workshop.FeatureType;
import org.openuri.bea.samples.workshop.ProviderType;
import org.openuri.bea.samples.workshop.OrderType;
import org.openuri.bea.samples.workshop.CustomerType;
import org.openuri.bea.samples.workshop.SearchResultData;
import org.openuri.bea.samples.workshop.SearchResultDataDocument;
import org.openuri.bea.samples.workshop.OrderAllData;
import org.openuri.bea.samples.workshop.OrderAllDataDocument;
import org.openuri.bea.samples.workshop.CancelOrderData;
import org.openuri.bea.samples.workshop.CancelOrderDataDocument;
import org.openuri.bea.samples.workshop.ConfirmOrderData;
import org.openuri.bea.samples.workshop.ConfirmOrderDataDocument;
import org.openuri.bea.samples.workshop.PackageAllData;
import org.openuri.bea.samples.workshop.PackageAllDataDocument;
import org.openuri.bea.samples.workshop.ProviderAllData;
import org.openuri.bea.samples.workshop.ProviderAllDataDocument;
import org.openuri.bea.samples.workshop.TestData;
import org.openuri.bea.samples.workshop.TestDataDocument;
import org.openuri.bea.samples.workshop.RemoveServiceData;
import org.openuri.bea.samples.workshop.RemoveServiceDataDocument;

import com.bea.xml.XmlLoader;
import java.util.Iterator;
import java.io.Serializable;
import com.bea.ide.*;
import java.util.HashMap;
import com.bea.xml.XmlOptions;
import org.openuri.bea.samples.workshop.SearchResultData;
import java.util.Vector;
import JavaClass.*;

public class BusinessProcessor
{
   /**
    * @common:control
    */
   private BusinessProcessorWebService.myDatabaseControl myDatabase;

   /** @common:context */
   JwsContext context;

   /**
    * @common:operation
    * @jws:protocol soap-style="rpc"
    */
   public String SearchService(String user, String password, String searchText)
   {
      int[] packageIdArray = myDatabase.findPackagIdByFeature(searchText);

      PackageAllData ResultDataXML;
      ProviderType provider;

      try
      {
         ResultDataXML = PackageAllData.Factory.newInstance();

         //add all the packages
```

```
        for( int j=0; j<packageIdArray.length; j++)
        {
          PackageType pkg = ResultDataXML.addNewPackage();
          ProviderType prov = pkg.addNewProvider();
          prov.setName("SBC");
          prov.setEmail("sbc@sbc.com");
          prov.setAddress("2003 SBC Street,Austin");
          prov.setId("SBC");

          java.sql.ResultSet res =
            myDatabase.showAllFeatureWithPackageId(packageIdArray[j]);
          boolean firstEntered = false;
          //add all the features into package
          while( res.next() )
          {
            FeatureType a = pkg.addNewFeature();

            if( firstEntered == false )
            {
              pkg.setId( res.getString("packagename") );
              pkg.setPrice( (float)res.getFloat("price") );
              firstEntered = true;
            }
            a.setId(res.getString("featurename"));
            a.setOption(res.getBoolean("fea_option"));
            a.setPrice((float) res.getFloat("feaprice"));
          }
        }
          return ResultDataXML.xmlText();

    }catch( Exception e){ return null;}

}


 /**
  * @common:operation
  * @jws:protocol soap-style="rpc"
  */
 public void PlaceOrder(String user, String password, String orderXML )
 {
    try
    {
       OrderAllData orderInfo = OrderAllData.Factory.parse(orderXML);
       OrderType oneOrder = orderInfo.getOrder();
       CustomerType Customer = oneOrder.getCustomer();
       PackageType[] packageArray = oneOrder.getPackageArray();

       // add client information
       dclient ac = new dclient();
       ac.email = Customer.getEmail();
       ac.address = Customer.getAddress();
       ac.fname = Customer.getFirstname();
       ac.lname =  Customer.getLastname();

       int id;
       id = myDatabase.findClientByEmail(ac.email);
       if( id == 0 ) // add a new client
       {
          myDatabase.addClientAccount(ac);
          id = myDatabase.findNewClientId();
       }
```

```
        // used an ups portal site's order id as a reference number
        // for each order, for future cancelling
        myDatabase.addEntryInOrderTable( id, oneOrder.getId() );
        int id2 = myDatabase.findNewIdInOrderTable();

        // update database table
        for( int j=0; j<packageArray.length; j++)
        {

            // Add entry for each package
            myDatabase.addEntryInOrderPackgeTable(id2, packageArray[j].getId());

            FeatureType[] featureArray = packageArray[j].getFeatureArray();

            for( int k=0; k<featureArray.length; k++)
            {
               // Add Entry for each feature
               myDatabase.addEntryInOrderPkgFeatureTable(id2,
               packageArray[j].getId(),featureArray[k].getId());
             }
         }

    }catch( Exception e )
    {
      return;
    }
    return;
}



/**
 * @common:operation
 * @jws:protocol soap-style="rpc"
 */
public String Test( String user, String password )
{
    TestData testDataXML;
    ProviderType provider;
    try
    {
        testDataXML=TestData.Factory.newInstance();
        provider = testDataXML.addNewProvider();

        provider.setName("SBC");
        provider.setEmail("sbc@sbc.com");
        provider.setAddress("2003 SBC Rd. Austin");
        provider.setId("SBC");

        return testDataXML.xmlText();

    }catch( java.lang.IllegalArgumentException iae )
    {
        return "failed";
    }
}


/**
 * @common:operation
 * @jws:protocol soap-style="rpc"
```

```
     */
    //a reference num of the order from UPS to remove this order
    public boolean CancelService(String user, String password, String packagename, int refno)
    {
        if( user.equals("ups") && password.equals("ups") )
        {
          int orderid = myDatabase.findOrderIdByRefNo(refno);
          myDatabase.cancelOnePackageByClient(orderid, packagename);

          return true;
        }
        else
          return false;
    }
}
```

2. myDatabaseControl.jcs

```
package BusinessProcessorWebService;

import weblogic.jws.control.*;
import java.sql.SQLException;
import JavaClass.*;
/**
 * Defines a new database control.
 *
 * The @jc:connection tag indicates which WebLogic data source will be used by
 * this database control. Please change this to suit your needs. You can see a
 * list of available data sources by going to the WebLogic console in a browser
 * (typically http://localhost:7001/console) and clicking Services, JDBC,
 * Data Sources.
 *
 * @jc:connection data-source-jndi-name="UPSDataSource"
 */
public interface myDatabaseControl extends DatabaseControl
{
    // Sample database function.  Uncomment to use

    // static public class Customer
    // {
    //   public int id;
    //   public String name;
    // }
    //
    // /**
    //  * @jc:sql statement="SELECT ID, NAME FROM CUSTOMERS WHERE ID = {id}"
    //  */
    // Customer findCustomer(int id);

    // Add "throws SQLException" to request that SQLExeptions be thrown on errors.
     /**
     * @jc:sql statement::
     * SELECT SBC_PACKAGE.PACKAGEID, PACKAGENAME, PRICE,
SBC_FEATURE.FEATURENAME, FEA_OPTION, FEAPRICE
     * FROM SBC_PACKAGE, SBC_PACKFEATURE, SBC_FEATURE
     * WHERE STATUS = 'active'
     * AND SBC_PACKAGE.PACKAGEID = SBC_PACKFEATURE.PACKAGEID
     * AND SBC_PACKFEATURE.FEATURENAME = SBC_FEATURE.FEATURENAME::

     */
    java.sql.ResultSet showAllPackages();
```

```
    /**
     * @jc:sql statement::
     * SELECT SBC_PACKAGE.PACKAGEID, PACKAGENAME, PRICE,
SBC_FEATURE.FEATURENAME, FEA_OPTION, FEAPRICE
     * FROM SBC_PACKAGE, SBC_PACKFEATURE, SBC_FEATURE
     * WHERE STATUS = 'active'
     * AND SBC_PACKAGE.PACKAGEID = SBC_PACKFEATURE.PACKAGEID
     * AND SBC_PACKFEATURE.FEATURENAME = SBC_FEATURE.FEATURENAME
     * AND SBC_FEATURE.FEATURENAME = {featurename}::

     */
    java.sql.ResultSet searchPackageWithFeature(String featurename);

    /**
     * @jc:sql statement::
     * UPDATE SBC_PACKAGE SET STATUS = 'non_active'
     * WHERE PACKAGENAME={packagename}::

     */
    void removeOnePackage(String packagename);

    /**
     * @jc:sql statement::
     * UPDATE SBC_ORDER SET ORD_STATUS = 'cancelled'
     *  WHERE ORDERID = {refno}::

     */
    void removeOneOrder(int refno);

    /**
     * @jc:sql statement::
     * SELECT SBC_ORDER.ORDERID,  PKG_STATUS, REFNO, FNAME, LNAME, PACKAGENAME
     * FROM SBC_CLIENT, SBC_ORDER, SBC_ORDPACKAGE
     * WHERE CLIENTID = UID
     * AND SBC_ORDER.ORDERID = SBC_ORDPACKAGE.ORDERID::

     */
    java.sql.ResultSet showAllOrders();

    /**
     * @jc:sql statement="  INSERT INTO SBC_ORDER VALUES({clientid}, 'active', CURRENT_DATE,
{refno})"
     */
    void addEntryInOrderTable(int clientid, int refno);

    /**
     * @jc:sql statement="SELECT MAX(ORDERID) FROM SBC_ORDER"
     */
    int findNewIdInOrderTable();

    /**
     * @jc:sql statement="INSERT INTO SBC_ORDPACKAGE VALUES ({orderid}, {packagename},
'active')"
     */
    void addEntryInOrderPackgeTable(int orderid, String packagename);

    /**
     * @jc:sql statement="INSERT INTO SBC_ORDPKGFEATURE VALUES({orderid}, {packagename},
{featurename},'active')"
     */
    void addEntryInOrderPkgFeatureTable(int orderid, String packagename, String featurename);
```

```
  /**
   * @jc:sql statement::
   * INSERT INTO SBC_CLIENT (EMAIL,PW,LNAME,FNAME,ADDRESS,CITY,STATE,
ZIP,CREDIT_CARD_NUM)
   *
VALUES({ac.email},{ac.pw},{ac.lname},{ac.fname},{ac.address},{ac.city},{ac.state},{ac.zip},{ac.credit_card
_num})::

   */
  void addClientAccount(dclient ac );


  /**
   * @jc:sql statement::
   * SELECT UID FROM SBC_CLIENT WHERE
   * EMAIL={email}::

   */
  int findClientByEmail(String email);

  /**
   * @jc:sql statement="SELECT MAX(UID) FROM SBC_CLIENT"
   */
  int findNewClientId();

  /**
   * @jc:sql statement::
   * SELECT SBC_PACKAGE.PACKAGEID
   * FROM SBC_PACKAGE, SBC_PACKFEATURE, SBC_FEATURE
   * WHERE STATUS = 'active'
   * AND SBC_PACKAGE.PACKAGEID = SBC_PACKFEATURE.PACKAGEID
   * AND SBC_PACKFEATURE.FEATURENAME = SBC_FEATURE.FEATURENAME
   * AND SBC_FEATURE.FEATURENAME ={featurename}::

   */
  int[] findPackagIdByFeature(String featurename);

  /**
   * @jc:sql statement::
   * SELECT SBC_PACKAGE.PACKAGEID, PACKAGENAME, PRICE,
SBC_FEATURE.FEATURENAME, FEA_OPTION, FEAPRICE
   * FROM SBC_PACKAGE, SBC_PACKFEATURE, SBC_FEATURE
   * WHERE STATUS = 'active'
   * AND SBC_PACKAGE.PACKAGEID = SBC_PACKFEATURE.PACKAGEID
   * AND SBC_PACKFEATURE.FEATURENAME=SBC_FEATURE.FEATURENAME
   * AND SBC_PACKAGE.PACKAGEID={packageid}::

   */
  java.sql.ResultSet showAllFeatureWithPackageId(int packageid );

  /**
   * @jc:sql statement::
   * SELECT SBC_PACKAGE.PACKAGEID, PACKAGENAME, PRICE
   * FROM SBC_PACKAGE
   * WHERE STATUS = 'active'::

   */
  java.sql.ResultSet showAllPackageSimple();

  /**
   * @jc:sql statement="INSERT INTO SBC_PACKAGE VALUES({packagename},{price},'active')"
```

```
     */
    void addEntryOfPackageTable( String packagename, double price);

    /**
     * @jc:sql statement="SELECT MAX(PACKAGEID) FROM SBC_PACKAGE"
     */
    int findNewPackageId();

    /**
     * @jc:sql statement="INSERT INTO SBC_PACKFEATURE
VALUES({packageid},{featurename},{fea_option})"

     */
    void addEntryInPackFeatureTable(int packageid,String featurename,boolean fea_option);

    /**
     * @jc:sql statement="SELECT ORDERID FROM SBC_ORDER WHERE REFNO={refno}"
     */
    int findOrderIdByRefNo(int refno );

    /**
     * @jc:sql statement::
     * UPDATE SBC_ORDPACKAGE SET PKG_STATUS='cancelled'
     * WHERE ORDERID={orderid} AND PACKAGENAME={packagename}                                    ::

     */
    void cancelOnePackageByClient(int orderid, String packagename );

}
```

3. AdminAPageFlowController.jpf

```
// ----------------------------------------------------------------------------
//  Generated by Weblogic Workshop
//
//  Created By: Yutu
// ----------------------------------------------------------------------------
package AdminAPageFlow;

import com.bea.wlw.netui.pageflow.FormData;
import com.bea.wlw.netui.pageflow.Forward;

import AdminWebControl.AdminAWebService1Control;

/**
 * PageFlow class generated from control AdminAWebService1Control
 *
 */
public class AdminAPageFlowController extends com.bea.wlw.netui.pageflow.PageFlowController
{
    /**
     *  This is the control used to generate this pageflow
     *  @common:control
     */
    private AdminAWebService1Control myControl;
    private AdminWebControl.AdminAWebService1Control.simPackage[] simPkg;
    private AdminWebControl.AdminAWebService1Control.OrderForm[] orders;
```

```
 /**
 * Getter for databinding.
 */
public AdminWebControl.AdminAWebService1Control.simPackage[] getSimPackage()
{
   return simPkg;
}

 /**
 * Getter for databinding.
 */
public AdminWebControl.AdminAWebService1Control.OrderForm[] getOrderForm()
{
   return orders;
}


//     Uncomment this declaration to access Global.app.
//
//     protected global.Global globalApp;
//


/**
 * This method represents the point of entry into the pageflow
 *
 * @jpf:action
 * @jpf:forward name="success" path="index.jsp"
 */
protected Forward begin()
{
    return new Forward( "success" );
}

/**
 * This method represents the point of entry into the pageflow
 *
 * @jpf:action
 * @jpf:forward name="success" path="index.jsp"
 */
protected Forward Logout()
{
    return new Forward( "success" );
}

  /**
 * This method represents the point of entry into the pageflow
 *
 * @jpf:action
 * @jpf:forward name="success" path="LoginResult.jsp"
 */
protected Forward HomePage()
{
    return new Forward( "success" );
}

/**
 * Action encapsulating the control method : ShowAllOrders
 *
 * @jpf:action
 * @jpf:forward name="success" path="ShowAllOrders.jsp"
 * @jpf:forward name="failed" path="LoginResult.jsp"
```

```
 */
public Forward ShowAllOrders()
{
    try
    {
        orders = myControl.ShowAllOrders();

        return new Forward( "success" );

    }
    catch( Throwable ex )
    {
        ex.printStackTrace();
    }
    return null;
}


/**
 * Action encapsulating the control method : AddNewPackageByAdmin
 *
 * @jpf:action
 * @jpf:forward name="success" path="LoginResult.jsp"
 */
public Forward AddNewPackageByAdmin( AddNewPackageByAdminForm aForm )
{
    try
    {
        myControl.AddNewPackageByAdmin(aForm.packagename, aForm.price, aForm.featurename1,
                aForm.option1, aForm.featurename2, aForm.option2);

        return new Forward( "success" );

    }
    catch( Throwable ex )
    {
        ex.printStackTrace();
    }
    return null;
}

/**
 * Action encapsulating the control method : ShowAllPackages
 *
 * @jpf:action
 * @jpf:forward name="success" path="ShowAllPackages.jsp"
 * @jpf:forward name="failed" path="LoginResult.jsp"
 */
public Forward ShowAllPackages()
{
    try
    {
        simPkg = myControl.ShowAllPackages();
        return new Forward( "success" );

    }
    catch( Throwable ex )
    {
        ex.printStackTrace();
    }
    return null;
}
```

```
/**
 * Action encapsulating the control method : RemovePackage
 *
 * @jpf:action
 * @jpf:forward name="success"  path="LoginResult.jsp"
 * @jpf:forward name="failed" path="LoginResult.jsp"
 */
public Forward RemovePackage( RemovePackageForm aForm )
{
   //get the id of the removed package
   String pkname = getRequest().getParameter( "packagename" );
   int pkid = -1;

   try
   {
      for( int i=0; i<simPkg.length; i++)
      {
        if( (simPkg[i].packagename).equals(pkname) )
        {
           pkid = simPkg[i].packageid;
           break;
        }
      }

      if( pkname == null && pkid == -1 )
        return new Forward("failed");

      boolean var = myControl.RemovePackage(pkid, pkname);

      return new Forward( "success" );

   }
   catch( Throwable ex )
   {
      ex.printStackTrace();
   }
   return null;
}

/**
 * Action encapsulating the control method : RemoveServiceFromUPS
 *
 * @jpf:action
 * @jpf:forward name="success" path="RemoveServiceFromUPS.jsp"
  * @jpf:forward name="failed" path="LoginResult.jsp"
 */
public Forward RemoveServiceFromUPS( RemoveServiceFromUPSForm aForm )
{
   try
   {
      boolean var = myControl.RemoveServiceFromUPS(aForm.providerName, aForm.user,
                          aForm.password, aForm.url);
      getRequest().setAttribute( "results", new Boolean ( var ) );

      return new Forward( "success" );

   }
   catch( Throwable ex )
   {
```

```
        ex.printStackTrace();
    }
    return null;
}

/**
 * Action encapsulating the control method : Login
 *
 * @jpf:action
 * @jpf:forward name="success" path="LoginResult.jsp"
 * @jpf:forward name="failed" path="index.jsp"
 */
public Forward Login( LoginForm aForm )
{
    try
    {
        boolean var = myControl.Login(aForm.user, aForm.password);
        getRequest().setAttribute( "results", new Boolean ( var ) );

        if( var == true )
          return new Forward( "success" );
        else
        {
          aForm.setMessage("Login failed, Please try again.");
          return new Forward("failed", aForm);
        }


    }
    catch( Throwable ex )
    {
        ex.printStackTrace();
    }
    return null;
}

/**
 * FormData class AddNewPackageByAdminForm
 *
 */
public static class AddNewPackageByAdminForm extends FormData
{
    private java.lang.String packagename;
    private double price;
    private java.lang.String featurename1;
    private boolean option1;
    private java.lang.String featurename2;
    private boolean option2;

    public void setPackagename( java.lang.String packagename )
    {
        this.packagename = packagename;
    }

    public java.lang.String getPackagename()
    {
        return packagename;
    }

    public void setPrice( double price )
    {
        this.price = price;
```

```
    }

    public double getPrice()
    {
        return price;
    }

    public void setFeaturename1( java.lang.String featurename1 )
    {
        this.featurename1 = featurename1;
    }

    public java.lang.String getFeaturename1()
    {
        return featurename1;
    }

    public void setOption1( boolean option1 )
    {
        this.option1 = option1;
    }

    public boolean getOption1()
    {
        return option1;
    }

    public void setFeaturename2( java.lang.String featurename2 )
    {
        this.featurename2 = featurename2;
    }

    public java.lang.String getFeaturename2()
    {
        return featurename2;
    }

    public void setOption2( boolean option2 )
    {
        this.option2 = option2;
    }

    public boolean getOption2()
    {
        return option2;
    }
}

/**
 * FormData class RemovePackageForm
 *
 */
public static class RemovePackageForm extends FormData
{
    private int orderid;
    private java.lang.String packagename;

    public void setOrderid( int orderid )
    {
        this.orderid = orderid;
    }
```

```
    public int getOrderid()
    {
        return orderid;
    }

    public void setPackagename( java.lang.String packagename )
    {
        this.packagename = packagename;
    }

    public java.lang.String getPackagename()
    {
        return packagename;
    }
}

/**
 * FormData class RemoveServiceFromUPSForm
 *
 */
public static class RemoveServiceFromUPSForm extends FormData
{
    private java.lang.String providerName;
    private java.lang.String user;
    private java.lang.String password;
    private java.lang.String url;

    public void setProviderName( java.lang.String providerName )
    {
        this.providerName = providerName;
    }

    public java.lang.String getProviderName()
    {
        return providerName;
    }

    public void setUser( java.lang.String user )
    {
        this.user = user;
    }

    public java.lang.String getUser()
    {
        return user;
    }

    public void setPassword( java.lang.String password )
    {
        this.password = password;
    }

    public java.lang.String getPassword()
    {
        return password;
    }

    public void setUrl( java.lang.String url )
    {
        this.url = url;
    }
```

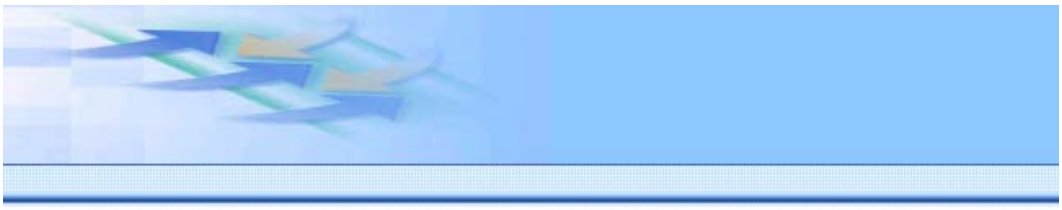
```
    public java.lang.String getUrl()
    {
        return url;
    }
}

/**
 * FormData class LoginForm
 *
 */
public static class LoginForm extends FormData
{
    private java.lang.String user;
    private java.lang.String password;
    private java.lang.String message;

    public void setMessage( java.lang.String message )
    {
        this.message = message;
    }

    public java.lang.String getMessage()
    {
        return message;
    }

    public void setUser( java.lang.String user )
    {
        this.user = user;
    }

    public java.lang.String getUser()
    {
        return user;
    }

    public void setPassword( java.lang.String password )
    {
        this.password = password;
    }

    public java.lang.String getPassword()
    {
        return password;
    }
}

/**
 * FormData class messageForm
 *
 */
public static class messageForm extends FormData
{
    private java.lang.String message;

    public void setMessage( java.lang.String message )
    {
        this.message = message;
    }

    public java.lang.String getMessage()
    {
```

```
            return message;
        }
    }

}
```

4. CreditAgent.jws

```
package CreditAgent;

import weblogic.jws.control.JwsContext;

public class CreditAgent
{
    /** @common:context */
    JwsContext context;

    /**
     * @common:operation
     */
    public int CheckCredit(String cusomername, String accessname, String accesspwd)
    {
        if( accessname.equals("ups") && accesspwd.equals("ups") )
            return  5;
        else
            return -1;
    }
}
```

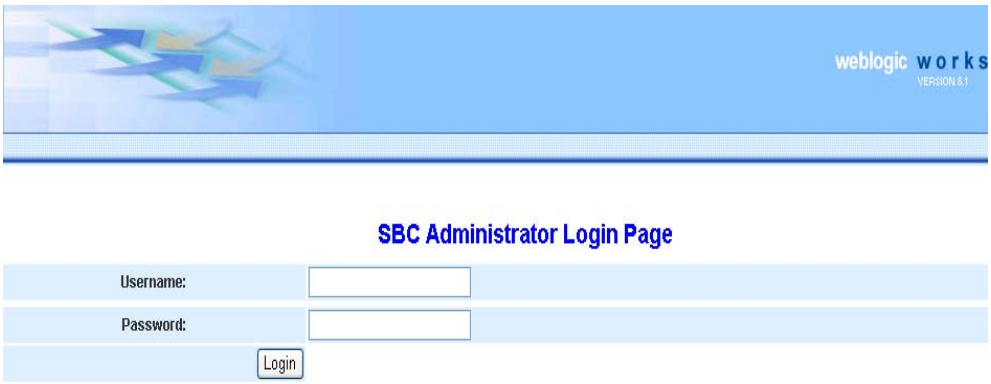
5. LoginResult.jsp



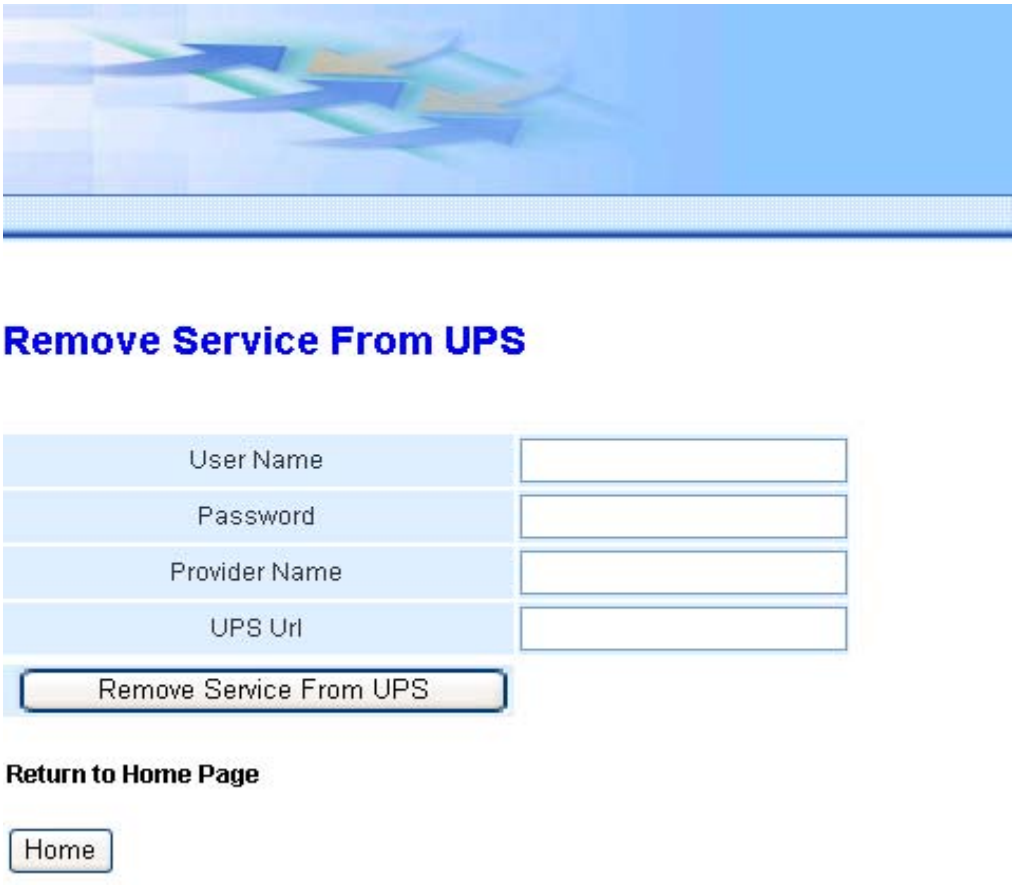
SBC Administrator Page

Actions With No Parameters	Input Forms For Actions With Parameter
Show All Orders	Add New Package
Show All Packages	Remove Service From UPS
	Logout

6. index.jsp



7. RemoveServiceFromUPS.jsp



8. ShowAllOrders.jsp

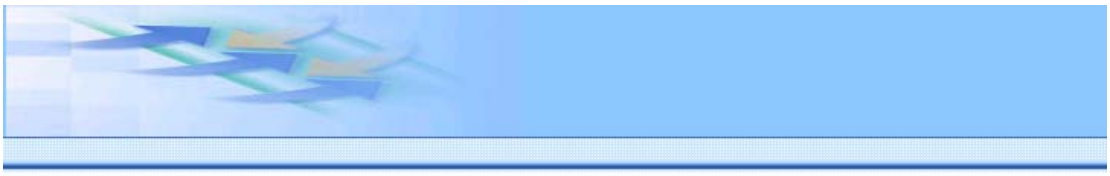
List of All Orders

Order ID	Order Status	Package Name	First Name	Last Name
1	active	Economic Phone	Smith	Jone
2	active	Middle Level	Joe	Ott
3	active	Economic Phone	Mark	Huang
4	active	Homeline	David	Mark
5	active	Deluxe Package	Will	Davis
6	active	Homeline Select	Woody	Woods
7	active	Homeline	Rogers	Chi
8	active	Economic Phone	Tom	Banks
9	active	Middle Level	David	Mark
10	active	Deluxe Package	Rogers	Chi
11	active	Homeline	Peter	smith

Return to Home Page

Home

9. ShowAllPackages.jsp



List of All Packages

Package ID	Package Name	Price	Remove
1	Economic Phone	19.99	Remove
2	Middle Level	29.99	Remove
3	Deluxe Package	39.99	Remove
4	Homeline Plus	39.99	Remove
5	Homeline Select	39.99	Remove
6	Homeline	9.99	Remove

Return to Home Page

Home

APPENDIX IV

SOURCE CODE OF CREATING DATABASE

The following material is the SQL source code to create database for the QOS registry, UPS portal and the two telecom service providers.

1. CreateTable_UPS
2. CreateTable_ Registry
3. CreateTable_ATT

1. CreateTable_UPS

```
DROP TABLE PROVIDER;
DROP TABLE CLIENT;
DROP TABLE ORDER;
DROP TABLE PACKAGE;
DROP TABLE FEATURE;

CREATE TABLE CLIENT
  (UID int identity(5000,1) primary key,
   EMAIL VARCHAR(30),
   PW VARCHAR(25),
   LNAME VARCHAR(25),
   FNAME VARCHAR(25),
   ADDRESS VARCHAR(50),
   CITY VARCHAR(20),
   STATE VARCHAR(20),
   ZIP VARCHAR(10),
   CREDIT_CARD_NUM VARCHAR(25)
);

insert into Client values('1@swt.edu','ups','Jone','Smith','street1','Austin','Tx','78778','1233445678');
insert into Client values('2@swt.edu','ups','Mark','David','street2','Houston','Tx','78777','3453467891');
insert into Client values('3@swt.edu','ups','Huang','Mark','street3','Dallas','Tx','78666','34512367871');
insert into Client values('4@swt.edu','ups','Smith','Peter','street4','Sanmacos','Tx','78999','3457446891');

create table Provider
(
ProviderId int identity primary key,
ProviderName varchar(40),
status varchar(20),
selfpw varchar(15),
address varchar(50),
email varchar(25),
url varchar(300),
comtype varchar(50),
accessproviderusername varchar(20),
accessproviderpw varchar(20)

);



CREATE TABLE ORDER(ORDERID int identity primary key,
   CLIENTID int,
   ACTIVE  VARCHAR(20),
   ORDERDATE date
   );


insert into Order values(5000,'active','1999-01-30');
insert into Order values(5004,'active','2000-03-31');
insert into Order values(5002,'active','2000-03-31');
insert into Order values(5008,'active','2000-03-31');
insert into Order values(5001,'pending','2000-03-31');
insert into Order values(5009,'pending','2000-03-31');
insert into Order values(5003,'cancelled','2000-03-31');


CREATE TABLE PACKAGE
  (
   PACKAGEID int identity PRIMARY KEY,
```

```
  PACKAGENAME VARCHAR(30),
  ORDERID int,
  PROVIDERNAME  VARCHAR(40),
  PRICE DOUBLE,
  STARTDATE date,
  ENDDATE date,
  STATUS varchar(25)
);


insert into package values('Economic Phone',1,'SBC',19.99,date'1999-01-30',date'2001-12-30','active');
insert into package values('Deluxe Pkg',1,'MCI',39.99,date'2000-03-31',date'2002-10-12','active');
insert into package values('Economic Phone',2,'SBC',19.99,date'1999-01-30',date'2001-12-30','active');
insert into package values('Most Basic',2,'Grande',9.99,date'2000-03-31',date'2002-10-12','active');
insert into package values('Deluxe Pkg',3,'MCI',39.99,date'2000-03-31',date'2002-10-12','active');
insert into package values('Super Deluxe',4,'ATT',39.99,date'2000-03-31',date'2002-10-12','active');
insert into package values('Most Basic',4,'Grande',9.99,date'2000-03-31',date'2002-10-12','active');
insert into package values('Economic Phone',5,'SBC',19.99,date'1999-01-30',date'2001-12-30','active');
insert into package values('Middle Level',5,'SBC',29.99,date'2000-03-31',date'2001-12-23','active');
insert into package values('Deluxe Pkg',5,'MCI',39.99,date'2000-03-31',date'2002-10-12','active');
insert into package values('Economic Phone',6,'SBC',19.99,date'1999-01-30',date'2001-12-30','active');
insert into package values('Deluxe Pkg',6,'MCI',39.99,date'2000-03-31',date'2002-10-12','active');
insert into package values('Economic Phone',7,'SBC',19.99,date'1999-01-30',date'2001-12-30','active');
insert into package values('Middle Level',7,'SBC',29.99,date'2000-03-31',date'2001-12-23','active');
insert into package values('Middle Level',8,'SBC',29.99,date'2000-03-31',date'2001-12-23','active');
insert into package values('Deluxe Pkg',8,'MCI',39.99,date'2000-03-31',date'2002-10-12','active');
insert into package values('Super Saving',8,'ATT',39.99,date'2000-03-31',date'2002-10-12','active');

CREATE TABLE FEATURE
 (PACKAGEID int,
  FEATURENAME VARCHAR(40),
  STATUS VARCHAR(20),
  PRICE double
  );

insert into Feature values(1,'touch tone','basic', 0.39);
insert into Feature values(1,'call waiting','basic', 1.99);
insert into Feature values(1,'caller ID','chosen', 3.99);
insert into Feature values(2,'caller ID','basic',3.99);
insert into Feature values(3,'touch tone','basic', 0.39);
insert into Feature values(3,'three way talking','chosen',2.99);
insert into Feature values(4,'touch tone','basic', 0.39);
insert into Feature values(4,'call waiting','basic', 1.99);
insert into Feature values(5,'touch tone','basic',0.29);
insert into Feature values(5,'call back','basic', 1.99);
insert into Feature values(5,'call wake up','chosen',1.99);
insert into Feature values(6,'emergency call','basic',0.99);
insert into Feature values(6,'call back','basic', 1.99);
insert into Feature values(6,'caller forward','chosen',1.59);

commit;



2 CreateTable_Registry

DROP TABLE REGISTRYPROVIDER;
DROP TABLE REGISTRYKEYS;

create table RegistryProvider
(
Providerid int identity primary key,
```

```
Providername varchar(40),
status varchar(20),
selfpw varchar(15),
address varchar(50),
email varchar(25),
url varchar(300),
servicetype varchar(50),
totalaccessed int,
price double,
avgexetime double,
aprap double,
transaction int,
timeout int,
compensaterate double,
penaltyrate double,
accessproviderusername varchar(20),
accessproviderpw varchar(20)
);

insert into RegistryProvider values('SBC','active','ups','2005 SBC Street', 'sbc@sbc.com',
'http://localhost:7001/BusinessProcessorA/BusinessProcessorWebService/BusinessProcessor.jws?WSDL'
,'telcom', 1, 25, 2, 3, 5, 60,  0.5,  0.5, 'ups', 'ups');

insert into RegistryProvider values('ATT','active','ups','2005 ATT
Street','att@att.com','http://localhost:7001/BusinessProcessorB/BusinessProcessorWebService/BusinessP
rocessor.jws?WSDL','telcom', 1,  40, 2, 3, 5, 200, 0.8, 0.1,'ups','ups');

insert into RegistryProvider values('Houston Credit','active','ups','203 Hillcroft
Street','houston@houston.com','http://localhost:7001/HoustonCreditAgent/CreditAgent/CreditAgent.jws?W
SDL','credit', 1,5, 3, 3, 5, 60, 0.5, 0.5,'ups','ups');

insert into RegistryProvider values('Austin Credit','active','ups','203 Congress
Road','austin@austin.com','http://localhost:7001/AustinCreditAgent/CreditAgent/CreditAgent.jws?WSDL','c
redit', 1, 15, 3, 3, 5, 200, 0.8, 0.1,'ups','ups');


create table Registrykeys
(
keyid int identity primary key,
password int,
providername varchar(20),
whenentered int
);

commit;



3 CreateTable_ATT

DROP TABLE ATT_CLIENT;
DROP TABLE ATT_ORDER;
DROP TABLE ATT_PACKAGE;
DROP TABLE ATT_FEATURE;
DROP TABLE ATT_PACKFEATURE;
DROP TABLE ATT_ORDPACKAGE;
DROP TABLE ATT_ORDPKGFEATURE;


CREATE TABLE ATT_CLIENT
(
  UID int identity(500,1) primary key,
```

```
  EMAIL VARCHAR(25),
  PW VARCHAR(25),
  LNAME VARCHAR(25),
  FNAME VARCHAR(25),
  ADDRESS VARCHAR(40),
  CITY VARCHAR(20),
  STATE VARCHAR(15),
  ZIP VARCHAR(10),
  CREDIT_CARD_NUM VARCHAR(20)
);

insert into ATT_Client values('1@swt.edu','ups','Jone','Smith','street1','Austin','Tx','78778','1233445678');

insert into ATT_Client values('2@swt.edu','ups','Mark','David','street2','Houston','Tx','78777','3453467891');

insert into ATT_Client values('3@swt.edu','ups','Huang','Mark','street3','Dallas','Tx','78666','34512367871');

insert into ATT_Client
values('4@swt.edu','ups','smith','Peter','street4','Sanmacos','Tx','78999','3457446891');

insert into ATT_Client values('5@swt.edu','ups','Ott','Joe','street5','Dallas','Tx','78666','3456732871');

insert into ATT_Client
values('6@swt.edu','ups','Zinger','Paul','street6','Sanmacos','Tx','78999','343476891');

insert into ATT_Client
values('7@swt.edu','ups','Davis','Will','street7','Sanmacos','Tx','78999','3482734691');

insert into ATT_Client values('8@swt.edu','ups','Woods','Woody','street8','Dallas','Tx','78666','98623946');

insert into ATT_Client
values('9@swt.edu','ups','Banks','Tom','street9','Sanmacos','Tx','78999','239847394');

insert into ATT_Client
values('10@swt.edu','ups','Chi','Rogers','street10','Sanmacos','Tx','78969','239847334');



CREATE TABLE ATT_ORDER
(
  ORDERID int identity primary key,
  CLIENTID int,
  ORD_STATUS  VARCHAR(20),
  ORDERDATE date,
  REFNO int
);

insert into ATT_Order values(500,'active','1999-01-30',1);
insert into ATT_Order values(504,'active','2000-03-31',2);
insert into ATT_Order values(502,'active','2000-03-31',3);
insert into ATT_Order values(501,'active','2000-03-31',4);
insert into ATT_Order values(506,'active','2000-03-31',5);
insert into ATT_Order values(507,'active','2000-03-31',6);
insert into ATT_Order values(509,'active','2000-03-31',7);
insert into ATT_Order values(508,'active','2000-03-31',8);
insert into ATT_Order values(501,'pending','2000-03-31',9);
insert into ATT_Order values(509,'pending','2000-03-31',10);
insert into ATT_Order values(503,'cancelled','2000-03-31',11);



CREATE TABLE ATT_ORDPACKAGE
```

```
(
   ORDERID int,
   PACKAGENAME VARCHAR(30),
   PKG_STATUS varchar(20)
);

insert into ATT_ORDPACKAGE values(1,'Economic Phone ATT','active');
insert into ATT_ORDPACKAGE values(2,'Middle Level ATT','active');
insert into ATT_ORDPACKAGE values(3,'Economic Phone ATT','active');
insert into ATT_ORDPACKAGE values(4,'Homeline ATT','active');
insert into ATT_ORDPACKAGE values(5,'Deluxe Package ATT','active');
insert into ATT_ORDPACKAGE values(6,'Homeline Select ATT','active');
insert into ATT_ORDPACKAGE values(7,'Homeline ATT','active');
insert into ATT_ORDPACKAGE values(8,'Economic Phone ATT','active');
insert into ATT_ORDPACKAGE values(9,'Middle Level ATT','active');
insert into ATT_ORDPACKAGE values(10,'Deluxe Package ATT','active');
insert into ATT_ORDPACKAGE values(11,'Homeline ATT','active');


CREATE TABLE ATT_PACKAGE
(
  PACKAGEID int identity PRIMARY KEY,
  PACKAGENAME VARCHAR(30),
  PRICE double,
  STATUS varchar(15)
);

insert into ATT_package values('Economic Phone ATT',19.99,'active');
insert into ATT_package values('Middle Level ATT',29.99,'active');
insert into ATT_package values('Deluxe Package ATT',39.99,'active');
insert into ATT_package values('Homeline Plus ATT',39.99,'active');
insert into ATT_package values('Homeline Select ATT',39.99,'active');
insert into ATT_package values('Homeline ATT',9.99,'active');


CREATE TABLE ATT_ORDPKGFEATURE
(
  ORDERID int,
  PACKAGENAME VARCHAR(30),
  FEATURENAME VARCHAR(40),
  FEA_STATUS VARCHAR(20)
 );

insert into ATT_ORDPKGFEATURE values(1,'Economic Phone ATT','caller ID','active');
insert into ATT_ORDPKGFEATURE values(2,'Middle Level ATT','three way calling','active');
insert into ATT_ORDPKGFEATURE values(3,'Homeline ATT','caller ID','active');
insert into ATT_ORDPKGFEATURE values(3,'Homeline ATT','three way calling','active');
insert into ATT_ORDPKGFEATURE values(4,'Economic Phone ATT','caller ID','active');
insert into ATT_ORDPKGFEATURE values(4,'Economic Phone ATT','call blocker','active');
insert into ATT_ORDPKGFEATURE values(4,'Economic Phone ATT','call return','active');
insert into ATT_ORDPKGFEATURE values(8,'Deluxe Package ATT','call blocker','active');
insert into ATT_ORDPKGFEATURE values(9,'Homeline Select ATT','call return','active');
insert into ATT_ORDPKGFEATURE values(10,'Homeline Plus ATT','caller ID','active');
insert into ATT_ORDPKGFEATURE values(10,'Homeline Plus ATT','three way calling','active');


CREATE TABLE ATT_FEATURE
(
   FEATURENAME VARCHAR(40),
   FEAPRICE double
);
```

```
insert into ATT_Feature values('speed dial',0.39);
insert into ATT_Feature values('call waiting',1.99);
insert into ATT_Feature values('caller ID',3.99);
insert into ATT_Feature values('three way calling',2.99);
insert into ATT_Feature values('emergency call',0.11);
insert into ATT_Feature values('call return',1.39);
insert into ATT_Feature values('call blocker',2.29);
insert into ATT_Feature values('call forwarding',1.99);
insert into ATT_Feature values('auto redial',0.99);



CREATE TABLE ATT_PACKFEATURE
(
    PACKAGEID int,
    FEATURENAME VARCHAR(40),
    FEA_OPTION boolean
  );

insert into ATT_PACKFEATURE values(1, 'speed dial', true);
insert into ATT_PACKFEATURE values(1, 'call waiting', true);
insert into ATT_PACKFEATURE values(1, 'caller ID', false);
insert into ATT_PACKFEATURE values(1, 'call blocker', false);
insert into ATT_PACKFEATURE values(1, 'call return', false);
insert into ATT_PACKFEATURE values(2, 'speed dial', true);
insert into ATT_PACKFEATURE values(2, 'three way calling', false);
insert into ATT_PACKFEATURE values(2, 'call blocker', false);
insert into ATT_PACKFEATURE values(2, 'call return', false);
insert into ATT_PACKFEATURE values(3, 'speed dial', true);
insert into ATT_PACKFEATURE values(3, 'three way calling', false);
insert into ATT_PACKFEATURE values(3, 'call blocker', false);
insert into ATT_PACKFEATURE values(3, 'call return', true);
insert into ATT_PACKFEATURE values(3, 'auto redial', true);
insert into ATT_PACKFEATURE values(3, 'three way calling', false);
insert into ATT_PACKFEATURE values(4, 'call blocker', true);
insert into ATT_PACKFEATURE values(4, 'call return', true);
insert into ATT_PACKFEATURE values(4, 'speed dial', true);
insert into ATT_PACKFEATURE values(4, 'three way calling', false);
insert into ATT_PACKFEATURE values(4, 'auto redial', true);
insert into ATT_PACKFEATURE values(4, 'caller ID', false);
insert into ATT_PACKFEATURE values(5, 'speed dial', true);
insert into ATT_PACKFEATURE values(5, 'three way calling', false);
insert into ATT_PACKFEATURE values(5, 'call return', false);
insert into ATT_PACKFEATURE values(5, 'caller ID', true);
insert into ATT_PACKFEATURE values(6, 'three way calling', false);
insert into ATT_PACKFEATURE values(6, 'auto redial', true);
insert into ATT_PACKFEATURE values(6, 'caller ID', false);
insert into ATT_PACKFEATURE values(6, 'speed dial', true);

commit;
```

BIBLIOGRAPHY

[1] Boualem Benatallha and Fabio Casati, editors.
Distributed and Parallel Database, Special issue on Web Sevices
Springer-Verlag, 2002

[2] Peter Farkas, Hassan Charaf.
Web Services Planning Concepts. *Journal of WSCG*, 11(1)
February, 2003

[3] Li-jie Jin, Vijay Machiraju, Akhi Sahai.
Analysis on Service Level Agreement of Web Services, Software Technology Laboratory, HP Laboratories Palo Alto, *HPL-2002-180*, June 21st, 2002

[4] E.Michael Maximilien, Munindar P. Singh.
Conceptual Model of Web Services Reputation
SIGMOD Record, October 2002

[5] E.Michael Maximilien, Munindar P. Singh.
Reputation and Endorsement for Web Services
ACM SIGecom Exchanges 3(1):24-31, 2002

[6] Daniel A. Menasce.
QoS Issues in Web Services
IEEE Internet Computing, 6(6), 2002

[7] J.O'Sullivan, D. Edmond, and A. ter Hofstede.
What's in a Service?
Distributed and Parallel Database, 12(2-3):117-133, September 2002

[8] M.P. Papazoglou and D. Georgakopoulos
Service-Oriented Computing.
Communications of the ACM, 46(10):25-65, 2003

[9] Shuping Ran.
A Model for Web Services Discovery With QoS
ACM SIGecom Exchanges 4(1):1-10, 2003

[10] Amit Sheth, Jorge Cardoso, John Miller and Krys Kochut.
QoS for Service-oriented Middleware.
Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI'02), Vol. 8, Orlando, Florida (July 2002) pp. 528-534.

[11] Aad van Moorsel.
Metrics for the Internet Age: Quality of Experience and Quality of Business. Technical Report HPL-2001-179, HP Labs, August 2001. Also published in 5th Performability Workshp, September 2001, Erlangen, Germany

[12] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng.
Quality Driven Web Services Composition. *In Proceedings of the 12th international conference on World Wide Web (WWW), Budapest, Hungary.* ACM Press, May 2003.

[13] D. Alur, J.Crupi, D. Malks.
Core J2EE Patterns.
Prentice Hall PTR, 2001

[14] Steve Graham, Simeon Simeonov, Toufic Boubez, Doug Davis, Clen Daniels
Building Web Services with Java
Sams Publishing, 2002

[15] Google Business
http://www.google.com/services/websearch.html

[16] UDDI. Universal Description Discovery and Integration, 2003
http://www.uddi.org/

[17] WSDL. Web Services Description Language, 2001
http://www.w3.org/TR/wsdl

[18] SOAP. Simple Object Access Protocol 2000
http://www.w3.org/TR/SOAP/

[19] QoS for Web Services
http://www.santra.com/knowledge/?id=qos

[20] Anbazhagan Mani, Arun Nagarajan
Understanding Quality of Service for Web Service,
http://www-106.ibm.com/developerworks/library/ws-quality.html

[21] Web Service Transaction
http://www-106.ibm.com/developerworks/webservices/library/ws-transpec/

[22] Web Service Flow Language
http://www-106.ibm.com/developerworks/webservices/library/ws-wsfl1/

[23] BPEL In Action Using the Google Search Web Service
http://www.collaxa.com/devpack.samples.GoogleSearch.hmtl

[24] Definition of QoS
http://www.ewh.ieee.org/r2/baltimore/Chapter/Comm/dlt_talk/sld012.htm

[25] Masksim A. Aleksandrov, Vladislav S. Voinov
Designing and Implementing QoS
Management of the Web

[26] Jorge Carodos, John Miller, Amit Sheth and Jonathan Arnold
Modeling Quality of Services for Workflows and Web Service Process
Technical Report 02-002 V2, LSDIS Lab

[27] Anbazhagan Mani, Arun Nagarajan
Understanding Quality of Service for Web Service
http://www-106.ibm.com/developerworks/library/ws-quality.html

VITA

Yutu Liu was born in Yiyang, Hunan, China on October 1, 1965. After completing his work at Yiyang NO.1 High School, Yiyang, Hunan, China in 1984, he entered Central South University in Changsha, Hunan, China. In January 2001, he entered the Graduate School of Southwest Texas State University, San Marcos, Texas.

Permanent Address: Baoji Radio Plant
Baoji, Shaanxi, China

This thesis was typed by Yutu Liu.