Design and Implementation of Java Platform for Web Service Based Application Development

GRADUATE PROJECT

Submitted to the Faculty of
the Department of Computing and Mathematical Sciences
Texas A&M University- Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Srinivas Jaini
Fall 2006

Committee Members

Dr. Dulal C. Kar
Committee Chairperson

Dr. Long-zhuang Li
Committee Member

Dr. David Thomas
Committee Member
ABSTRACT

Web services technology is an effective technology that provides an environment for building loosely coupled, decentralized applications where diverse systems can collaborate in a platform-independent and language-agnostic way. Enterprise data has evolved into various forms including databases, Enterprise Resource Planning (ERP), flat files and many others. Data is one of the important factors to help make business decisions and analysis effectively. Disparate data and geographically distributed data pose a major challenge in presenting data at one central place to enable business intelligence in current business environments. A platform named Java Web Services and JavaScript (JWSJS) is developed using Java language in this project to enhance the existing Web services technology and use JavaScript to provide rich user interfaces like real-time data. The JWSJS platform developed in this project consists of reusable templates that can be customized to develop custom applications. To demonstrate the implementation, templates of the JWSJS platform are used to build three applications that connect distributed, disparate systems and provide rich user interface. Among the custom applications developed, a Web application that provides data in real-time is the significant one utilizing all of the JWSJS platform’s features. This Web application presents data in real-time without unnecessary resources and complexity. Over all, the JWSJS platform developed in this project is used to develop other applications that are a need of modern day business and scientific computing. This code will be contributed to open source community.
# TABLE OF CONTENTS

ABSTRACT ........................................................................................................................................... i

TABLE OF CONTENTS ......................................................................................................................... ii

LIST OF FIGURES ............................................................................................................................... iv

1. BACKGROUND AND RATIONALE .............................................................................................. 1
   1.1 THE PROBLEM ......................................................................................................................... 1
   1.2 WHAT IS A WEB SERVICE? ....................................................................................................... 2
   1.3 WHY USE WEB SERVICES? ......................................................................................................... 2
   1.4 SYSTEM ARCHITECTURE .......................................................................................................... 4
   1.5 THE PROBLEM OF REAL-TIME DATA ..................................................................................... 6
   1.6 ENTERPRISE APPLICATIONS – INDUSTRY STANDARDS ......................................................... 6
   1.7 THE NEED FOR AN ARCHITECTURE ....................................................................................... 7

2. JAVA WEB SERVICES AND JAVASCRIPT .................................................................................... 9
   2.1 J2EE WEB SERVICE IMPLEMENTATION ............................................................................... 9
   2.2 WEB SERVICE ARCHITECTURE ............................................................................................ 10
   2.3 REAL-TIME DATA .................................................................................................................... 14
   2.4 JAVA WEB SERVICES AND JAVASCRIPT PLATFORM .......................................................... 16
   2.5 DEVELOPING APPLICATIONS USING THE JWSJS PLATFORM .............................................. 17
   2.6 WEB APPLICATION IMPLEMENTATION .................................................................................. 19
   2.7 WEB SERVICE CONSUMER ..................................................................................................... 21

3. SYSTEM DESIGN ............................................................................................................................ 23
   3.1 DEVELOPING WEB SERVICE TEMPLATES ............................................................................. 26
   3.2 DEVELOPING WEB APPLICATION TEMPLATES ..................................................................... 26
   3.3 DEVELOPING THE PLATFORM .................................................................................................. 27
   3.4 JWSJS PLATFORM AND REUSABILITY ................................................................................... 28
   3.5 THE ENHANCED PLATFORM - PROVIDING DATA IN REAL-TIME ........................................... 30
   3.6 DEVELOPING APPLICATIONS USING THE JWSJS PLATFORM ............................................. 31
      3.6.1 Zip code look up Web application .................................................................................... 32
### 3.6.2 Stocks quote Web application

3.6.3 Real-time Stocks Web application

### 4. TESTING AND EVALUATION

4.1 UNIT TESTING

4.1.1 Web service client

4.1.2 Ajax client user interface testing

4.2 USABILITY TESTING

4.3 INTEGRATION TESTING

4.4 THE PLATFORM TESTING

### 5. FUTURE WORK

### 6. CONCLUSION

BIBLIOGRAPHY AND REFERENCES

APPENDIX A – Data Dictionary

APPENDIX B – Page Flow Controller for Client Web Application

APPENDIX C – Welcome Screen JSP

APPENDIX D – Initial Prototype of Response Screen

APPENDIX E – RealTime Stocks Page Flow Controller

APPENDIX F – stockquote.js - Java Script

APPENDIX G – Template Web Service Control

APPENDIX H – Template PageFlow Controller

APPENDIX I – Template Request Page

APPENDIX J – Template Results Page

APPENDIX K – Template JavaScript

APPENDIX L – Zip code lookup JavaScript

APPENDIX M – RealTime Stocks Web Service Control
LIST OF FIGURES

Figure 1.1  System Architecture Diagram.........................................................5
Figure 2.1  Package Structure of Web Service....................................................11
Figure 2.2  Architecture of a Typical Web Service Operation..............................13
Figure 2.3  Comparison of Classic and Ajax Web Application Model......................15
Figure 2.4  Overview diagram............................................................................18
Figure 2.5  Stocks Portal Home Page.................................................................19
Figure 2.6  Stocks Portal Form Page.................................................................20
Figure 2.7  Stocks Portal Results Page..............................................................20
Figure 2.8  Web service consumer request page................................................21
Figure 2.9  Web service consumer result page...................................................22
Figure 3.1  Web Services Network Diagram.......................................................23
Figure 3.2  Development and process flow chart...............................................25
Figure 3.3  Page Flow Controller for Client Web Application...............................27
Figure 3.4  Re-usable PageFlow Controller.......................................................28
Figure 3.5  Component diagram of the JWSJS platform.......................................29
Figure 3.6  stockquote.js JavaScript...................................................................30
Figure 3.7  zip code look up JavaScript function...............................................32
Figure 3.8  Real time stocks Web service control...............................................34
Figure 4.1  Web service client unit test results...................................................35
Figure 4.2  SOAP request XML.......................................................................36
Figure 4.3  SOAP response XML......................................................................37
Figure 4.4  Form Page (Improved)....................................................................39
1. BACKGROUND AND RATIONALE

In this chapter the complexity of integrating disparate data and the need for real-time data are discussed. A brief introduction of Web service technology, its advantages in comparison with other existing technologies is provided along with rationale behind the need for a new architecture.

1.1 The Problem

In the modern business, scientific and computing world, data has a specific significance for various reasons. Data can be used for empirical analysis, history, pattern design, business intelligence and scientific research. Over the course of time data have been produced in various shapes and sizes posing a challenge to its usefulness. Some forms of data are database, spread-sheets, and files. There is a need to integrate data and provide transparency into these data systems. This need has become a very important aspect in the world of computing over the last decade. Various application middle-ware vendors are developing new solutions to provide real-time interfaces for disparate and distributed systems to provide data in one central location for users. Using the platform developed in this project, data can be made available to users in a single place called portal instead of searching for data at various locations. The platform developed uses Web services technology in combination with Asynchronous JavaScript And eXtensible Markup Language (Ajax) to provide real-time data to users. The platform developed in this project is used to develop applications that provide real-time data in scenarios like stock tickers. This prototype will provide a foundation for various other use cases where data needs to be presented in real-time to a user who uses Web browser.
1.2 What is a Web Service?

Web Service Architecture Group defines a Web service as a software system identified by a URI (Uniform Resource Identifier), whose public interfaces and bindings are defined and described using XML (Extensible Mark-up Language) [W3C 2005]. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by internet protocols.

A Web service is programmable application logic accessible using standard Internet protocols. Web services combine the best aspects of component-based development and the Web. Like components, Web services represent black-box functionality that can be reused without worrying about how the service is implemented. A Web service can also be defined as a software component that provides a consistent, transparent API for invocation of services by using message-oriented transport mechanisms, which can be dynamically located and bound, and by utilizing XML for data representation and transformation.

1.3 Why use Web services?

Web services are one-way, asynchronous messages mapped onto executable software programs. Web services define a data format independent of programming language, operating system, network transport, and data storage mechanism; therefore data has to be mapped into and out of the independent format. Data typing and structure abstracted from underlying implementations of services [Newcomer 2002].
Web services are not as much like traditional distributed computing technologies such as CORBA or DCOM, as they are like Web servers, HTML, and HTTP, on which they are based.

Web services are often compared to remote procedure call invocations or software components. Web services are more appropriately compared to enterprise application integration adapters. Web services define a canonical message format, as EAI (Enterprise Application Integration) software systems, such as IBM MQSeries, TIBCO, do and define the way in which the message is directed to a service interface through which the data is mapped or transformed onto an underlying application. In other words, the intelligence for understanding how to map a message into a software program is not contained within the interface itself, as it is in CORBA, J2EE (Java 2 Enterprise Edition), and DCOM, all of which are based on Remote Procedure Call (RPC) concepts, which tightly couple the service name to the program being invoked. Rather, that intelligence is contained within the XML processor, which consumes the message and follows associated instructions on how to parse the message and map the data into whatever program implements the Web service.

In addition, Web services do not require or assume the existence of the same software system on both ends of a communication path. EAI adapters similarly accept a canonical message format and map the information in the message to an enterprise resource planning (ERP) or other type of enterprise application. Web services are defined at a similar level of abstraction, which allows the same message type to be mapped to multiple applications, including, but certainly not limited to, RPC-based components.
Unlike RPC-oriented middleware, such as CORBA and DCOM, Web services use unidirectional, asynchronous messaging, which is mapped to a message queuing system, such as MQSeries or Java Message Service (JMS), than to CORBA or DCOM. Web services support a request/ response paradigm which is typical of synchronous, RPC-style communications through emulation; that is, the XML processor. The Web services emulation of an RPC is easily mapped to traditional RPC-based systems such as CORBA, EJB, and DCOM, although qualities of service (e.g., security, transactions, and exception handling), are unique when compared to traditional technologies.

The interactions with Web services are accomplished through programs and databases to which the Web services are mapped. The user experience is likely to be very different from a typical browser-based experience.

Web services by themselves are not executable but instead have to be mapped to a program, an object, a middleware system, or a database management system [Newcomer 2002].

Taking into consideration all the above mentioned qualities, Web services technology is used to develop this project.

1.4 System Architecture

Web services technology is used to build an application that operates with a system that provides a Web service provider and offers its services. It is a challenge to find a Web service that is publicly available which provides data in real time.

In Figure 1.1 three systems are interacting with each other synchronously by transmitting data over the Internet. The following six steps elucidate the flow in Figure 1.1:
1. Browser requests information by passing a HTTP request

2. Application server processes the request, generates a Simple Object Access Protocol (SOAP) request and sends it to Web service provider via internet.

3. Web service provider receives the SOAP request.

4. Web service provider processes the request and sends SOAP response back to the sender via internet.

5. The application server receives the SOAP response.

Figure 1.1 System Architecture Diagram
6. The application server processes the SOAP response and sends HTTP response back to the browser.

The browser represents an end user using internet browser, such as Internet Explorer or Netscape Navigator, to request and get response from a Web site. The application server receives user requests, processes requests by sending an SOAP request to Web service producer and re-transmitting the processed SOAP response from the Web service provider back to the user in the form of a HTTP response via the Internet.

1.5 The Problem of Real-time Data

Providing real-time data is an important factor in the modern business, computing and scientific world. Web services provide inter-operability and platform independence. In combination with these features Web services provide, data in real-time would be a great success, and also more of a need, in the world of distributed and diverse applications. A Web service provider that provides real-time data is chosen to develop a Web service client that requests data from the provider without human invocation to refresh Web page on the client side. As this project is developed, several software vendors are in the process of developing tools that can be used not only to connect distributed, heterogenous platform systems but also to provide Rich Internet Application (RIA).

1.6 Enterprise Applications – Industry Standards

J2EE, .Net besides C++, Perl, CGI and Python have been some of the major platforms recently used to build enterprise applications. Java 2 Platform, Enterprise Edition (J2EE) defines the standard for developing component-based multi-tier enterprise applications. J2EE simplifies building enterprise applications that are portable, scalable,
and that integrate easily with legacy applications and data. Hence Java is chosen to develop an enhanced platform in this project.

1.7 The Need for an Architecture

The generalized term "Web services" did not, at the time of its inception, describe a coherent or necessarily consistent set of technologies, architectures, or even visions. The community of Web services evangelists, architects, developers, and vendors represented a merging of at least three major sources of inspiration, with various ideas taken from other sources as well. Several streams of thought and practice were converged to produce an amalgam of what became known as "Web services" [WAWG 2006]. This community includes:

- "Distributed Objects" or "Application Integration" -- exchange of programming objects or invocation of software functions over a network.
- The World Wide Web itself - accessing human readable documents and posting requests for information, products, or services via the HTTP protocol.

The Java platform, J2EE, is used to combine the power of JavaScript technology with Web services technology to develop the Java Web Services and JavaScript (JWSJS) platform. Integration of these two technologies is carefully done to properly implement Object Oriented Programming (OOP) concepts and principles. Re-usability, cohesion and code-to-interface are some of the important principles being considered in developing this platform.

In this project the JWSJS platform is conceived, designed and developed. Custom applications are developed using the JWSJS platform to demonstrate the implementation.
Web services architecture and operations are explained in chapter 2. The conception, design and implementation plan of the JWSJS platform are introduced in chapter 2. Chapter 3 describes the development of the JWSJS platform and also describes the various custom applications that are developed using the platform. In chapter 4, the testing methodology and results of testing are listed. Chapter 5 and 6 discuss future work and conclusion respectively.
2. JAVA WEB SERVICES AND JAVASCRIPT

In this chapter Web service implementation is described in detail. This chapter encompasses introduction of the JWSJS platform, its design and use. Three major components of the project development are:

- A platform using Java to build Web applications that connect with distributed and heterogenic applications that provide Web service interface.
- Enhance the platform with ability to provide real-time data using JavaScript technology. This platform with enhancements is named Java Web Services and JavaScript (JWSJS).
- Develop several implementations, including Web applications that provide data in real-time, using the JWSJS platform.

J2EE has been an industry standard to build enterprise applications. This project uses Java platform, J2EE, to develop the JWSJS platform as well as all the implementations like Web services and Web applications. These applications are deployed on BEA WebLogic application server.

2.1 J2EE Web Service Implementation

J2EE is a platform for building and using Web services [Sun 2005]. It incorporates Web services standards such as those in the Web Services Interoperability (WS-I) Basic Profile. This means that Web services in a J2EE-compliant environment can interoperate with Web services in non-J2EE environments such as .Net. Web service features in J2EE 1.4 address both the server and client sides of Web services [Sommers
2003]. The features extend J2EE to allow existing server-side enterprise Java components to become Web services and specify how a J2EE client container can invoke Web services.

The technologies for both aims have existed for a while, and the new J2EE specifications rely on those existing APIs for Web services support. The new specifications add to the existing technologies a set of interoperability requirements, and a programming and deployment model for Web service integration. There are two specifications that explicitly outline those added features: Java Specification Request 151, the umbrella Java Specification Request (JSR) for J2EE 1.4, and JSR 109, Web Services for J2EE.

2.2 Web Service Architecture

A Web service is composed of one or more operations, whereby each operation may be implemented using different backend components [Mountjoy 2004]. A separate set of message handlers for each operation can be created. For example, a Web service operation can be implemented by a single method of a standard Java object, or by a combination of SOAP message handlers and a remote method of a stateless session Enterprise Java Beans (EJB). Because Web services extend the capabilities of Web applications, WebLogic requires that web-services.xml, a deployment descriptor be defined that captures the vital information describing a Web service. This XML-formatted descriptor file is located under the /WEB-INF folder of a Web application and includes the following information on each Web service:

- It specifies the backend components used to implement the various operations of the Web service.
• It defines any SOAP handler chains that intercept incoming and outgoing SOAP messages.

• It specifies the XML Schema definitions for any custom data types that are used as parameters and return values for Web service operations. It also provides the XML-to-Java data type mappings that specify the serialization class and Java classes for the custom data types.

• It declares the actual operations supported by the Web service and associates each operation with a backend component and/or SOAP handler chain.

The `web-services.xml` descriptor file is crucial to properly configuring Web services. WebLogic uses `web-services.xml` descriptor file to automatically generate WSDL document for the deployed Web service. Figure 2.1 illustrates the package structure of a typical Web service.

![Figure 2.1: Package Structure of a Web Service](image-url)
Web services are packaged into standard J2EE enterprise applications EAR, acronym for Enterprise Archives. The EAR file includes the Web application (WAR, acronym for Web Archive) that contains the `web-services.xml` descriptor file and any Java classes that implement the Web services, the message handlers and support classes for handling custom data types. It also packages any EJB JAR files for any stateless session EJBs or JMS (Java Message Services) consumers and producers that implement the Web services operations. WebLogic provides several Ant, acronym for Another Neat Tool, tasks for assembling the various components of the Web service into an EAR file. Ant (Another Neat Tool) is a build utility tool that is used to build deployable archive files from code base.

Because WebLogic’s Web services are packaged within enterprise applications, they can integrate with the rest of the J2EE framework. Web services can automatically benefit from WebLogic support for various J2EE features: access to JDBC connection pools and JTA (Java Transaction API, where API stands for Application Programmable Interface) transactions, the business objects within the enterprise application, and simple and unified security model. Figure 2.2 illustrates the architecture of a typical Web service operation.

The following eight steps explain the operational flow from Figure 2.2. When a client invokes an operation exposed by a Web service, the following actions occur:

1. Translate the JAX-RPC (Java API for XML, Remote Procedure Call) call to a SOAP request.
2. Read and parse the resulting SOAP message.
3. SOAP message goes through sequence of message handlers if the operation is associated with message handlers.

4. WebLogic deserializes the request XML using appropriate classes.

5. Web service invokes back end component.

6. WebLogic converts returned Java values to SOAP formatted XML.

7. If an operation is associated with message handlers, the message will go through message handlers in reverse order.

8. WebLogic returns the SOAP message to the client.

The client-side libraries translate the JAX-RPC call to a SOAP request, which is then sent to server. The server (Web service producer host) inspects the URI (Uniform Resource Identifier) of the incoming request to determine which Web services are to be invoked. The Web service also needs to parse the SOAP message to determine which operation needs to be invoked. If the operation is associated with a chain of message handlers, the SOAP message must go through the sequence of message handlers. Each
message handler may potentially alter the SOAP message or may even abort the operation. Using the appropriate deserializer classes, WebLogic builds a Java representation of the inbound parameters from their XML representation. This deserialization logic relies both on WebLogic’s support for handling built-in data types, and on deserialization classes that handle any custom data types used. The Web service then invokes the backend component associated with the operation parsing with Java parameters. WebLogic then converts the return values from Java to XML using the appropriate serializer classes and creates a SOAP message response for the client. If the operation is associated with a chain of message handlers, the SOAP response must again pass through a chain of SOAP message handlers, but this time in reverse order. Once again, each message handler may alter the SOAP response or abort the operation. WebLogic finally sends the SOAP message back to client. At the client’s end the client-side stubs intercept the returned SOAP message and extract the return value(s) before passing them on to the client.

2.3 Real-time Data

In most cases, not only is data needed but also it is needed in real-time to be useful. Wikipedia defines real-time as: An operation within a larger dynamic system is called a real-time operation if the combined reaction- and operation-time of a task is shorter than the maximum delay that is allowed, in view of circumstances outside the operation [Wikipedia 2006].

A change in database data, most often, requires a Web browser to be refreshed to get the new data. This process involves in network, database, and application resources besides a manual task of refreshing the Web page. If data changes rapidly, a Web user
might have to refresh the browser several times and all this in turn consumes a lot of resources as described above. Addressing a solution to this issue is a critical phase of the project.

Ajax is a technology that can be used to solve this problem of real-time data. Asynchronous JavaScript and XML, or its acronym Ajax, is a Web development technique for creating interactive Web applications. Figure 2.3 shows difference between a general browser client and a browser client with Ajax technology. Ajax enhances a browser to behave like a rich browser with automatic data loading without user refresh. The JavaScript on browser uses the Ajax engine to accomplish this task and thus there is no need to download or install new libraries on the client browser [Garrett 2006]. This ability of Ajax is used to customize and build features in the JWSJS platform.

![Figure 2.3 Comparison of Classic and Ajax Web Application Model](image)

Figure 2.3 Comparison of Classic and Ajax Web Application Model
Finding a real-time data Web service provider is a challenging job since most of them are not available in public. Some possible sources of real-time Web services that could be used for this project are:

- MapPoint Web Service [MapPoint 2006]
- Yahoo Traffic Web Service [YahooTraffic 2006]
- StrikeIron Real-time Stocks Quote [StrikeIron 2006]

### 2.4 Java Web Services and JavaScript Platform

Leveraging Web service API developed by Web services community, a platform is developed in this project called Java Web Services and JavaScript (JWSJS). This platform is developed on Java platform utilizing the following technologies:

- Apache Beehive’s PageFlow framework
- Apache Beehive’s Web service control framework
- JavaScript (Ajax)

This platform provides reusable and customizable code to create Web service implementations with rich user interfaces. JWSJS is enhanced using Asynchronous JavaScript and XML, with acronym Ajax, a technology that can be used to handle elements on a Web page. To demonstrate, the capability of this enhanced platform, an application is developed that provides real-time data via a Web page to users. A rich Web user interface using JWSJS platform that has the following characteristics is developed:

- No manual intervention required- browser clients do not have to click the refresh button in the browser to get data in real-time.
• Does not need expensive resources consumption tasks- a typical browser refresh uses application, network and database resources each time it is refreshed.

Using JWSJS technology, with the embedded JavaScript technology, only the data that is changed is being transmitted between browser and the Web application server. This platform is used to develop several customized applications in this project.

2.5 Developing applications using the JWSJS platform

Figure 2.4 is an overview diagram that shows how the JWSJS platform and its various components are used to develop custom applications. The JWSJS platform contains template files that can be broadly classified into three components:

• JavaScript templates

• Web application templates

• Web service control templates

These template files are used to develop various customized Web applications that interact with remote Web service providers. These custom Web applications can use some or all of the features provided by the JWSJS platform.
Web applications can be developed using JWSJS platform to provide the data processed by Web service consumer to users. The end user interacts with the Web application via a browser. This Web application has a form that processes user requests, submits data to Web service client which interacts with Web service provider. The results are then displayed on the user’s browser.
2.6 Web Application Implementation

A stocks Web application is developed using JWSJS platform on WebLogic Workshop IDE. The Web application consists of Java Server Pages that provide user interface. All the Web application files are developed using the template files of the JWSJS platform.

Figure 2.5 is the portal’s home page. The home page lists all the available services that are provided by this application. Figure 2.6 is the form page where a user inputs the information requested and submits the data to get results. Figure 2.7 shows the results page as seen by the user.
Figure 2.6 Stocks Portal Form Page

Figure 2.7 Stocks Portal Results Page
2.7 Web Service Consumer

Web service consumer is the component of this system that is built using the JWSJS platform, just like the Web application component. The Web service consumer uses the standards of Web services. It communicates synchronously with StrikeIron Web service provider [StrikeIron 2006] to request data using SOAP and XML.

Figure 2.9 is a snapshot of request page of Web service consumer. The page’s right column has several blocks each of which is a Web service operation. This can be clarified by checking the operations mentioned in WSDL. The first operation takes exchange, symbol and bid-to-ask parameters. The button in the block is for form submission. On button click, the request, SOAP formatted XML, is sent to Web service producer and the results are displayed back in the browser.

Figure 2.8 Web service consumer request page
Figure 2.9 shows the result received for the above request. Figure 2.9 shows both request and response XML. The values can be seen in bold in the response. It should be noted that this Web service consumer is the key bridge that submits and gets data. Web application provides a medium to display this data. But with the implementation of Ajax using the JWSJS platform, the Web application is transformed into what is popularly known as Rich Internet Application.
3. SYSTEM DESIGN

This chapter encompasses detailed description of development lifecycle of the JWSJS platform and complete details of development and testing of custom applications built using the JWSJS platform.

In the first phase of this project, Web services technology is implemented and used to build a prototype as shown in Figure 3.1.

![Web Service Network Diagram](image)

Figure 3.1 Web Service Network Diagram.

The application that hosts Web service provides interface via the Internet. This interface is used as mode of communication to request all available information from the Web service. The information is passed over the Internet and hence is the best to connect geographically distributed, disparate systems.
The two major components developed in this project are:

- Web service consumer
- Web application

A Java Web application is designed and developed to act as a Web service client, in the sense that a Web service producer will be requested for data. Several Web service providers are available publicly and data can be requested via the Internet. WSDL (Web Service Description Language) provides public contract of any given Web service provider. A public contract is a description of what requests will be processed by a Web service and how the responses will be formatted and delivered.

The code base created in the development process of this prototype is then used to build the JWSJS platform. Figure 3.2 shows high level development and process flow diagram. As can be seen from Figure 3.2, this project is executed in several steps starting with development of a prototype as a proof of concept. The development of prototype was implemented in several iterations and along the process bugs and issues were fixed. On successful completion of prototype, a thorough implementation of a Web service consumer and Web application is executed, described in more detail in sections 3.1 and 3.2. The development of this code is executed in several iterations fixing issues and bugs along the way. On successful completion of Web service client implementation, the code is re-factored and generalized to develop the JWSJS platform. This platform is made up of several re-usable template files that can be used to build various applications. Using the JWSJS platform a Web application is developed and tested.

The JWSJS platform is then enhanced with JavaScript technology called Ajax. The enhanced JWSJS platform provides features to develop rich and real-time user
interfaces. A real-time stocks portal is then developed using all the features of the enhanced the JWSJS platform.

Figure 3.2 shows a process named ‘Future work’, which shows that this platform development will be an on-going effort and more features will be added over time that would enable development of more sophisticated applications.

![Development and process flow chart](image-url)

Figure 3.2 Development and process flow chart.
3.1 Developing Web Service Templates

Web Services are typically comprised of a Web service producer, Web service consumer and WSDL (a public contract to specify services provided by the Web service).

Using the StrikeIron WSDL, listed in Appendix M, a Web service control is generated. This Web service control is used to create a Web service control for the JWSJS platform. Appendix G lists the Web service control template that is used to build the JWSJS platform.

3.2 Developing Web Application Templates

After careful consideration of possible frameworks including Apache Struts and Beehive Java Page Flows for Web application development, this project is developed on Page Flow framework, a part of Apache Beehive, an open-source project [Jaini 2005]. Page Flows provide rich features like modularity, rich JSP tags, session data management, and ease of use.

The client Web application is a Page Flow. Page Flows are MVC (Model View Controller) frameworks that have a controller file which acts as director to change views based on requests and business logic (model) responses. Figure 3.3 is template prototype of page flow controller of a Web application developed using Java Page Flows. Page Flow controller is a java class that contains member variables and action methods. Action methods contain the logic that determines which view is to be displayed depending on
public class Controller extends PageFlowController {
    private static class bookObject {
        private String bookTitle;
        private String bookAuthor;
        private String bookPrice;
        private boolean bookAvailability;
    };
    public void Action() {
        //logic
    }
    //other methods like Action() method
}

Figure 3.3 Page Flow controller for client Web application

The listing in Figure 3.3 shows a controller Java class with ‘private’ (sometimes ‘public’) variables and public action methods which perform the logic and change the view based the logic and specification. Appendix B lists the complete Page Flow controller developed for the Web application. Appendix H lists the page flow controller template, developed using the page flow controller listed in Appendix B. The page flow template is used in building the JWSJS platform.

3.3 Developing the Platform

The code developed in section 3.1 and 3.2 is generalized and wrapped up into a code base, a platform called JWSJS, which is used to develop other applications with minimal effort. Files of the code base are listed in Appendix G (Web service control), Appendix H (Page Flow controller), Appendix I (request JSP), Appendix J (result JSP) and Appendix K (JavaScript) provide complete listings of template files developed in
JWSJS platform. The development of the JWSJS platform conforms to Object Oriented Programming principles like inheritance, polymorphism and reusability. Certain parameters like ‘endUrl’, ‘MAX_SECONDS’, and ‘noOfSeconds’ are configurable to change the behavior and configuration of the application. This implies that this code can be used to develop applications that do updates at a different interval or connect to a different Web service.

```java
public class CustomizedPageFlowController extends PageFlowController {
    /**
     * This
     * @common:control
     */
    private RealTimeStockQuotesTestControl myControl;
}
```

Figure 3.4 Re-usable PageFlow Controller

Figure 3.4 shows a section of customized PageFlow controller developed using the JWSJS platform. The control declared in the PageFlow determines the Web service that it connects to. So the PageFlow controller file listed in Appendix E can be used to build new Web applications with minimal changes.

3.4 JWSJS platform and reusability

The JWSJS platform is used to develop real-time Web applications that connect distributed systems. Appendices G, H, I, J and K list the template files that provide base platform used to develop various applications. The JWSJS platform developed in this
project is supported by BEA WebLogic Platform 8.1. Figure 3.5 is a pictorial representation of the JWSJS platform along with external components this system interacts with.

![Component diagram of the platform.](image)

Figure 3.5 Component diagram of the platform.

The following three applications implementations are developed in this project using the JWSJS code base.

- CDYNE real-time stock quotes implementation. [Cdyne 2006]
- WebserviceX.net Zip code implementation. [WebserviceX 2006]
- StrikeIron stocks quote real-time implementation. [StrikeIron 2006]

Similar applications, that need to inter-operate with distributed and heterogenic platforms, can be developed using the JWSJS platform. Depending on the scenario real-time feature can either be used or not used. Thus this platform is not only a reusable code base but also an effective tool to deliver similar applications with less development effort.
3.5 The Enhanced Platform - providing data in real-time

Ajax technology is customized and adapted to provide real-time data, from a distributed, disparate data source, in a browser. Ajax uses XmlHttpRequest object to interact with the HTTP (Hyper Text Transfer Protocol) request and html page

```javascript
function stockquote_update() {
    //window.alert("Starting...");
    if (secs==0)
    {
        // Specify web service provider
        var endurl = //example
            escape("http://ws.cdyne.com/delayedstockquote/delayedstockquote.asmx/GetQuote");
        // Endpoint of the Proxy servlet that will make the call on our behalf
        var localurl = "/Portal/ProxyServlet";
        // Input parameters: for the ZIP Code
        var symbol = document.getElementById("stockquote_symbol").value;
        // Use the XmlHttpRequest object, to make HTTP request
        var url = localurl + "?url=" + endurl + 
            "&StockSymbol=" + symbol + "&LicenseKey=0"; // The server-side script
        iteration = iteration +1;
        self.status = "updating...";
        // URL connection - XMLHttpObject.
        stockquote_http.open("GET", url, true);
        // Set a callback handler to a local JavaScript method
        stockquote_http.onreadystatechange = stockquote_http.onreadystatechange =
            stockquote_handleHttpResponse;
        // Make the call. You can replace the null value with XML request data if you are doing
        //a SOAP-style call instead of using HTTP request parameters.
        stockquote_http.send(null);
        secs = MAX_SECONDS;
        //Iteration logic with a loop
    }
    else
    {  //Decrement 'secs' variagle and show status }
}
```

Figure 3.6 stockquote.js JavaScript.
Figure 3.6 is pseudo code of how a real-time data is polled at a given interval that can be set to any natural number. In this particular instance the interval is set to 10 seconds. So data is refreshed on the page every 10 seconds without any user intervention.

The JavaScript shown in Figure 3.6 uses XMLHttpRequest object to control elements of HTML displayed on the page and hence enables a granular control of different elements of page without refreshing the whole page. Appendix F has the complete implementation of this JavaScript.

With minimal changes like modifying variables ‘endurl’, ‘url’, this JavaScript can be used to change the Web service it connects to. The time interval at which updates are made can be changed by modifying variables like ‘MAX_SECONDS’, ‘noOfTimes’.

3.6 Developing applications using the JWSJS platform

The JWSJS platform can be used by any developer who has proficiency in Java, Web services and JavaScript to develop custom applications. The intent of this project is to open-source the JWSJS platform, which means that developers interested in enhancing this platform can register and add new feature in accordance with open-source policies. Open source describes practices in production and development that promote access to the end product's source materials—typically, their source code. Some consider it as a philosophy, and others consider it as a pragmatic methodology. Before open source became widely adopted, developers and producers used a variety of phrases to describe the concept; the term open source gained popularity with the rise of the Internet and its enabling of diverse production models, communication paths, and interactive communities. Subsequently, open source software became the most prominent face of
open source practices [Wikipedia 2006]. Contributing this platform to open-source is future work which is discussed in chapter five.

Three applications are developed using the JWSJS platform to demonstrate the variety of applications that can be developed using this platform. As shown in Figure 3.6, the platform template files are used to develop customized the following applications.

3.6.1 Zip code look up Web application

This application developed using the JWSJS platform, requests data from a publicly available Web service provider. The zip code Web service provider [WebserviceX 2006] operates with the standard SOAP and XML protocol. The Web application developed using JavaScript feature of the platform to request data from the Web service provider.

```javascript
function zipcode_updateDirect() {
    var url = "http://www.webservicex.net/uszip.asmx/GetInfoByZIP?USZip="; // The server-side script
    var zipValue = document.getElementById("zipcode_USZip").value;

    // Open a url connection using the XMLHttpRequest. The third parameter specifies that the
    // call should be made asynchronously. Set this to false to make this call synchronous.
    zipcode_http.open("GET", url + escape(zipValue), false);

    // Set a callback handler to a local Javascript method
    zipcode_http.onreadystatechange = zipcode_http.onreadystatechange;

    // Make the call. You can replace the null value with XML request data if you are doing
    // a SOAP-style call instead of using HTTP request parameters.
    zipcode_http.send(null);
}
```

Figure 3.7 zip code look up JavaScript function.
Figure 3.7 lists the main function in JavaScript that makes Web service request and processes the results. Full Listing is provided in Appendix K. The JavaScript generated using the template template.js, provided in Appendix L, makes Web service request and processes the response and update the Web page with the result. Note that only those elements of the Web page that correspond to the Web service response are updated. In this Web application the page gets updated with values that correspond to city and state when a zip code is entered.

3.6.2 Stocks quote Web application

Stocks quote Web application requests data from a Web service provided by Cdyne Corporation [Cdyne 2006]. This Web service is available publicly and conforms to Web service standards. Stocks quote Web application is developed using template Web service control, listed in Appendix G. This Web application takes user input, which is a stock symbol and returns the data provided by the Web service.

3.6.3 Real-time Stocks Web application

StrikeIron provides a Web service that provides data in real-time [StrikeIron 2006]. This Web service is available publicly but the trial license is only valid for 1000 requests. This Web application is developed using the JWSJS platform. Templates listed in Appendices G, H, I, J and K are used to develop this Web application. This Web application uses templates of the JWSJS platform to utilize the Web service and real-time capabilities of the platform. Appendix E provides listing of the Web application’s PageFlow named StocksPageFlowController.jpf. This PageFlow lists the logic and declaration of Web service control, listed in Appendix M. Figure 3.8 lists a section of the Web service control.
Figure 3.8 Real time stocks Web service control.

Figure 3.8 lists a class SIRealTimeQuote whose objects store the state of result fields that are displayed to the user. So, when a query is made by a user, a SOAP request is generated and sent to the Web service provider. The result is mapped onto SIRealTimeQuote object which is displayed to the user. The real-time feature is accomplished using template.js listed in Appendix K. The JavaScript polls the Web service every n seconds, where n is configured by default to 5 but can be configured to any natural number, and updates the Web page with new values. Note that this does not use all the expensive resources involved in a Web page refresh. Also note that the Web page is updated without any human interaction with the Web page.
4. TESTING AND EVALUATION

This chapter describes and lists the testing methodologies, including their results, that are used to test the JWSJS platform and the applications developed using it. The components of all the custom applications including Web application, JavaScript scripts and Web service control are tested and results are provided.

4.1 Unit Testing

Unit testing is performed on all components of the applications developed using the JWSJS platform. Unit tests are done on Web service client and rich Ajax client user interface testing.

4.1.1 Web service client

Web service client is tested by passing different parameters to verify the results. The following are the results recorded during these tests

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th>Obtained results (yes/no)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOOG/goog</td>
<td>Yes</td>
</tr>
<tr>
<td>YHOO/yhoo</td>
<td>Yes</td>
</tr>
<tr>
<td>MSFT/msft</td>
<td>Yes</td>
</tr>
<tr>
<td>VZB/vzb</td>
<td>No</td>
</tr>
</tbody>
</table>

Figure 4.1 Web service client unit test results.

SOAP XML request, for example when ‘MSFT’ is requested, is shown in Figure 4.2. It consists of SOAP envelope that wraps XML elements with values that are required...
by the Web service producer to process the request. In this case, UserID, Password, Exchange, Symbol, IncludeBidAsk are the values being passed in the request.

Submitted at Sunday, September 3, 2006 7:26:32 PM PDT

```xml
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <ns1:LicenseInfo xmlns:ns1="http://ws.strikeiron.com">
      <RegisteredUser>
        <UserID>srinivasjaini@yahoo.com</UserID>
        <Password>password</Password>
      </RegisteredUser>
    </ns1:LicenseInfo>
  </soapenv:Header>
  <SOAP-ENV:Body>
    <ns:GetRealQuote xmlns:ns="http://www.strikeiron.com">
      <ns:Exchange>INET</ns:Exchange>
      <ns:Symbol>msft</ns:Symbol>
      <ns:IncludeBidAsk>true</ns:IncludeBidAsk>
    </ns:GetRealQuote>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 4.2 SOAP Request XML

Figure 4.2 is a SOAP request sent to the Web service provider. Data transmission is done over the Internet and the mode used in this operation is synchronous, which means that the client submits the request and waits until it gets the response.

Submitted at Sunday, September 3, 2006 7:26:32 PM PDT

```xml
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Header>
    <ResponseInfo xmlns="http://www.strikeiron.com">
```

36
Figure 4.3 SOAP response XML

Figure 4.3 shows the response sent by Web service provider for the request SOAP XML shown in Figure 4.2. Figure 4.3 shows the XML response with the details of the
quote requested, including values for ‘Name’, ‘Rank’, ‘ChangeFromPrevious’, ‘Ask’, ‘Bid’ and other values.

### 4.1.2 Ajax client user interface testing

This test is performed to verify if data on the page is updated at various intervals specified. Data is toggled from normal to bold on every update to make page updates visually obvious and clear to a user. Tests are performed by setting update interval to 10, 5, 2 and 1 seconds. Tests showed results as expected i.e., when interval was set to 10 seconds, the values on page polled for real-time data every 10 seconds.

### 4.2 Usability Testing

Web user interface is an integral part of the system built in this project. Web components are tested for human usability using the following factors:

- **Time on Task:** How long does it take to complete basis tasks? (For example, how long does it take to submit request and get response?)
- **Accuracy:** How many mistakes did people make?
- **Recall:** How much does a person remember or recollect after periods of non-use?
- **Emotional Response:** How stressful did a user feel about the user interface and tool?

<table>
<thead>
<tr>
<th>User</th>
<th>Time on Task (seconds)</th>
<th>Accuracy</th>
<th>Recall (percentage)</th>
<th>Emotional Response (excellent, good, bad)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Praveena</td>
<td>20</td>
<td>0</td>
<td>100</td>
<td>Excellent</td>
</tr>
<tr>
<td>Bill</td>
<td>50</td>
<td>0</td>
<td>100</td>
<td>Excellent</td>
</tr>
<tr>
<td>Jeff</td>
<td>30</td>
<td>0</td>
<td>100</td>
<td>Excellent</td>
</tr>
</tbody>
</table>
Each user is tested and their responses are captured along with the above statistics. The user input is then iteratively used to enhance the user interface to provide better human usability. User interface pages are enhanced from Figures 2.6 and 2.7 to 4.3 and 4.4 respectively.

Figure 4.4 Form Page (improved)

Figure 4.5: Results Page (improved)
Figure 4.4 is improvised form of Figure 2.6. Additional text and input examples are provided for better user experience. Figure 2.7 is improvised to Figure 4.5 by listing the fields that are relevant for the users and in a convenient order and also providing more clarity on the link that takes user back to home page.

4.3 Integration Testing

Integration is performed at application layer. The components, Web application and Web service consumer are integrated to consummate the system. The system is integrated and testing is done from the user interface using the unit tests mentioned in section 4.1.1. The integrated application showed results as expected. In the next level, integration is done with Ajax engine to provide data in real-time without manual intervention. The expectation that once the user enters data the results page shows up with data in real time is met, which means that the results page is updated whenever the data in the Web service provider changes.

4.4 The Platform Testing

The platform developed in this project is thoroughly tested by developing several implementations of the code for scenarios listed below.

- CDYNE real-time stock quotes implementation. [Cdyne 2006]
- WebserviceX.net Zip code implementation. [WebserviceX 2006]
- StrikelIron stocks quote real-time implementation. [StrikelIron 2006]

Each of these implementations is tested using the above testing methods including usability testing, unit testing, and integration testing. Usability testing is iteratively used to enhance Web usability and provide better user experience. Unit tests are done separately on each component of the above applications and bugs are fixed. The final
integration testing is done using black-box testing methodology on each of the above applications.
5. FUTURE WORK

The JWSJS platform can be used to develop advanced features and solutions to enhance user experience like Google Maps [Google Maps 2006] and even more. Future work could involve development of, but not limited to, more features in the JWSJS platform like communicating with multiple Web service providers. Development of an API that will provide a standard interface for developers to leverage features available in the JWSJS platform is one of the potential future works. Planning and implementation methodologies on how to maintain and enhance the platform as an open source project need to be done.

The open source model can allow for the concurrent use of different agendas and approaches in production, in contrast with more centralized models of development such as those typically used in commercial software companies. "Open source" as applied to culture defines a culture in which fixations are made generally available [Wikipedia 2006]. Registered participants can contribute to maintain and enhance the JWSJS platform.
6. CONCLUSION

This project is developed using Web services technology based on enterprise standards and using technologies like J2EE, WebLogic, XML, and SOAP. This project solves the problem of connecting distributed systems together using a platform independent technology. More importantly this project solves the issue of providing real-time data among distributed connected systems. This project is executed in three phases:

- Developed the JWSJS platform that combines Web services and JavaScript (Ajax) technologies.
- Developed various implementations of Web service consumer including features like real-time data availability. Three different implementations are developed using the platform developed in this project.

In totality, this project builds major components using Web services technology and demonstrates inter-operability as well as real-time data availability, as an experimental combination of two modern technologies. The JWSJS platform developed as part of this project will be provided to open source community and the pursuit to enhance this platform will be in progress. So any developer who wants to contribute to this on-going effort can download the code available and build applications besides adding new features to the platform itself. Developers who wish to contribute to the platform need to be proficient and experienced in Java, Web services and JavaScript.
BIBLIOGRAPHY AND REFERENCES


APPENDIX A – Data Dictionary

Ajax: Asynchronous JavaScript and XML, or its acronym Ajax, is a Web development technique for creating interactive web applications.

API: Application Programming Interface. An API is a set of definitions of the ways one piece of computer software communicates with another. It is a method of achieving abstraction, usually between lower-level and higher-level software.

EJB: Enterprise JavaBeans (EJB) technology is the server-side component architecture for Java Platform, Enterprise Edition (Java EE). EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology.

J2EE: Java 2 Enterprise Edition is a programming platform — part of the Java platform — for developing and running distributed multi-tier architecture applications, based largely on modular components running on an application server. The Java EE platform is defined by a specification. Java EE is also considered informally to be a language or standard because providers must agree to certain conformance requirements in order to declare their products as Java EE compliant; albeit with no ISO or ECMA standard.

JAR: Java Archive is a ZIP file used to distribute a set of Java classes. It is used to store compiled Java classes and associated metadata that can constitute
a program. OpenDocument files are also Java archives which store XML files and other objects.

Java: Object Oriented Programming language developed initially by James Gosling and colleagues at Sun Microsystems.

JAX-RPC: Java API for XML based Remote Procedure Call, developed by Sun Microsystems defines the client API needed for invoking a Web service. WebLogic supplies a client JAR that includes an implementation of the JAX-RPC 1.0 specification, thereby allowing Java clients to access both WebLogic and other non-WebLogic Web services. The JAX-RPC standard is central to understanding WebLogic’s Web services framework.

JSP: Java Server Pages, known to some as the Java Scripting Preprocessor, is a Java technology that allows developers to dynamically generate HTML, XML or some other type of Web page. The technology allows Java code and certain pre-defined actions to be embedded into static content.

JSR: Java Specification Request, used by Java Community Process to take into consideration the standards and needs of industry and develop new versions of Java technologies.

MVC: Model View Controller framework/design pattern to develop Web applications.

Page Flows: Model 2 framework, an extension of Struts framework to build MVC (Model View Controller) design pattern.
SOAP: Simple Object Access Protocol. A standard used to envelope XML messages that are transmitted across internet. SOAP is a lightweight, XML-based communications protocol for exchanging messages.

Struts: Model 2 framework built on MVC (Model View Controller) design pattern

TIBCO: A middle-ware vendor with Enterprise Application Integration tools.

XML: Extensible Markup Language. A standard to exchange data

UDDI: Universal Description, Discovery and Integration, specification defines a standard way to describe Web services, publish Web services over a registry and discover other registered Web services. WebLogic supports the UDDI 2.0 standard for publishing Web services over a registry and for inquiring about other registered Web services.

W3C: World Wide Web Consortium is an international consortium where member organizations, a full-time staff, and the public, work together to develop standards for the World Wide Web. W3C's mission is: "To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web". W3C also engages in education and outreach, develops software, and serves as an open forum for discussion about the Web. The Consortium is headed by Tim Berners-Lee, the original creator of the World Wide Web and primary author of the URL (Uniform Resource Locator), HTTP (HyperText Transfer Protocol) and HTML (HyperText Markup Language) specifications, the principal technologies that form the basis of the Web.
WSDL: The Web Services Description Language (WSDL) is the XML application that is used to describe a Web service. A WSDL document describes the various operations exposed by a Web service, their input and output parameters, and how to access the Web service. WebLogic supports WSDL 1.1-compliant descriptions for Web services.

WebLogic: BEA WebLogic is a J2EE application server and also an HTTP Web server by BEA Systems of San Jose, California, for Unix, Linux, Microsoft Windows, and other platforms. WebLogic supports Oracle, DB2, Microsoft SQL Server, and other JDBC-compliant databases. WebLogic Server supports WS-Security.
import com.bea.wlw.netui.pageflow.FormData;
import com.bea.wlw.netui.pageflow.PageFlowController;
import com.bea.wlw.netui.pageflow.Forward;
import java.util.ArrayList;

/**
 * This is the default controller for a blank Web application.
 *
 * @jpf:controller
 * @jpf:view-properties view-properties::
 * <view-properties>
 * <pageflow-object id="pageflow:/Controller.jpf"/>
 * <pageflow-object id="page:error.jsp">
 *   <property name="x" value="260"/>
 *   <property name="y" value="60"/>
 * </pageflow-object>
 * </view-properties>
 * <pageflow-object id="formbean:Controller.RequestWebSvcForm"/>
 * <pageflow-object id="action:requestWebSvc.do#Controller.RequestWebSvcForm">
 *   <property value="200" name="x"/>
 *   <property value="200" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="formbeanprop:Controller.RequestWebSvcForm#authorName#java.lang.String"/>
 * <pageflow-object id="formbeanprop:Controller.RequestWebSvcForm#publisher#java.lang.String"/>
 * <pageflow-object id="formbeanprop:Controller.RequestWebSvcForm#edition#java.lang.String"/>
 * <pageflow-object id="page:requestPage.jsp">
 *   <property value="60" name="x"/>
 *   <property value="200" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="page:responsePage.jsp">
 *   <property value="360" name="x"/>
 *   <property value="200" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="action-call:@page:requestPage.jsp@@action:requestWebSvc.do#Controller.RequestWebSvcForm@">
 *   <property value="96,130,130,164" name="elbowsX"/>
 *   <property value="192,192,192,192" name="elbowsY"/>
 */
private static class bookObject {
    private String bookTitle;
    private String bookAuthor;
    private String bookPrice;
    private boolean bookAvailability;
}

public ArrayList booksList;

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

/**
 * @jpf:action
* @jpf:forward name="success"
  path="responsePage.jsp"
*/

protected Forward requestWebSvc(RequestWebSvcForm form)
{
    return new Forward("success");
}

/**
 * FormData get and set methods may be overwritten by the Form Bean editor.
 */
public static class RequestWebSvcForm extends FormData
{
    private String yearPublished;
    private String edition;
    private String publisher;
    private String authorName;

    public void setAuthorName(String authorName)
    {
        this.authorName = authorName;
    }

    public String getAuthorName()
    {
        return this.authorName;
    }

    public void setPublisher(String publisher)
    {
        this.publisher = publisher;
    }

    public String getPublisher()
    {
        return this.publisher;
    }

    public void setEdition(String edition)
    {
        this.edition = edition;
    }

    public String getEdition()
    {
        return this.edition;
    }
}
public void setYearPublished(String yearPublished)
{
    this.yearPublished = yearPublished;
}

public String getYearPublished()
{
    return this.yearPublished;
}
}
APPENDIX C – Welcome Screen JSP

```jsp
<%@ page language="java" contentType="text/html;charset=UTF-8" %>
<%@ taglib uri="netui-tags-databinding.tld" prefix="netui-data" %>
<%@ taglib uri="netui-tags-html.tld" prefix="netui" %>
<%@ taglib uri="netui-tags-template.tld" prefix="netui-template" %>

<netui:html>
  <p> <b>Search Books</b> </p>
  <netui:form action="requestWebSvc">
    <table>
      <tr valign="top">
        <td>AuthorName:</td>
        <td>
          <netui:textBox dataSource="{actionForm.authorName}"/>
        </td>
      </tr>
      <tr valign="top">
        <td>Edition:</td>
        <td>
          <netui:textBox dataSource="{actionForm.edition}"/>
        </td>
      </tr>
      <tr valign="top">
        <td>Publisher:</td>
        <td>
          <netui:textBox dataSource="{actionForm.publisher}"/>
        </td>
      </tr>
      <tr valign="top">
        <td>YearPublished:</td>
        <td>
          <netui:textBox dataSource="{actionForm.yearPublished}"/>
        </td>
      </tr>
    </table>
    <br/>
    <netui:button value="Submit" type="submit"/>
  </netui:form>
</netui:html>
```
APPENDIX D – Initial Prototype of Response Screen

<%@ page language="java" contentType="text/html;charset=UTF-8" %>
<%@ taglib uri="netui-tags-databinding.tld" prefix="netui-data" %>
<%@ taglib uri="netui-tags-html.tld" prefix="netui" %>
<%@ taglib uri="netui-tags-template.tld" prefix="netui-template" %>
<netui:html>

<p>
Search Results
</p>
<table>
Books matching criteria
<tr>
<td> Book Title </td>
<td> Author </td>
<td> Price </td>
<td> Availability </td>
</tr>
</table>
</netui:html>

APPENDIX E – RealTime Stocks Page Flow Controller
package portlets.stocks;
import com.bea.wlw.netui.pageflow.FormData;
import com.bea.wlw.netui.pageflow.Forward;
import com.bea.wlw.netui.pageflow.PageFlowController;

import controls.RealTimeStockQuotesTestControl;

/**
 * PageFlow class generated from control
 * RealTimeStockQuotesTestControl
 * @jpf:controller
 * @jpf:view-properties view-properties::
 * <!-- This data is auto-generated. Hand-editing this
 * section is not recommended. -->
 * <view-properties>
 * <pageflow-object
 * id="pageflow:/portlets/stocks/StocksController.jsp"
 * />
 * <pageflow-object id="action:begin.do">
 * <property value="240" name="x"/>
 * <property value="100" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="action:GetRealQuote.do#portlets.stocks.StocksController.GetRealQuoteForm">
 * <property value="720" name="x"/>
 * <property value="300" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="action:GetRealQuoteLink.do">
 * <property value="400" name="x"/>
 * <property value="300" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="action-call:@page:index.jsp@#@action:GetRealQuoteLink.do@">
 * <property value="East_1" name="fromPort"/>
 * <property value="West_1" name="toPort"/>
 * <property value="276,320,320,364" name="elbowsX"/>
 * <property value="292,292,292,292" name="elbowsY"/>
 * </pageflow-object>
 * <pageflow-object id="page:index.jsp">
 * <property value="240" name="x"/>
 * <property value="300" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="action-call:@page:Results.jsp@#@action:begin.do@">
 * <property value="684,480,480,276" name="elbowsX"/>
public class StocksController extends PageFlowController {

    /**
     * This is the control used to generate this pageflow
     * @common:control
     */
    private RealTimeStockQuotesTestControl myControl;

    // Uncomment this declaration to access Global.app.
    //
    //    protected global.Global globalApp;
    //
    // For an example of page flow exception handling see
    // the example "catch" and "exception-handler"
protected Forward begin()
{
    return new Forward("success");
}

public Forward GetRealQuote(GetRealQuoteForm aForm)
throws Exception
{
    controls.RealTimeStockQuotesTestControl.SIRealTimeStockQuotesTestControl.SIRealTimeStockQuotesTestControl.SIRealTim
    eQuote var =
    myControl.GetRealQuote(aForm._Exchange,
    aForm._Symbol, aForm._IncludeBidAsk);
    Forward f = new Forward("success");
    f.addPageInput("result", var);
    return f;
}

public Forward GetRealQuoteLink()
throws Exception
{
    return new Forward("success");
}

/* @jpf:exception-handler
*/
/* @jpf:exception-handler
*/

/**
 * 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
 * :GetRealQuote
 * @jpf:action
 * @jpf:catch method="exceptionHandler"
 * type="Exception"
 * @jpf:forward name="success" path="Results.jsp"
 */
public Forward GetRealQuote(GetRealQuoteForm aForm)
throws Exception
{
    controls.RealTimeStockQuotesTestControl.SIRealTimeStockQuotesTestControl.SIRealTimeStockQuotesTestControl.SIRealTim
    eQuote var =
    myControl.GetRealQuote(aForm._Exchange,
    aForm._Symbol, aForm._IncludeBidAsk);
    Forward f = new Forward("success");
    f.addPageInput("result", var);
    return f;
}
protected Forward exceptionHandler(Exception ex,String
    actionName,String message,FormData form)
{
    String displayMessage = "An exception occurred in
    the action " + actionName;
    getRequest().setAttribute("errorMessage",
        displayMessage);
    return new Forward("errorPage");
}

/**
* FormData class GetRealQuoteForm
* FormData get and set methods may be overwritten by
  
  the Form Bean editor.
* /
public static class GetRealQuoteForm extends FormData
{
  private java.lang.String _Exchange;
  private java.lang.String _Symbol;
  private boolean _IncludeBidAsk;

  public void set_Exchange(java.lang.String Exchange)
  {
      this._Exchange = Exchange;
  }

  public java.lang.String get_Exchange()
  {
      return _Exchange;
  }

  public void set_Symbol(java.lang.String Symbol)
  {
      this._Symbol = Symbol;
  }

  public java.lang.String get_Symbol()
  {
      return _Symbol;
  }

  public void set_IncludeBidAsk(boolean
      IncludeBidAsk)
  {
{  
    this._IncludeBidAsk = IncludeBidAsk;
}

public boolean get_IncludeBidAsk() {
{  
    return _IncludeBidAsk;
}
}
// This is the request object used to run an AJAX style query
var stockquote_http = new XMLHttpRequest();

var stockquote_symbol;
var stockquote_name;
var stockquote_last;
var stockquote_change;
var isEven = true;

var secs = 0;
var MAX_SECONDS = 2;
var timerID = null;
var timerRunning = false;
var delay = 1000;
var numberOfTimes = 10;
var iteration = 0;

// This is a callback method used to handle returned data from a query.
// If you had multiple AJAX queries on this page you would make multiple
// handlers and name them uniquely.
function stockquote_handleHttpResponse()
{
    if (stockquote_http.readyState == 4) {
        if(stockquote_http.status == 200) {
            var name;
            var last;
            var change;
            var doc = Sarissa.getDomDocument();

            if(isMSIE) {
                doc.async = false;
            }

            doc = (new DOMParser()).parseFromString(stockquote_http.responseText, "text/xml");

            // Check for parse errors
if(doc.parseError != 0) alert("Not well formed or other parser error!");

// Testing
try {

document.getElementById('stockquote_status').innerHTML = stockquote_http.statusText;

if(isMSIE) {
    // I have no idea why this is necessary for this particular XML result set,
    // but it is. If you can come up with a cross-browser way to make this one work,
    // please send me an email at jmargagl@bea.com!

    var nodes = doc.documentElement.childNodes;
    for(var i = 0; i < nodes.length; i++)
    {
        if(nodes[i].baseName == 'CompanyName') name = nodes[i].text;
        if(nodes[i].baseName == 'LastTradeAmount') last = nodes[i].text;
        if(nodes[i].baseName == 'StockChange') change = nodes[i].text;
    }
}
else {
    name = doc.getElementsByTagName('CompanyName')[0].firstChild.data;
    last = doc.getElementsByTagName('LastTradeAmount')[0].firstChild.data;
    change = doc.getElementsByTagName('StockChange')[0].firstChild.data;
}
} catch(e) {
    window.alert("Error parsing XML: " + e.name + " -- " + e.message);
}

try {
    // Set the form fields with the new data
    if(isEven) {  

document.getElementById('stockquote_name').innerHTML = "<b>"+name+"</b>";

document.getElementById('stockquote_last').innerHTML = "<b>"+last+"</b>";

document.getElementById('stockquote_change').innerHTML = "<b>"+change+"</b>";
    isEven = false;

}
else {

document.getElementById('stockquote_name').innerHTML = name;

document.getElementById('stockquote_last').innerHTML = last;

document.getElementById('stockquote_change').innerHTML = change;
    isEven = true;

}
}
catch(e) {
    window.alert("Error updating document: " + e.name + " -- " + e.message);
}
}
else {

document.getElementById('stockquote_status').innerHTML = "Loading...";
}
}
else {

document.getElementById('stockquote_status').innerHTML = "Loading...";
}
}

function stockquote_showResults() {
    window.alert(stockquote_http.responseText);
}
// This is the method that is called when the Proxy button is clicked.
// This function calls the web service through a local proxy so that
// Mozilla's security mechanisms are not invoked. As far as the browser is
// concerned, all of the information is coming from one server.
function stockquote_update() {
    //window.alert("Starting...");

    if (secs==0)
    {
        // This is the endpoint for the ZIP Code web service
        var endurl = escape("http://ws.cdyne.com/delayedstockquote/delayedstockquote.asmx/GetQuote");

        // This is the endpoint of the Proxy servlet that will make the call on our behalf
        var localurl = "/Portal/ProxyServlet";

        // This is the input parameter for the ZIP Code
        var symbol = document.getElementById("stockquote_symbol").value;

        // This is the actual URL that will be called using the XmlHttpRequest object
        var url = localurl + "?url=" + endurl + "&StockSymbol=" + symbol + ";LicenseKey=0"; // The server-side script
        iteration = iteration +1;
        self.status = "updating...";
        // Open a url connection using the XMLHttpRequest.
        The third parameter specifies that the // call should be made asynchronously. Set this to false to make this call synchronous.
        stockquote_http.open("GET", url, true);

        // Set a callback handler to a local JavaScript method
        stockquote_http.onreadystatechange = stockquote_handleHttpResponse;
    }
}
// Make the call. You can replace the null value with XML request data if you are doing // a SOAP-style call instead of using HTTP request parameters.
stockquote_http.send(null);
secs = MAX_SECONDS;
if (iteration < numberOfTimes)
{
    self.setTimeout("stockquote_update()", delay);
}
else
{
    // alert("Time out. No updates anymore.");
    self.status = "Updates done."
    iteration = 0;
}
else
{
    secs = secs - 1;
    self.status = "Page will be updated in "+secs+ " seconds.";
    timerRunning = true;
    timerID = self.setTimeout("stockquote_update()", delay);
}
package controls.Templates;

/**
 * @jc:location http-url="TemplateWebSvc.jws" jms-url="TemplateWebSvc.jws"
 * @jc:wsdl file="#WebSvcWsdl"
 */
public interface TemplateWebSvcControl extends com.bea.control.ControlExtension,
        com.bea.control.ServiceControl
{
    public static class BeanA
        implements java.io.Serializable
    {
        public java.lang.String fieldA;
        public java.lang.String fieldB;
        public double fieldC;
    }

    public static class BeanB
        implements java.io.Serializable
    {
        public java.lang.String fieldA;
    }

    public BeanA RequestOperationA (java.lang.String reqA,
            java.lang.String reqB);

    public BeanB RequestOperationB (java.lang.String reqA);

    static final long serialVersionUID = 1L;
}

/** @common:define name="TemplateWebSvcWsdl" value::
 * copy past WSDL below
 <?xml version="1.0" encoding="utf-8"?>
 <definitions>
  **Web service operations would be listed here.
 </definitions>
 */
package portlets.TemplatePageFlowController;
import com.bea.wlw.netui.pageflow.Forward;
import com.bea.wlw.netui.pageflow.PageFlowController;

/**
 * @jpf:controller
 * @jpf:view-properties view-properties:
 * <!-- This data is auto-generated. Hand-editing this section is not recommended. -->
 * <view-properties>
 * <pageflow-object id="pageflow:/portlets/TemplatePageFlowController/TemplatePageFlowControllerController.jpf"/>
 * <pageflow-object id="action:begin.do">
 *   <property value="80" name="x"/>
 *   <property value="100" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="page:TemplateIndex.jsp">
 *   <property value="240" name="x"/>
 *   <property value="100" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="forward:path#success#TemplateIndex.jsp#@action:begin.do@">
 *   <property value="116,160,160,204" name="elbowsX"/>
 *   <property value="92,92,92,92" name="elbowsY"/>
 *   <property value="East_1" name="fromPort"/>
 *   <property value="West_1" name="toPort"/>
 *   <property value="success" name="label"/>
 * </pageflow-object>
 * <pageflow-object id="action:goToFormPage.do">
 *   <property value="460" name="x"/>
 *   <property value="100" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="page:TemplateFormPage.jsp">
 *   <property value="460" name="x"/>
 *   <property value="260" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="forward:path#success#TemplateFormPage.jsp#@action:goToFormPage.do@">
 *   <property value="460,460,460,460" name="elbowsX"/>
 *   <property value="144,180,180,216" name="elbowsY"/>
 *   <property value="South_1" name="fromPort"/>
 * </pageflow-object>
public class TemplatePageFlowControllerController extends PageFlowController {

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

}
// For an example of page flow exception handling see
// the example "catch" and "exception-handler"
// annotations in {project}/WEB-INF/src/global/Global.app

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

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

/**
 * @jpf:action
 * @jpf:forward name="success"
 * path="TemplateResultsPage.jsp"
 */
protected Forward requestWebSvcForResults()
{
    return new Forward("success");
}
APPENDIX I – Template Request Page

<netui:form action="requestWebSvcForResults">
  <table class="tablebody">
    <tr class="tablebody">
      <td>Field A:</td>
      <td>
        <netui:checkBox dataSource="{actionForm._fieldA}"/>
      </td>
    </tr>
    <tr class="tablebody">
      <td>Field B:</td>
      <td>
        <netui:textBox dataSource="{actionForm._fieldB}"/>
      </td>
    </tr>
    <tr class="tablebody">
      <td>Field C:</td>
      <td>
        <netui:textBox dataSource="{actionForm._fieldC}"/>
      </td>
    </tr>
  </table>
  <br/>
  <netui:button value="Submit" type="submit"/>
</netui:form>
### APPENDIX J – Template Results Page

```html
<netui-data:declarePageInput name="result"
    type="ObjectTypeReturned"/>

<div>
    <netui:anchor action="begin">Go to home page</netui:anchor>
</div>
<br>
<br>
<table border="1">
<tr>
    <td> Field A: </td>
    <td><netui:label
        value="{pageInput.result.fieldA}"/></td>
</tr>
<tr>
    <td> Field B: </td>
    <td><netui:label
        value="{pageInput.result.fieldB}"/></td>
</tr>
</table>
<br>
<br>
<div>
    <netui:anchor action="begin">Go to home page</netui:anchor>
</div>
<br>
<br>
<input value="Go to Web Services provider"
    type="button"
    onclick="window.open('http://www.xyz.com','mywindow')">
```
```
APPENDIX K– Template JavaScript

// This is the XMLHttpRequest request object used to run
AJAX style queries
var template_http = new XMLHttpRequest();
var variable1, variable2;

// template_handleHttpResponse() is a callback method used
to handle data returned from a query.
// For multiple queries multiple call back handlers need to
be used.
function template_handleHttpResponse() {
    if (template_http.readyState == 4) {
        if(template_http.status == 200) {
            var doc = (new
                DOMParser()).parseFromString(template_http.respon
                seText, "text/xml");

            document.getElementById('template_status').innerHTML = template_http.statusText;

            var var1Obj =
                doc.getElementsByTagName('TAGNAME1');
            var var2Obj = doc.getElementsByTagName('TAGNAME2');

            variable1 = cityObj.item(0).firstChild.data;
template_state = stateObj.item(0).firstChild.data;

            // update the form fields
            document.getElementById('variable1').innerHTML = variable1;
document.getElementById('variable2').innerHTML = variable2;
        }
        else {
            document.getElementById('template_status').innerHTML = "Loading...";
        }
    }
    else {
        document.getElementById('template_status').innerHTML = "Loading...";
    }
}

function template_showResults() {
window.alert(template_http.responseText);
}

// This function calls the web service directly. This Web service lives on a remote server, so there are security issues when running this method in Mozilla. IE prompts //if you would like to skip security and make the call.
function template_updateDirect() {
    var url = "http://www.xyz.com?query1="; // The server-side script
    var varValue =
        document.getElementById("template_var1Obj").value
    ;

    // Open a url connection using the XMLHttpObject.
    // The third parameter, set to false, specifies that the call should be made asynchronously.
    // Set this to true to make this call synchronous.
    template_http.open("GET", url + escape(varValue), false);

    // Set a callback handler to a local Javascript method
    template_http.onreadystatechange =
        template_handleHttpResponse;

    // The null value can be replaced with XML request data
    if the call is
    // SOAP-style call instead of HTTP request.
    template_http.send(null);
}
APPENDIX L – Zip code lookup JavaScript

// This is the request object used to run an AJAX style query
var zipcode_http = new XMLHttpRequest();
var zipcode_city, zipcode_state;

// This is a callback method used to handle returned data from a query.
// If you had multiple AJAX queries on this page you would make multiple
// handlers and name them uniquely.
function zipcode_handleHttpResponse() {
    if (zipcode_http.readyState == 4) {
        if (zipcode_http.status == 200) {
            var doc = (new DOMParser()).parseFromString(zipcode_http.responseText, "text/xml");
            document.getElementById('zipcode_status').innerHTML = zipcode_http.statusText;

            var cityObj = doc.getElementsByTagName('CITY');
            var stateObj = doc.getElementsByTagName('STATE');

            zipcode_city = cityObj.item(0).firstChild.data;
            zipcode_state = stateObj.item(0).firstChild.data;

            // Set the form fields with the new data
            document.getElementById('zipcode_city').innerHTML = zipcode_city;
            document.getElementById('zipcode_state').innerHTML = zipcode_state;
        } else {
            document.getElementById('zipcode_status').innerHTML = "Loading...";
        }
    } else {
        document.getElementById('zipcode_status').innerHTML = "Loading...";
    }
}

function zipcode_showResults() {
window.alert(zipcode_http.responseText);
}

// This is the method that is called when the Direct button is clicked.
// This function calls the web service directly. This web service lives on a remote
// server, so there are security issues when running this method in Mozilla. IE will
// ask you if you would like to skirt security and just make the call.
function zipcode_updateDirect() {
    var url =
        "http://www.webservicex.net/uszip.asmx/GetInfoByZIP?USZip=
    "; // The server-side script
    var zipValue =
        document.getElementById("zipcode_USZip").value;

    // Open a url connection using the XMLHttpObject. The third parameter specifies that the
    // call should be made asynchronously. Set this to false to make this call synchronous.
    zipcode_http.open("GET", url + escape(zipValue), false);

    // Set a callback handler to a local Javascript method
    zipcode_http.onreadystatechange =
        zipcode_handleHttpResponse;

    // Make the call. You can replace the null value with XML request data if you are doing
    // a SOAP-style call instead of using HTTP request parameters.
    zipcode_http.send(null);
}

// This is the method that is called when the Proxy button is clicked.
// This function calls the web service through a local proxy so that
// Mozilla's security mechanisms are not invoked. As far as the browser is
// concerned, all of the information is coming from one server.
function zipcode_updateProxy() {
    // This is the endpoint for the ZIP Code web service
var endurl = escape("http://www.webservicex.net/uszip.asmx/GetInfoByZIP");

// This is the endpoint of the Proxy servlet that will make the call on our behalf
var localurl = "/Portal/ProxyServlet";

// This is the input parameter for the ZIP Code
var zipValue = document.getElementById("zipcode_USZip").value;

// This is the actual URL that will be called using the XmlHttpRequest object
var url = localurl + "?url=" + endurl + ";USZip=" + zipValue; // The server-side script

// Open a url connection using the XMLHttpRequest. The third parameter specifies that the call should be made asynchronously. Set this to false to make this call synchronous.
zipcode_http.open("GET", url, true);

// Set a callback handler to a local Javascript method
zipcode_http.onreadystatechange = zipcode_handleHttpResponse;

// Make the call. You can replace the null value with XML request data if you are doing a SOAP-style call instead of using HTTP request parameters.
zipcode_http.send(null);
APPENDIX M – RealTime Stocks Web Service Control

// Stocks Web Service Control developed on BEA WebLogic platform
package controls;

/**
 * @jc:location http-url="RealTimeStockQuotesTest.jws" jms-url="RealTimeStockQuotesTest.jws"
 * @jc:wsdl file="#RealTimeStockQuotesTestWsdl"
 * @editor-info:link autogen-style="java"
 * source="RealTimeStockQuotesTest.jws"
 * autogen="true"
 */

class RealTimeStockQuotesTestControl extends
com.bea.control.ControlExtension,
com.bea.control.ServiceControl
{
    public static class SIRealTimeQuote
    implements java.io.Serializable
    {
        public java.lang.String Symbol;
        public java.lang.String CUSIP;
        public java.lang.String CIK;
        public java.lang.String Name;
        public java.lang.String Date;
        public java.lang.String Time;
        public double Last;
        public int Quantity;
        public double ChangeFromPrevious;
        public double PercentChangeFromPrevious;
        public double Open;
        public double ChangeFromOpen;
        public double PercentChangeFromOpen;
        public double Bid;
        public double Ask;
        public double Spread;
        public int BidQuantity;
        public int AskQuantity;
        public int Volume;
        public int ECNVolume;
        public double Highest;
        public double Lowest;
        public java.lang.String Rank;
    }
}

public static class SIExtendedRealTimeQuote
implements java.io.Serializable
{
    public java.lang.String Symbol;
    public java.lang.String CUSIP;
    public java.lang.String CIK;
    public java.lang.String Name;
    public double LastPrice;
    public double LastQuantity;
    public java.lang.String LastTradeType;
    public java.lang.String LastTradeDirection;
    public java.lang.String Date;
    public java.lang.String Time;
    public double Volume;
    public int ECNVolume;
    public double LastMarketClosePrice;
    public double ChangeFromLastMarketClose;
    public double PercentChangeFromLastMarketClose;
    public double LastClosePrice;
    public double ChangeFromLastClose;
    public double PercentChangeFromLastClose;
    public double OpenPrice;
    public double ChangeFromOpen;
    public double PercentChangeFromOpen;
    public double MarketOpenPrice;
    public double ChangeFromMarketOpen;
    public double PercentChangeFromMarketOpen;
    public double MarketClosePrice;
    public double ChangeFromMarketClose;
    public double PercentChangeFromMarketClose;
    public double PreviousTradePrice;
    public double ChangeFromPreviousTrade;
    public double PercentChangeFromPreviousTrade;
    public double BidPrice;
    public int BidQuantity;
    public java.lang.String BidTime;
    public int Bids;
    public int BidVolume;
    public double AverageBid;
    public double AskPrice;
    public int AskQuantity;
    public java.lang.String AskTime;
    public int Asks;
    public double AverageAsk;
    public int AskVolume;
    public double Spread;
    public double Highest;
    public java.lang.String HighestTime;
}
public double Lowest;
public java.lang.String LowestTime;
public int TotalOrders;
public int TotalCancellations;
public int TotalExecutions;
public long TotalQuantityOrdered;
public long TotalQuantityExecuted;
public long TotalQuantityCancelled;
public double AverageOrderSize;
public double AverageExecutionSize;
public long SharesOutstanding;
public double MarketCap;
}

public static class SITickerTape
    implements java.io.Serializable
{
    public java.lang.String Tape;
}

public static class SITime
    implements java.io.Serializable
{
    public java.lang.String Hour;
    public java.lang.String Minute;
    public java.lang.String AMPMType;
}

public static class SISingleTick
    implements java.io.Serializable
{
    public java.lang.String Symbol;
    public java.lang.String Time;
    public double Price;
    public int Quantity;
}

public static class SITicks
    implements java.io.Serializable
{
    public java.lang.String Symbol;
    public SITick[] Ticks1;
}

public static class SITick
    implements java.io.Serializable
{
public java.lang.String Time;
public double Price;
public int Quantity;
public double Change;
public double PercentChange;
public double High;
public double Low;
}

public static class SIBooleanResult
    implements java.io.Serializable
{
    public java.lang.String Symbol;
    public boolean Value;
}

public static class SIBook
    implements java.io.Serializable
{
    public java.lang.String Symbol;
    public SIOOrderSummary[] Bids;
    public SIOOrderSummary[] Asks;
}

public static class SIOOrderSummary
    implements java.io.Serializable
{
    public double Price;
    public int Ordered;
    public int Filled;
    public int Cancelled;
    public int Unfilled;
    public int Rank;
}

public static class SIDetailedBook
    implements java.io.Serializable
{
    public java.lang.String Symbol;
    public SIOOrder[] Bids;
    public SIOOrder[] Asks;
}

public static class SIOOrder
    implements java.io.Serializable
{
    public java.lang.String Reference;
}
public java.lang.String Time;
public double Price;
public int Ordered;
public int Filled;
public int Cancelled;
public int Unfilled;
public int Rank;
}

public static class SITop
    implements java.io.Serializable
{
    public java.lang.String Type;
    public SIRealTimeQuote[] Quotes;
}

public static class SISearchResult
    implements java.io.Serializable
{
    public double Change;
    public SIRealTimeQuote Quote;
    public SISecurity Security;
    public double SharesOutstanding;
    public double MarketCap;
    public double Last;
    public double LastClose;
    public double PercentChange;
    public double LastYearClose;
    public double YTDPercentChange;
    public double OneYearClose;
    public double OneYearPercentChange;
}

public static class SISecurity
    implements java.io.Serializable
{
    public java.lang.String CIK;
    public java.lang.String Cusip;
    public java.lang.String Symbol;
    public java.lang.String Name;
    public java.lang.String Market;
    public java.lang.String CategoryOrIndustry;
}
public SIRealTimeQuote GetRealQuote (java.lang.String Exchange, java.lang.String Symbol, boolean IncludeBidAsk);

public SIIExtendedRealTimeQuote GetExtendedRealQuote (java.lang.String Exchange, java.lang.String Symbol);

public SIRealTimeQuote[] GetRealQuotes (java.lang.String Exchange, java.lang.String Symbols, boolean IncludeBidAsk);

public SIIExtendedRealTimeQuote[] GetExtendedRealQuotes (java.lang.String Exchange, java.lang.String Symbols);

public SITickerTape GetRealQuotesTicker (java.lang.String Symbols);

/** *
 * @jc:protocol form-post="false" form-get="false"
 */
public SISingleTick GetTick (java.lang.String Exchange, java.lang.String Symbol, SITime Time);

/** *
 * @jc:protocol form-post="false" form-get="false"
 */
public SITicks GetTicks (java.lang.String Exchange, java.lang.String Symbol, SITime StartTime, SITime EndTime, java.lang.String TickPrecision, int TickPeriods);

/** *
 * @jc:protocol form-post="false" form-get="false"
 */
public SITicks GetHistoricalTicks (java.lang.String Exchange, java.lang.String Symbol, java.lang.String HistoricalPeriod, SITime StartTime, SITime EndTime, java.lang.String TickPrecision, int TickPeriods);

public SIBooleanResult IsTraded (java.lang.String Exchange, java.lang.String Symbol);

public SIBook GetBook (java.lang.String Exchange, java.lang.String Symbol, int MaximumOrders);
public SIDetailedBook GetBookDetails (java.lang.String Exchange, java.lang.String Symbol, int MaximumOrders);

public SITop GetRealTopMovers (java.lang.String Exchange, int Count);

public SITop GetRealTopGainers (java.lang.String Exchange, int Count);

public SITop GetRealTopLosers (java.lang.String Exchange, int Count);

public SISearchResult[] SearchByMarketCap (java.lang.String Exchange, double PercentChange, double MinimumMarketCap, double MaximumMarketCap);

public SITop[] GetRealTop (java.lang.String Exchange, int Count);

public void GetRemainingHits ();

static final long serialVersionUID = 1L;
}

/** @common:define name="RealTimeStockQuotesTestWsdl" value::
</?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://www.openuri.org/
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/
targetNamespace="http://www.openuri.org/">
<types>
<s:schema elementFormDefault="qualified"
   targetNamespace="http://www.openuri.org/
   xmlns:s="http://www.w3.org/2001/XMLSchema"
   xmlns:ope="http://www.openuri.org/">
<s:element name="GetRealQuote">
   <s:complexType>
      <s:sequence>
         <s:element name="Exchange" type="s:string" minOccurs="0"/>
         <s:element name="Symbol" type="s:string" minOccurs="0"/>
         <s:element name="IncludeBidAsk" type="s:boolean"/>
      </s:sequence>
   </s:complexType>
</s:element>
<s:element name="GetRealQuoteResponse">
   <s:complexType>
      <s:sequence>
         <s:element name="GetRealQuoteResult" type="ope:SIRealTimeQuote" minOccurs="0"/>
      </s:sequence>
   </s:complexType>
</s:element>
<s:element name="SIRealTimeQuote" nillable="true" type="ope:SIRealTimeQuote"/>
<s:element name="GetExtendedRealQuote">
   <s:complexType>
      <s:sequence>
         <s:element name="Exchange" type="s:string" minOccurs="0"/>
         <s:element name="Symbol" type="s:string" minOccurs="0"/>
      </s:sequence>
   </s:complexType>
</s:element>
<s:element name="GetExtendedRealQuoteResponse">
   <s:complexType>
      <s:sequence>
         <s:element name="GetExtendedRealQuoteResult" type="ope:SIExtendedRealTimeQuote" minOccurs="0"/>
      </s:sequence>
   </s:complexType>
</s:element>
</s:schema>
</types>
<s:element name="SIExtendedRealTimeQuote" nillable="true" type="ope:SIExtendedRealTimeQuote"/>
<s:element name="GetRealQuotes"
    <s:complexType>
        <s:sequence>
            <s:element name="Exchange" type="s:string" minOccurs="0"/>
            <s:element name="Symbols" type="s:string" minOccurs="0"/>
            <s:element name="IncludeBidAsk" type="s:boolean"/>
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="GetRealQuotesResponse">
    <s:complexType>
        <s:sequence>
            <s:element name="GetRealQuotesResult" type="ope:ArrayOfSIRealTimeQuote" minOccurs="0"/>
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="ArrayOfSIRealTimeQuote" nillable="true" type="ope:ArrayOfSIRealTimeQuote"/>
<s:element name="GetExtendedRealQuotes">
    <s:complexType>
        <s:sequence>
            <s:element name="Exchange" type="s:string" minOccurs="0"/>
            <s:element name="Symbols" type="s:string" minOccurs="0"/>
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="GetExtendedRealQuotesResponse">
    <s:complexType>
        <s:sequence>
            <s:element name="GetExtendedRealQuotesResult" type="ope:ArrayOfSIExtendedRealTimeQuote" minOccurs="0"/>
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="SIExtendedRealTimeQuote" nillable="true" type="ope:SIExtendedRealTimeQuote"/>
<s:element name="GetRealQuotes">
    <s:complexType>
        <s:sequence>
            <s:element name="Exchange" type="s:string" minOccurs="0"/>
            <s:element name="Symbols" type="s:string" minOccurs="0"/>
            <s:element name="IncludeBidAsk" type="s:boolean"/>
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="GetRealQuotesResponse">
    <s:complexType>
        <s:sequence>
            <s:element name="GetRealQuotesResult" type="ope:ArrayOfSIRealTimeQuote" minOccurs="0"/>
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="ArrayOfSIRealTimeQuote" nillable="true" type="ope:ArrayOfSIRealTimeQuote"/>
<s:element name="GetExtendedRealQuotes">
    <s:complexType>
        <s:sequence>
            <s:element name="Exchange" type="s:string" minOccurs="0"/>
            <s:element name="Symbols" type="s:string" minOccurs="0"/>
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="GetExtendedRealQuotesResponse">
    <s:complexType>
        <s:sequence>
            <s:element name="GetExtendedRealQuotesResult" type="ope:ArrayOfSIExtendedRealTimeQuote" minOccurs="0"/>
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="ArrayOfSIExtendedRealTimeQuote" nillable="true"
type="ope:ArrayOfSIExtendedRealTimeQuote"/>
<s:element name="GetRealQuotesTicker">
  <s:complexType>
    <s:sequence>
      <s:element name="Symbols" type="s:string" minOccurs="0"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="GetRealQuotesTickerResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="GetRealQuotesTickerResult" type="ope:SITickerTape" minOccurs="0"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="SITickerTape" nillable="true" type="ope:SITickerTape"/>
<s:element name="GetTick">
  <s:complexType>
    <s:sequence>
      <s:element name="Exchange" type="s:string" minOccurs="0"/>
      <s:element name="Symbol" type="s:string" minOccurs="0"/>
      <s:element name="Time" type="ope:SITime" minOccurs="0"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="GetTickResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="GetTickResult" type="ope:SISingleTick" minOccurs="0"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="GetTicks">
  <s:complexType>
    <s:sequence>
      <s:element name="Exchange" type="s:string" minOccurs="0"/>
    </s:sequence>
  </s:complexType>
</s:element>

<s:element name="Symbol" type="s:string" minOccurs="0"/>
  <s:element name="StartTime" type="ope:SITime" minOccurs="0"/>
  <s:element name="EndTime" type="ope:SITime" minOccurs="0"/>
  <s:element name="TickPrecision" type="s:string" minOccurs="0"/>
  <s:element name="TickPeriods" type="s:int"/>
</s:complexType>
</s:element>
<s:element name="GetTicksResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="GetTicksResult" type="ope:SITicks" minOccurs="0"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="GetHistoricalTicks">
  <s:complexType>
    <s:sequence>
      <s:element name="Exchange" type="s:string" minOccurs="0"/>
      <s:element name="Symbol" type="s:string" minOccurs="0"/>
      <s:element name="HistoricalPeriod" type="s:string" minOccurs="0"/>
      <s:element name="StartTime" type="ope:SITime" minOccurs="0"/>
      <s:element name="EndTime" type="ope:SITime" minOccurs="0"/>
      <s:element name="TickPrecision" type="s:string" minOccurs="0"/>
      <s:element name="TickPeriods" type="s:int"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="GetHistoricalTicksResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="GetHistoricalTicksResult" type="ope:SITicks" minOccurs="0"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:complexType>
<s:element name="IsTraded">
    <s:complexType>
        <s:sequence>
            <s:element name="Exchange" type="s:string" minOccurs="0"/>
            <s:element name="Symbol" type="s:string" minOccurs="0"/>
        </s:sequence>
    </s:complexType>
</s:element>
</s:complexType>
</s:element>
<s:element name="IsTradedResponse">
    <s:complexType>
        <s:sequence>
            <s:element name="IsTradedResult" type="ope:SIBooleanResult" minOccurs="0"/>
        </s:sequence>
    </s:complexType>
</s:element>
</s:complexType>
</s:element>
<s:element name="SIBooleanResult" nillable="true" type="ope:SIBooleanResult"/>
<s:element name="GetBook">
    <s:complexType>
        <s:sequence>
            <s:element name="Exchange" type="s:string" minOccurs="0"/>
            <s:element name="Symbol" type="s:string" minOccurs="0"/>
            <s:element name="MaximumOrders" type="s:int"/>
        </s:sequence>
    </s:complexType>
</s:element>
</s:complexType>
</s:element>
<s:element name="GetBookResponse">
    <s:complexType>
        <s:sequence>
            <s:element name="GetBookResult" type="ope:SIBook" minOccurs="0"/>
        </s:sequence>
    </s:complexType>
</s:element>
</s:complexType>
</s:element>
<s:element name="SIBook" nillable="true" type="ope:SIBook"/>
<s:element name="GetBookDetails">
    <s:complexType>
        <s:sequence>
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="Exchange" type="s:string" minOccurs="0"/>
<s:element name="Symbol" type="s:string" minOccurs="0"/>
<s:element name="MaximumOrders" type="s:int"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="GetBookDetailsResponse">
<s:complexType>
<s:sequence>
<s:element name="GetBookDetailsResult" type="ope:SIDetailedBook" minOccurs="0"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="SIDetailedBook" nillable="true" type="ope:SIDetailedBook"/>
<s:element name="GetRealTopMovers">
<s:complexType>
<s:sequence>
<s:element name="Exchange" type="s:string" minOccurs="0"/>
<s:element name="Count" type="s:int"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="GetRealTopMoversResponse">
<s:complexType>
<s:sequence>
<s:element name="GetRealTopMoversResult" type="ope:SITop" minOccurs="0"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="SITop" nillable="true" type="ope:SITop"/>
<s:element name="GetRealTopGainers">
<s:complexType>
<s:sequence>
<s:element name="Exchange" type="s:string" minOccurs="0"/>
<s:element name="Count" type="s:int"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="SITop" nillable="true" type="ope:SITop"/>
<s:element name="GetRealTopGainersResponse">
<s:complexType>
<s:sequence>
<s:element name="GetRealTopGainersResult" type="ope:SITop" minOccurs="0"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="GetRealTopGainersResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="GetRealTopGainersResult" type="ope:SITop" minOccurs="0"/>
    </s:sequence>
  </s:complexType>
</s:element>

<s:element name="GetRealTopLosers">
  <s:complexType>
    <s:sequence>
      <s:element name="Exchange" type="s:string" minOccurs="0"/>
      <s:element name="Count" type="s:int"/>
    </s:sequence>
  </s:complexType>
</s:element>

<s:element name="GetRealTopLosersResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="GetRealTopLosersResult" type="ope:SITop" minOccurs="0"/>
    </s:sequence>
  </s:complexType>
</s:element>

<s:element name="SearchByMarketCap">
  <s:complexType>
    <s:sequence>
      <s:element name="Exchange" type="s:string" minOccurs="0"/>
      <s:element name="PercentChange" type="s:double"/>
      <s:element name="MinimumMarketCap" type="s:double"/>
      <s:element name="MaximumMarketCap" type="s:double"/>
    </s:sequence>
  </s:complexType>
</s:element>

<s:element name="SearchByMarketCapResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="SearchByMarketCapResult" type="ope:ArrayOfSISearchResult" minOccurs="0"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="ArrayOfSIISearchResult" nillable="true" type="ope:ArrayOfSIISearchResult"/>
<s:element name="GetRealTop">
    <s:complexType>
        <s:sequence>
            <s:element name="Exchange" type="s:string" minOccurs="0"/>
            <s:element name="Count" type="s:int"/>
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="GetRealTopResponse">
    <s:complexType>
        <s:sequence>
            <s:element name="GetRealTopResult" type="ope:ArrayOfSITop" minOccurs="0"/>
        </s:sequence>
    </s:complexType>
</s:element>
<s:element name="ArrayOfSITop" nillable="true" type="ope:ArrayOfSITop"/>
<s:element name="GetRemainingHits">
    <s:complexType>
        <s:sequence/>
    </s:complexType>
</s:element>
<s:element name="GetRemainingHitsResponse">
    <s:complexType>
        <s:sequence/>
    </s:complexType>
</s:element>
<s:complexType name="SIRealTimeQuote">
    <s:sequence>
        <s:element name="Symbol" type="s:string" minOccurs="0"/>
        <s:element name="CUSIP" type="s:string" minOccurs="0"/>
        <s:element name="CIK" type="s:string" minOccurs="0"/>
        <s:element name="Name" type="s:string" minOccurs="0"/>
        <s:element name="Date" type="s:string" minOccurs="0"/>
        <s:element name="Time" type="s:string" minOccurs="0"/>
        <s:element name="Last" type="s:double"/>
    </s:sequence>
</s:complexType>
<s:element name="LastMarketClosePrice" type="s:double"/>
<s:element name="ChangeFromLastMarketClose" type="s:double"/>
<s:element name="PercentChangeFromLastMarketClose" type="s:double"/>
<s:element name="LastClosePrice" type="s:double"/>
<s:element name="ChangeFromLastClose" type="s:double"/>
<s:element name="PercentChangeFromLastClose" type="s:double"/>
<s:element name="OpenPrice" type="s:double"/>
<s:element name="ChangeFromOpen" type="s:double"/>
<s:element name="PercentChangeFromOpen" type="s:double"/>
<s:element name="MarketOpenPrice" type="s:double"/>
<s:element name="ChangeFromMarketOpen" type="s:double"/>
<s:element name="PercentChangeFromMarketOpen" type="s:double"/>
<s:element name="MarketClosePrice" type="s:double"/>
<s:element name="ChangeFromMarketClose" type="s:double"/>
<s:element name="PercentChangeFromMarketClose" type="s:double"/>
<s:element name="PreviousTradePrice" type="s:double"/>
<s:element name="ChangeFromPreviousTrade" type="s:double"/>
<s:element name="PercentChangeFromPreviousTrade" type="s:double"/>
<s:element name="BidPrice" type="s:double"/>
<s:element name="BidQuantity" type="s:int"/>
<s:element name="BidTime" type="s:string" minOccurs="0"/>
<s:element name="Bids" type="s:int"/>
<s:element name="BidVolume" type="s:int"/>
<s:element name="AverageBid" type="s:double"/>
<s:element name="AskPrice" type="s:double"/>
<s:element name="AskQuantity" type="s:int"/>
<s:element name="AskTime" type="s:string" minOccurs="0"/>
<s:element name="Asks" type="s:int"/>
<s:element name="AverageAsk" type="s:double"/>
<s:element name="AskVolume" type="s:int"/>
<s:element name="Spread" type="s:double"/>
<s:element name="Highest" type="s:double"/>
<s:element name="HighestTime" type="s:string" minOccurs="0"/>
<s:element name="Lowest" type="s:double"/>
<s:element name="LowestTime" type="s:string" minOccurs="0"/>
<s:element name="TotalOrders" type="s:int"/>
<s:element name="TotalCancellations" type="s:int"/>
<s:element name="TotalExecutions" type="s:int"/>
<s:element name="TotalQuantityOrdered" type="s:long"/>
<s:element name="TotalQuantityExecuted" type="s:long"/>
<s:element name="TotalQuantityCancelled" type="s:long"/>
<s:element name="AverageOrderSize" type="s:double"/>
<s:element name="AverageExecutionSize" type="s:double"/>
<s:element name="SharesOutstanding" type="s:long"/>
<s:element name="MarketCap" type="s:double"/>
</s:sequence>
</s:complexType>
<s:complexType name="ArrayOfSIRealTimeQuote">
<s:sequence>
<s:element name="SIRealTimeQuote" type="ope:SIRealTimeQuote" nillable="true" minOccurs="0" maxOccurs="unbounded"/>
</s:sequence>
</s:complexType>
<s:complexType name="ArrayOfSIExtendedRealTimeQuote">
<s:sequence>
<s:element name="SIExtendedRealTimeQuote" type="ope:SIExtendedRealTimeQuote"/>
</s:sequence>
</s:complexType>

95
<s:complexType name="SITicks">
    <s:sequence>
        <s:element name="Symbol" type="s:string" minOccurs="0"/>
        <s:element name="Ticks1" type="ope:ArrayOfSITick" minOccurs="0"/>
    </s:sequence>
</s:complexType>

<s:complexType name="ArrayOfSITick">
    <s:sequence>
        <s:element name="SITick" type="ope:SITick" nillable="true" minOccurs="0" maxOccurs="unbounded"/>
    </s:sequence>
</s:complexType>

<s:complexType name="SITick">
    <s:sequence>
        <s:element name="Symbol" type="s:string" minOccurs="0"/>
        <s:element name="Time" type="s:string" minOccurs="0"/>
        <s:element name="Price" type="s:double"/>
        <s:element name="Quantity" type="s:int"/>
    </s:sequence>
</s:complexType>

<s:complexType name="SISingleTick">
    <s:sequence>
        <s:element name="Symbol" type="s:string" minOccurs="0"/>
        <s:element name="Time" type="s:string" minOccurs="0"/>
        <s:element name="Price" type="s:double"/>
        <s:element name="Quantity" type="s:int"/>
    </s:sequence>
</s:complexType>

<s:complexType name="SITime">
    <s:sequence>
        <s:element name="Hour" type="s:string" minOccurs="0"/>
        <s:element name="Minute" type="s:string" minOccurs="0"/>
        <s:element name="AMPMType" type="s:string" minOccurs="0"/>
    </s:sequence>
</s:complexType>

<s:complexType name="SITickerTape">
    <s:sequence>
        <s:element name="Tape" type="s:string" minOccurs="0"/>
    </s:sequence>
</s:complexType>

<s:complexType name="SITicker">
    <s:sequence>
        <s:element name="Symbol" type="s:string" minOccurs="0" maxOccurs="unbounded"/>
    </s:sequence>
</s:complexType>
<s:element name="Symbol" type="s:string" minOccurs="0"/>
<s:element name="Bids" type="ope:ArrayOfSIOrder" minOccurs="0"/>
<s:element name="Asks" type="ope:ArrayOfSIOrder" minOccurs="0"/>
</s:sequence></s:complexType>
<s:complexType name="ArrayOfSIOrder">
<s:sequence>
<s:element name="SIOrder" type="ope:SIOrder" nillable="true" minOccurs="0" maxOccurs="unbounded"/>
</s:sequence></s:complexType>
<s:complexType name="SIOrder">
<s:sequence>
<s:element name="Reference" type="s:string" minOccurs="0"/>
<s:element name="Time" type="s:string" minOccurs="0"/>
<s:element name="Price" type="s:double"/>
<s:element name="Ordered" type="s:int"/>
<s:element name="Filled" type="s:int"/>
<s:element name="Cancelled" type="s:int"/>
<s:element name="Unfilled" type="s:int"/>
<s:element name="Rank" type="s:int"/>
</s:sequence></s:complexType>
<s:complexType name="SITop">
<s:sequence>
<s:element name="Type" type="s:string" minOccurs="0"/>
<s:element name="Quotes" type="ope:ArrayOfSIRealTimeQuote" minOccurs="0"/>
</s:sequence></s:complexType>
<s:complexType name="ArrayOfSISearchResult">
<s:sequence>
<s:element name="SISearchResult" type="ope:SISearchResult" nillable="true" minOccurs="0" maxOccurs="unbounded"/>
</s:sequence></s:complexType>
<s:complexType name="SISearchResult">
<s:sequence>
<s:element name="Change" type="s:double"/>
</s:sequence></s:complexType>

98
<message name="GetRealQuoteSoapIn">
   <part name="parameters" element="s0:GetRealQuote"/>
</message>

<message name="GetRealQuoteSoapOut">
   <part name="parameters" element="s0:GetRealQuoteResponse"/>
</message>

<message name="GetExtendedRealQuoteSoapIn">
   <part name="parameters" element="s0:GetExtendedRealQuote"/>
</message>

<message name="GetExtendedRealQuoteSoapOut">
   <part name="parameters" element="s0:GetExtendedRealQuoteResponse"/>
</message>

<message name="GetRealQuotesSoapIn">
   <part name="parameters" element="s0:GetRealQuotes"/>
</message>

<message name="GetRealQuotesSoapOut">
   <part name="parameters" element="s0:GetRealQuotesResponse"/>
</message>

<message name="GetExtendedRealQuotesSoapIn">
   <part name="parameters" element="s0:GetExtendedRealQuotes"/>
</message>

<message name="GetExtendedRealQuotesSoapOut">
   <part name="parameters" element="s0:GetExtendedRealQuotesResponse"/>
</message>

<message name="GetRealQuotesTickerSoapIn">
   <part name="parameters" element="s0:GetRealQuotesTicker"/>
</message>

<message name="GetRealQuotesTickerSoapOut">
   <part name="parameters" element="s0:GetRealQuotesTickerResponse"/>
</message>

<message name="GetTickSoapIn">
   <part name="parameters" element="s0:GetTick"/>
</message>

<message name="GetTickSoapOut">
   <part name="parameters" element="s0:GetTickResponse"/>
</message>

<message name="GetTicksSoapIn">
   <part name="parameters" element="s0:GetTicks"/>
</message>

100
</message>
<message name="GetBookDetailsHttpGetOut">
  <part name="Body" element="s0:SIDetailedBook"/>
</message>
<message name="GetRealTopMoversHttpGetIn">
  <part name="Exchange" type="s:string"/>
  <part name="Count" type="s:string"/>
</message>
<message name="GetRealTopMoversHttpGetOut">
  <part name="Body" element="s0:SITop"/>
</message>
<message name="GetRealTopGainersHttpGetIn">
  <part name="Exchange" type="s:string"/>
  <part name="Count" type="s:string"/>
</message>
<message name="GetRealTopGainersHttpGetOut">
  <part name="Body" element="s0:SITop"/>
</message>
<message name="GetRealTopLosersHttpGetIn">
  <part name="Exchange" type="s:string"/>
  <part name="Count" type="s:string"/>
</message>
<message name="GetRealTopLosersHttpGetOut">
  <part name="Body" element="s0:SITop"/>
</message>
<message name="SearchByMarketCapHttpGetIn">
  <part name="Exchange" type="s:string"/>
  <part name="PercentChange" type="s:string"/>
  <part name="MinimumMarketCap" type="s:string"/>
  <part name="MaximumMarketCap" type="s:string"/>
</message>
<message name="SearchByMarketCapHttpGetOut">
  <part name="Body" element="s0:ArrayOfSISearchResult"/>
</message>
<message name="GetRealTopHttpGetIn">
  <part name="Exchange" type="s:string"/>
  <part name="Count" type="s:string"/>
</message>
<message name="GetRealTopHttpGetOut">
  <part name="Body" element="s0:ArrayOfSITop"/>
</message>
<message name="GetRemainingHitsHttpGetIn"/>
<message name="GetRemainingHitsHttpGetOut"/>
<message name="GetRealQuoteHttpPostIn">
  <part name="Exchange" type="s:string"/>
  <part name="Symbol" type="s:string"/>
</message>
<message name="GetRealQuoteHttpPostOut">
  <part name="Exchange" type="s:string"/>
  <part name="Symbol" type="s:string"/>
</message>
<part name="IncludeBidAsk" type="string"/>
</message>
<message name="GetRealQuoteHttpPostOut">
  <part name="Body" element="SIRealTimeQuote"/>
</message>
<message name="GetExtendedRealQuoteHttpPostIn">
  <part name="Exchange" type="string"/>
  <part name="Symbol" type="string"/>
</message>
<message name="GetExtendedRealQuoteHttpPostOut">
  <part name="Body" element="SIExtendedRealTimeQuote"/>
</message>
<message name="GetRealQuotesHttpPostIn">
  <part name="Exchange" type="string"/>
  <part name="Symbols" type="string"/>
  <part name="IncludeBidAsk" type="string"/>
</message>
<message name="GetRealQuotesHttpPostOut">
  <part name="Body" element="ArrayOfSIRealTimeQuote"/>
</message>
<message name="GetExtendedRealQuotesHttpPostIn">
  <part name="Exchange" type="string"/>
  <part name="Symbols" type="string"/>
</message>
<message name="GetExtendedRealQuotesHttpPostOut">
  <part name="Body" element="ArrayOfSIExtendedRealTimeQuote"/>
</message>
<message name="GetRealQuotesTickerHttpPostIn">
  <part name="Symbols" type="string"/>
</message>
<message name="GetRealQuotesTickerHttpPostOut">
  <part name="Body" element="SITickerTape"/>
</message>
<message name="IsTradedHttpPostIn">
  <part name="Exchange" type="string"/>
  <part name="Symbol" type="string"/>
</message>
<message name="IsTradedHttpPostOut">
  <part name="Body" element="SIBooleanResult"/>
</message>
<message name="GetBookHttpPostIn">
  <part name="Exchange" type="string"/>
  <part name="Symbol" type="string"/>
  <part name="MaximumOrders" type="string"/>
</message>
</message>
<message name="GetBookHttpPostOut">
    <part name="Body" element="s0:SIBook"/>
</message>
<message name="GetBookDetailsHttpPostIn">
    <part name="Exchange" type="s:string"/>
    <part name="Symbol" type="s:string"/>
    <part name="MaximumOrders" type="s:string"/>
</message>
<message name="GetBookDetailsHttpPostOut">
    <part name="Body" element="s0:SIDetailedBook"/>
</message>
<message name="GetRealTopMoversHttpPostIn">
    <part name="Exchange" type="s:string"/>
    <part name="Count" type="s:string"/>
</message>
<message name="GetRealTopMoversHttpPostOut">
    <part name="Body" element="s0:SITop"/>
</message>
<message name="GetRealTopGainersHttpPostIn">
    <part name="Exchange" type="s:string"/>
    <part name="Count" type="s:string"/>
</message>
<message name="GetRealTopGainersHttpPostOut">
    <part name="Body" element="s0:SITop"/>
</message>
<message name="GetRealTopLosersHttpPostIn">
    <part name="Exchange" type="s:string"/>
    <part name="Count" type="s:string"/>
</message>
<message name="GetRealTopLosersHttpPostOut">
    <part name="Body" element="s0:SITop"/>
</message>
<message name="SearchByMarketCapHttpPostIn">
    <part name="Exchange" type="s:string"/>
    <part name="PercentChange" type="s:string"/>
    <part name="MinimumMarketCap" type="s:string"/>
    <part name="MaximumMarketCap" type="s:string"/>
</message>
<message name="SearchByMarketCapHttpPostOut">
    <part name="Body" element="s0:ArrayOfSISearchResult"/>
</message>
<message name="GetRealTopHttpPostIn">
    <part name="Exchange" type="s:string"/>
    <part name="Count" type="s:string"/>
</message>
<message name="GetRealTopHttpPostOut">
    </message>
<message name="GetRealTopHttpPostOut">
  <part name="Body" element="s0:ArrayOfSITop"/>
</message>
<message name="GetRemainingHitsHttpPostIn"/>
<message name="GetRemainingHitsHttpPostOut"/>
<portType name="RealTimeStockQuotesTestSoap">
  <operation name="GetRealQuote">
    <input message="s0:GetRealQuoteSoapIn"/>
    <output message="s0:GetRealQuoteSoapOut"/>
  </operation>
  <operation name="GetExtendedRealQuote">
    <input message="s0:GetExtendedRealQuoteSoapIn"/>
    <output message="s0:GetExtendedRealQuoteSoapOut"/>
  </operation>
  <operation name="GetRealQuotes">
    <input message="s0:GetRealQuotesSoapIn"/>
    <output message="s0:GetRealQuotesSoapOut"/>
  </operation>
  <operation name="GetExtendedRealQuotes">
    <input message="s0:GetExtendedRealQuotesSoapIn"/>
    <output message="s0:GetExtendedRealQuotesSoapOut"/>
  </operation>
  <operation name="GetRealQuotesTicker">
    <input message="s0:GetRealQuotesTickerSoapIn"/>
    <output message="s0:GetRealQuotesTickerSoapOut"/>
  </operation>
  <operation name="GetTick">
    <input message="s0:GetTickSoapIn"/>
    <output message="s0:GetTickSoapOut"/>
  </operation>
  <operation name="GetTicks">
    <input message="s0:GetTicksSoapIn"/>
    <output message="s0:GetTicksSoapOut"/>
  </operation>
  <operation name="GetHistoricalTicks">
    <input message="s0:GetHistoricalTicksSoapIn"/>
    <output message="s0:GetHistoricalTicksSoapOut"/>
  </operation>
  <operation name="IsTraded">
    <input message="s0:IsTradedSoapIn"/>
    <output message="s0:IsTradedSoapOut"/>
  </operation>
  <operation name="GetBook">
    <input message="s0:GetBookSoapIn"/>
    <output message="s0:GetBookSoapOut"/>
  </operation>
</portType>
<operation>
<operation name="GetBookDetails">
  <input message="s0:GetBookDetailsSoapIn"/>
  <output message="s0:GetBookDetailsSoapOut"/>
</operation>
</operation>
<operation name="GetRealTopMovers">
  <input message="s0:GetRealTopMoversSoapIn"/>
  <output message="s0:GetRealTopMoversSoapOut"/>
</operation>
</operation>
<operation name="GetRealTopGainers">
  <input message="s0:GetRealTopGainersSoapIn"/>
  <output message="s0:GetRealTopGainersSoapOut"/>
</operation>
</operation>
<operation name="GetRealTopLosers">
  <input message="s0:GetRealTopLosersSoapIn"/>
  <output message="s0:GetRealTopLosersSoapOut"/>
</operation>
</operation>
<operation name="SearchByMarketCap">
  <input message="s0:SearchByMarketCapSoapIn"/>
  <output message="s0:SearchByMarketCapSoapOut"/>
</operation>
</operation>
<operation name="GetRealTop">
  <input message="s0:GetRealTopSoapIn"/>
  <output message="s0:GetRealTopSoapOut"/>
</operation>
</operation>
<operation name="GetRemainingHits">
  <input message="s0:GetRemainingHitsSoapIn"/>
  <output message="s0:GetRemainingHitsSoapOut"/>
</operation>
</operation>
</portType>
<portType name="RealTimeStockQuotesTestHttpGet">
<operation name="GetRealQuote">
  <input message="s0:GetRealQuoteHttpGetIn"/>
  <output message="s0:GetRealQuoteHttpGetOut"/>
</operation>
</operation>
<operation name="GetExtendedRealQuote">
  <input message="s0:GetExtendedRealQuoteHttpGetIn"/>
  <output message="s0:GetExtendedRealQuoteHttpGetOut"/>
</operation>
</operation>
<operation name="GetRealQuotes">
  <input message="s0:GetRealQuotesHttpGetIn"/>
  <output message="s0:GetRealQuotesHttpGetOut"/>
</operation>
</operation>
<operation name="GetExtendedRealQuotes">

<input
message="s0:GetExtendedRealQuotesHttpGetIn"/>
<output
message="s0:GetExtendedRealQuotesHttpGetOut"/>
</operation>
<operation name="GetRealQuotesTicker">
<input
message="s0:GetRealQuotesTickerHttpGetIn"/>
<output
message="s0:GetRealQuotesTickerHttpGetOut"/>
</operation>
<operation name="IsTraded">
<input message="s0:IsTradedHttpGetIn"/>
<output message="s0:IsTradedHttpGetOut"/>
</operation>
<operation name="GetBook">
<input message="s0:GetBookHttpGetIn"/>
<output message="s0:GetBookHttpGetOut"/>
</operation>
<operation name="GetBookDetails">
<input message="s0:GetBookDetailsHttpGetIn"/>
<output message="s0:GetBookDetailsHttpGetOut"/>
</operation>
<operation name="GetRealTopMovers">
<input message="s0:GetRealTopMoversHttpGetIn"/>
<output message="s0:GetRealTopMoversHttpGetOut"/>
</operation>
<operation name="GetRealTopGainers">
<input message="s0:GetRealTopGainersHttpGetIn"/>
<output message="s0:GetRealTopGainersHttpGetOut"/>
</operation>
<operation name="GetRealTopLosers">
<input message="s0:GetRealTopLosersHttpGetIn"/>
<output message="s0:GetRealTopLosersHttpGetOut"/>
</operation>
<operation name="SearchByMarketCap">
<input message="s0:SearchByMarketCapHttpGetIn"/>
<output message="s0:SearchByMarketCapHttpGetOut"/>
</operation>
<operation name="GetRealTop">
<input message="s0:GetRealTopHttpGetIn"/>
<output message="s0:GetRealTopHttpGetOut"/>
</operation>
<operation name="GetRemainingHits">
<input message="s0:GetRemainingHitsHttpGetIn"/>
<output message="s0:GetRemainingHitsHttpGetOut"/>
</operation>
</portType>
<portType name="RealTimeStockQuotesTestHttpPost">
<operation name="GetRealQuote">
<input message="s0:GetRealQuoteHttpPostIn"/>
<output message="s0:GetRealQuoteHttpPostOut"/>
</operation>
<operation name="GetExtendedRealQuote">
<input message="s0:GetExtendedRealQuoteHttpPostIn"/>
<output message="s0:GetExtendedRealQuoteHttpPostOut"/>
</operation>
<operation name="GetRealQuotes">
<input message="s0:GetRealQuotesHttpPostIn"/>
<output message="s0:GetRealQuotesHttpPostOut"/>
</operation>
<operation name="GetExtendedRealQuotes">
<input message="s0:GetExtendedRealQuotesHttpPostIn"/>
<output message="s0:GetExtendedRealQuotesHttpPostOut"/>
</operation>
<operation name="GetRealQuotesTicker">
<input message="s0:GetRealQuotesTickerHttpPostIn"/>
<output message="s0:GetRealQuotesTickerHttpPostOut"/>
</operation>
<operation name="IsTraded">
<input message="s0:IsTradedHttpPostIn"/>
<output message="s0:IsTradedHttpPostOut"/>
</operation>
<operation name="GetBook">
<input message="s0:GetBookHttpPostIn"/>
<output message="s0:GetBookHttpPostOut"/>
</operation>
<operation name="GetBookDetails">
<input message="s0:GetBookDetailsHttpPostIn"/>
<output message="s0:GetBookDetailsHttpPostOut"/>
</operation>
<operation name="GetRealTopMovers">
<input message="s0:GetRealTopMoversHttpPostIn"/>
<output message="s0:GetRealTopMoversHttpPostOut"/>
</operation>
</portType>
<operation name="GetRealTopGainers">
  <input message="s0:GetRealTopGainersHttpPostIn"/>
  <output
message="s0:GetRealTopGainersHttpPostOut"/>
</operation>

<operation name="GetRealTopLosers">
  <input message="s0:GetRealTopLosersHttpPostIn"/>
  <output
message="s0:GetRealTopLosersHttpPostOut"/>
</operation>

<operation name="SearchByMarketCap">
  <input message="s0:SearchByMarketCapHttpPostIn"/>
  <output
message="s0:SearchByMarketCapHttpPostOut"/>
</operation>

<operation name="GetRealTop">
  <input message="s0:GetRealTopHttpPostIn"/>
  <output message="s0:GetRealTopHttpPostOut"/>
</operation>

<operation name="GetRemainingHits">
  <input message="s0:GetRemainingHitsHttpPostIn"/>
  <output
message="s0:GetRemainingHitsHttpPostOut"/>
</operation>
</portType>

<binding name="RealTimeStockQuotesTestSoap"
type="s0:RealTimeStockQuotesTestSoap">  
<soap:binding
transport="http://schemas.xmlsoap.org/soap/http"
style="document"/>

<operation name="GetRealQuote">
  <soap:operation
soapAction="http://www.openuri.org/GetRealQuote"
style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>

<operation name="GetExtendedRealQuote">
  <soap:operation
soapAction="http://www.openuri.org/GetExtendedRealQuote"
style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
</operation>
</binding>
<input>
<output>
  <soap:body use="literal"/>
</output>
</operation>
<operation name="GetRealQuotes">
  <soap:operation
    soapAction="http://www.openuri.org/GetRealQuotes"
    style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="GetExtendedRealQuotes">
  <soap:operation
    soapAction="http://www.openuri.org/GetExtendedRealQuotes" style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="GetRealQuotesTicker">
  <soap:operation
    soapAction="http://www.openuri.org/GetRealQuotesTicker" style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="GetTick">
  <soap:operation
    soapAction="http://www.openuri.org/GetTick"
    style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="GetTicks">
  <soap:operation
    soapAction="http://www.openuri.org/GetTicks"
    style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="GetHistoricalTicks">
  <soap:operation
    soapAction="http://www.openuri.org/GetHistoricalTicks"
    style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="IsTraded">
  <soap:operation
    soapAction="http://www.openuri.org/IsTraded"
    style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="GetBook">
  <soap:operation
    soapAction="http://www.openuri.org/GetBook"
    style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="GetBookDetails">
  <soap:operation
    soapAction="http://www.openuri.org/GetBookDetails"
    style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<soap:operation>
  soapAction="http://www.openuri.org/GetBookDetails"
  style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>

<operation name="GetRealTopMovers">
  <soap:operation>
    soapAction="http://www.openuri.org/GetRealTopMovers"
    style="document"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</operation>

<operation name="GetRealTopGainers">
  <soap:operation>
    soapAction="http://www.openuri.org/GetRealTopGainers"
    style="document"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</operation>

<operation name="GetRealTopLosers">
  <soap:operation>
    soapAction="http://www.openuri.org/GetRealTopLosers"
    style="document"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</operation>

<operation name="SearchByMarketCap">
  <soap:operation>
    soapAction="http://www.openuri.org/SearchByMarketCap"
    style="document"/>
  </soap:operation>
</operation>
<input>
  <soap:body use="literal"/>
</input>
<output>
  <soap:body use="literal"/>
</output>
</operation>
<operation name="GetRealTop">
  <soap:operation
      soapAction="http://www.openuri.org/GetRealTop"
      style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="GetRemainingHits">
  <soap:operation
      soapAction="http://www.openuri.org/GetRemainingHits"
      style="document"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
</binding>
(binding name="RealTimeStockQuotesTestHttpGet"
type="s0:RealTimeStockQuotesTestHttpGet">
  <http:binding verb="GET"/>
  <operation name="GetRealQuote">
    <http:operation location="/GetRealQuote"/>
    <input>
      <http:urlEncoded/>
    </input>
    <output>
      <mime:mimeXml part="Body"/>
    </output>
  </operation>
  <operation name="GetExtendedRealQuote">
    <http:operation
      location="/GetExtendedRealQuote"/>
    <input>
      <http:urlEncoded/>
    </input>
    <output>
      <mime:mimeXml part="Body"/>
    </output>
  </operation>
</binding>
<operation name="GetRealQuotes">
    <http:operation location="/GetRealQuotes"/>
    <input>
        <http:urlEncoded/>
    </input>
    <output>
        <mime:mimeXml part="Body"/>
    </output>
</operation>

<operation name="GetExtendedRealQuotes">
    <http:operation location="/GetExtendedRealQuotes"/>
    <input>
        <http:urlEncoded/>
    </input>
    <output>
        <mime:mimeXml part="Body"/>
    </output>
</operation>

<operation name="GetRealQuotesTicker">
    <http:operation location="/GetRealQuotesTicker"/>
    <input>
        <http:urlEncoded/>
    </input>
    <output>
        <mime:mimeXml part="Body"/>
    </output>
</operation>

<operation name="IsTraded">
    <http:operation location="/IsTraded"/>
    <input>
        <http:urlEncoded/>
    </input>
    <output>
        <mime:mimeXml part="Body"/>
    </output>
</operation>

<operation name="GetBook">
    <http:operation location="/GetBook"/>
    <input>
        <http:urlEncoded/>
    </input>
</operation>
<operation name="GetBookDetails">
  <http:operation location="/GetBookDetails"/>
  <input>
    <http:urlEncoded/>
  </input>
  <output>
    <mime:mimeXml part="Body"/>
  </output>
</operation>

<operation name="GetRealTopMovers">
  <http:operation location="/GetRealTopMovers"/>
  <input>
    <http:urlEncoded/>
  </input>
  <output>
    <mime:mimeXml part="Body"/>
  </output>
</operation>

<operation name="GetRealTopGainers">
  <http:operation location="/GetRealTopGainers"/>
  <input>
    <http:urlEncoded/>
  </input>
  <output>
    <mime:mimeXml part="Body"/>
  </output>
</operation>

<operation name="GetRealTopLosers">
  <http:operation location="/GetRealTopLosers"/>
  <input>
    <http:urlEncoded/>
  </input>
  <output>
    <mime:mimeXml part="Body"/>
  </output>
</operation>

<operation name="SearchByMarketCap">
  <http:operation location="/SearchByMarketCap"/>
  <input>
    <http:urlEncoded/>
  </input>
  <output>
    <mime:mimeXml part="Body"/>
  </output>
</operation>
<operation name="GetRealTop">
   <http:operation location="/GetRealTop"/>
   <input>
      <http:urlEncoded/>
   </input>
   <output>
      <mime:mimeXml part="Body"/>
   </output>
</operation>

<operation name="GetRemainingHits">
   <http:operation location="/GetRemainingHits"/>
   <input>
      <http:urlEncoded/>
   </input>
   <output/>
</operation>
</binding>

<binding name="RealTimeStockQuotesTestHttpPost" type="s0:RealTimeStockQuotesTestHttpPost">
   <http:binding verb="POST"/>
   <operation name="GetRealQuote">
      <http:operation location="/GetRealQuote"/>
      <input>
         <mime:content type="application/x-www-form-urlencoded"/>
      </input>
      <output>
         <mime:mimeXml part="Body"/>
      </output>
   </operation>
   <operation name="GetExtendedRealQuote">
      <http:operation location="/GetExtendedRealQuote"/>
      <input>
         <mime:content type="application/x-www-form-urlencoded"/>
      </input>
      <output>
         <mime:mimeXml part="Body"/>
      </output>
   </operation>
   <operation name="GetRealQuotes">
      <http:operation location="/GetRealQuotes"/>
      <input>
         <http:urlEncoded/>
      </input>
      <output/>
   </operation>
</binding>
</operation>
<operation name="GetBookDetails">
  <http:operation location="/GetBookDetails"/>
  <input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </input>
  <output>
    <mime:mimeXml part="Body"/>
  </output>
</operation>

<operation name="GetRealTopMovers">
  <http:operation location="/GetRealTopMovers"/>
  <input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </input>
  <output>
    <mime:mimeXml part="Body"/>
  </output>
</operation>

<operation name="GetRealTopGainers">
  <http:operation location="/GetRealTopGainers"/>
  <input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </input>
  <output>
    <mime:mimeXml part="Body"/>
  </output>
</operation>

<operation name="GetRealTopLosers">
  <http:operation location="/GetRealTopLosers"/>
  <input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </input>
  <output>
    <mime:mimeXml part="Body"/>
  </output>
</operation>

<operation name="SearchByMarketCap">
  <http:operation location="/SearchByMarketCap"/>
  <input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </input>
</operation>
<output>
  <mime:mimeXml part="Body"/>
</output>
</operation>
<operation name="GetRealTop">
  <http:operation location="/GetRealTop"/>
  <input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </input>
  <output>
    <mime:mimeXml part="Body"/>
  </output>
</operation>
<operation name="GetRemainingHits">
  <http:operation location="/GetRemainingHits"/>
  <input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </input>
  <output/>
</operation>
</binding>
</service>
<port name="RealTimeStockQuotesTestSoap" binding="s0:RealTimeStockQuotesTestSoap">
  <soap:address location="http://localhost:7001/controls/RealTimeStockQuotesTest.jws"/>
</port>
<port name="RealTimeStockQuotesTestHttpGet" binding="s0:RealTimeStockQuotesTestHttpGet">
  <http:address location="http://localhost:7001/controls/RealTimeStockQuotesTest.jws"/>
</port>
<port name="RealTimeStockQuotesTestHttpPost" binding="s0:RealTimeStockQuotesTestHttpPost">
  <http:address location="http://localhost:7001/controls/RealTimeStockQuotesTest.jws"/>
</port>
</service>