2008

Building web applications using web services

Keerthi Nannapaneni

BUILDING WEB APPLICATION USING WEB SERVICES

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

by

Keerthi Nannapaneni

December 2008

BUILDING WEB APPLICATION USING WEB SERVICES

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Keerthi Nannapaneni

December 2008

Approved by:

[redacted]

Dr. Ernesto Gomez, Chair,
Computer Science

11/26/2008
Date

[redacted]

Dr. Tong Lai Yu

[redacted]

Dr. Kerstin Voigt

ABSTRACT

This project was performed to get hands on experience on the implementation of web services. During this project execution, significant time was spent researching about existing web services and various programming environments that can be used for building the application. The following technologies were used to build this web application - AJAX, JavaScript, Java, Java Server Pages, HTML, and CSS. Existing data content from the following web service resources were used in this application- Google API, ArcWebServices, Weather XML data feed. The purpose of this application is to show the map location along with the weather of the place selected by a user. Using this application, the user will also have the ability to get driving directions and current traffic information. This application can be easily integrated with university web pages to give easy access to weather and place information for students and staff in California State University at San Bernardino (CSUSB). Through the course of this project information about how to find and evaluate existing web services, technologies and methods of integrating various web services into one application were gathered. All the important steps that were followed during the implementation were documented in the order of

execution. This documentation will help programmers find useful information when building a web services based application.

ACKNOWLEDGMENTS

This project was carried out with the guidance and assistance of my project committee. First, I would like to thank my advisor Ernesto Gomez, for his continuous support and for providing me with an opportunity to undertake a project in web services. I would also like to thank Dr. Tong Lai Yu for providing valuable suggestions during the course of this project and also for assisting me during my independent study. I also want to express my sincere gratitude to Dr. Kerstin Voigt for her guidance, understanding, and for valuable advice.

Besides my Project committee, I would like to thank the rest of computer science department faculty and staff, Dr. Josephine Mendoza, graduate advisor, for guiding me throughout my masters program. I thank my parents for supporting me throughout my life and encouraging me to pursue my interests. I also thank my husband Rama K Atluri for his great help and support in finishing this project.

TABLE OF CONTENTS

CHAPTER FOUR: MAINTENANCE

CHAPTER FIVE: CONCLUSIONS AND RECOMMENDATIONS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

The need for distributed computers to communicate with each other via a network led Web service to evolve. A Web service is an application that can communicate with other applications over a network by using a set of standardized protocols.

As part of my graduate requirement, I choose to work on a project concerning Web services. With the help of my graduate advisor Dr. Ernesto Gomez, identified the requirements during the project proposal phase. The goal of this project is to use technologies that have flexibility to be easily customized and configured to meet any new requirements.

This project uses services and data content from various third party Web services. During the initial phase of the project significant research was conducted to find suitable web sources. The final phase of the project time was spent to figure out the best way to integrate these to create a more meaningful application that can be used by the university community.

1.1 Purpose

By doing this project, an attempt was made to demonstrate the advantages of using existing web services to provide useful content by not investing significant resources to acquire and maintain data, software and hardware. The development environment and tools used in this project are available for free of cost from various organizations.

Various services have been chosen and integrated into one cohesive application to provide a customized and valuable solution to the university staff and students. Users simply enter the place of interest, and will have the ability to select from a candidate list of matches. After the place is selected the application will show the current and forecast weather conditions of that place. Though there are some individual services which show the city and weather separately there is no application which provides these services based on a common place name. In this project, these services are integrated making it much easier to a user to just type the common place name, for example: when a user inputs 'white house', the application will return the location as well as the weather information of that location and the ability to look up directions to this place and traffic information.

1.2 Scope

For the sake of timely completion of this project I have defined the scope of the project to sufficiently demonstrate the core concepts of web services by using the software engineering principles that I learned during the course of my masters program.

The scope of project is to create an easy to use web application for the university staff and students to access the following information.

- Current Weather Conditions and Forecast
- Place Name and its Map Location
- Driving Directions
- Traffic Information

This information is assumed to be used frequently by the university staff and students and is available in few different portals but not in one web application. Also, to complete the project in a timely manner it was agreed upon to include only the above mentioned services. These services will sufficiently demonstrate the use and implementation of web services and will serve the purpose of this project.

1.3 Significance

The need to exchange data between distributed applications written in various programming languages led web services to evolve. Web services exchange data between systems with the help of open protocols and standards like the internet. There are vast amounts of valuable information that is available both freely and as paid service throughout the internet. It is up to us to find the resources and make use of them to select appropriate content and serve them in a form that makes sense to us. For example, getting access to imagery and weather information from a satellite is not possible for a normal user. There are specialized companies that already do this work and publish the data as a web service to be accessed by subscribers. By accessing these web services, information can be accessed by making a request to the web service. The web service provider will define the format for sending a request and reading the response the service will generate. The client program makes a request for the web services across the network. The web service performs action, and sends the response back to the program. The use of web services in this project is to provide the functionality which can make a request to a program running on another server to simplify the project.

Through this project I have integrated various unique web services and content which were complimentary to each other. Through this integration I was able to create a rich solution that provides more information than an individual service would provide when used as is.

1.4 Limitations

As with any application, this project also has some technology, data and schedule limitations. The whole concept of web services at its core is to consume a service that is hosted by some organization out of our control. This means that the application will only work if the web services are functional and you are totally counting on the service to be up and running and the infrastructure to access 'Internet' is available to access the web service.

There are also many other limitations like the richness of the data content. For example, the weather web service will only provide weather information for about 7,900 places in the world, so you may not be able find the current weather conditions for some places of your interest. Unless the service provider adds weather information for those places there is nothing that this application can do to show the weather. Similarly, the data coverage and attribute completeness of street data in Google Maps for many cities of the developing and under developed countries is not sufficient enough to provide driving directions.

1.5 Definitions of Terms

This section has a list of terms and its definitions that are referred in the body of this document.

Asynchronous JavaScript and XML (AJAX), is a group of interrelated web development techniques used for creating interactive web applications or content rich Internet applications. With AJAX, web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. [24]

Extensible Markup Language (XML) is a standard, simple, self-describing way of encoding both text and data so that content can be processed with relatively little human intervention and exchanged across diverse hardware, operating systems, and applications. [18]

Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation of structured documents written in a markup language like XML and HTML. [26]

Hypertext Markup Language (HTML) is the predominant markup language for Web pages and provides a means to describe the structure of text-based information in a document. [27]

Java Server Pages (JSP) is the Java platform technology for delivering dynamic content to web clients in a portable, secure and well-defined way. [15]

Web Service Definition Language (WSDL) is an XML grammar that defines the functionality offered by a Web service and the format of messages sent and received by the Web service· [33]

World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web. [28]

Simple Object Access Protocol (SOAP) is an XML-based messaging protocol that defines a set of rules for structuring messages that can be used for simple one-way messaging. [17]

Application Programming Interface (API) is a set of functions, procedures, methods or classes that an operating system, library or service provides to support requests made by computer programs. [29]

Environmental Systems Research Institute (ESRI) is the world leader in geographic information system modeling and mapping software and technology. [6]

Really Simple Syndication (RSS) is a format for delivering regularly changing web content by retrieving the latest content from the sites. [16]

Java Development Kit (JDK) is bundle of software that you can use to develop Java based software. [11]

Java Runtime Kit (JRE) is an implementation of the Java Virtual Machine which actually executes Java programs. [11]

Java Archive (JAR) is a platform-independent file format that bundles classes, images, and other files into one compressed file, speeding download time. [9]

Object Oriented Programming (OOPs) is a programming paradigm that uses "objects" and their interactions to design applications and computer programs. [30]

Service Oriented Architecture (SOA) is a new architecture for the development of loosely coupled distributed applications. [14]

Web services Inspection Language, also known as WS-Inspection (WSIL) is an XML-based specification about how to locate Web services. [20]

Universal Description, Discovery, and Integration (UDDI) is both a client-side API and a SOAP-based server implementation that can be used to store and retrieve information on service providers and Web services. [20]

Hypertext Transport Protocol (HTTP) is a communications protocol used for the transfer of information on the Internet. [31]

CHAPTER TWO

REVIEW OF RELATED LITERATURE

Before identifying the topic for the project, significant time was spent conducting research reading various books and materials on the web. Major research included evaluation and identification of various technologies and tools that were open source and had good documentation.

The section below explains what web services are and how they work.

2.1 Web Services

Web services are self-contained and dynamic applications that require no additional software to integrate with other applications. They use XML messages for communication to send requests and get the responses which makes it system and platform independent. The basic platforms for Web services are HTTP and XML.

The architecture of web service consists of three modules service provider, service broker and service requester as illustrated in the Figure 1 below.

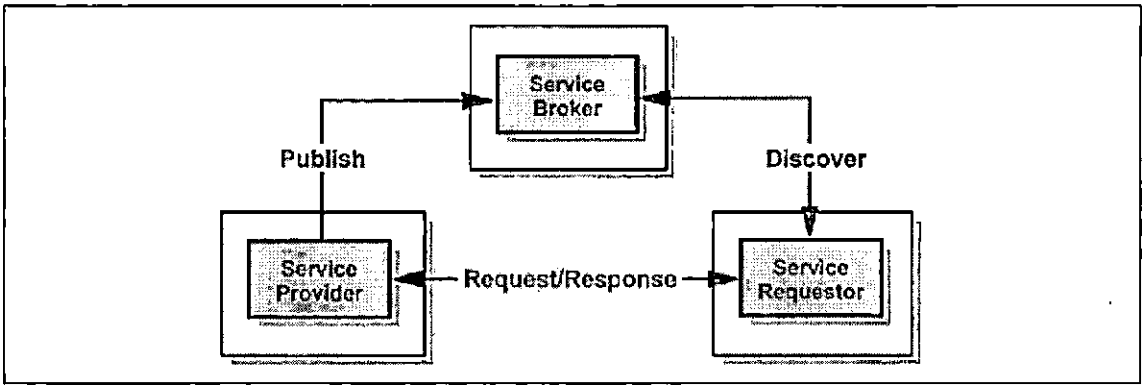


Figure 1. Web Service Components [20]

The service provider implements and hosts the service and publishes the access methods to service broker. Service broker acts as an interface and provides the access information to the service requestor. It acts as a centralized directory of all the services. The service requestor with the help of the service broker discovers the various access methods to the service and sends in the request to the service provider.

This architecture is based on Service Oriented Architecture which is a collection of services and the communication between them. The components used in the communication are Simple Object Access Protocol, Universal Description, Discovery and Integration and Web services Description Language.

The Web services Description Language is the basis for web services. It describes and specifies the location and operations the web service exposes. It uses XML to define the messages which enclose the data in tags. So that the consumer locate the data based on the tag information rather than the order. The service provider describes its service using WSDL and publishes to service broker who uses Universal Description, Discovery, and Integration. It acts a directory or registry to which all the services are published.

The service consumer looks up the service in the directory. The WDSL document is passed to the client who describes the methods and variables needed to access the service. All the messages are sent by SOAP between the various components in the architecture.

It is a platform and language independent communication protocol used for communication between applications via internet. The consumer builds a request based on the WDSL and sends it to service provider over HTTP and receives a response in the format mentioned in WDSL.

A detailed relationship between the various elements of web services is provided in the Figure 2 below.

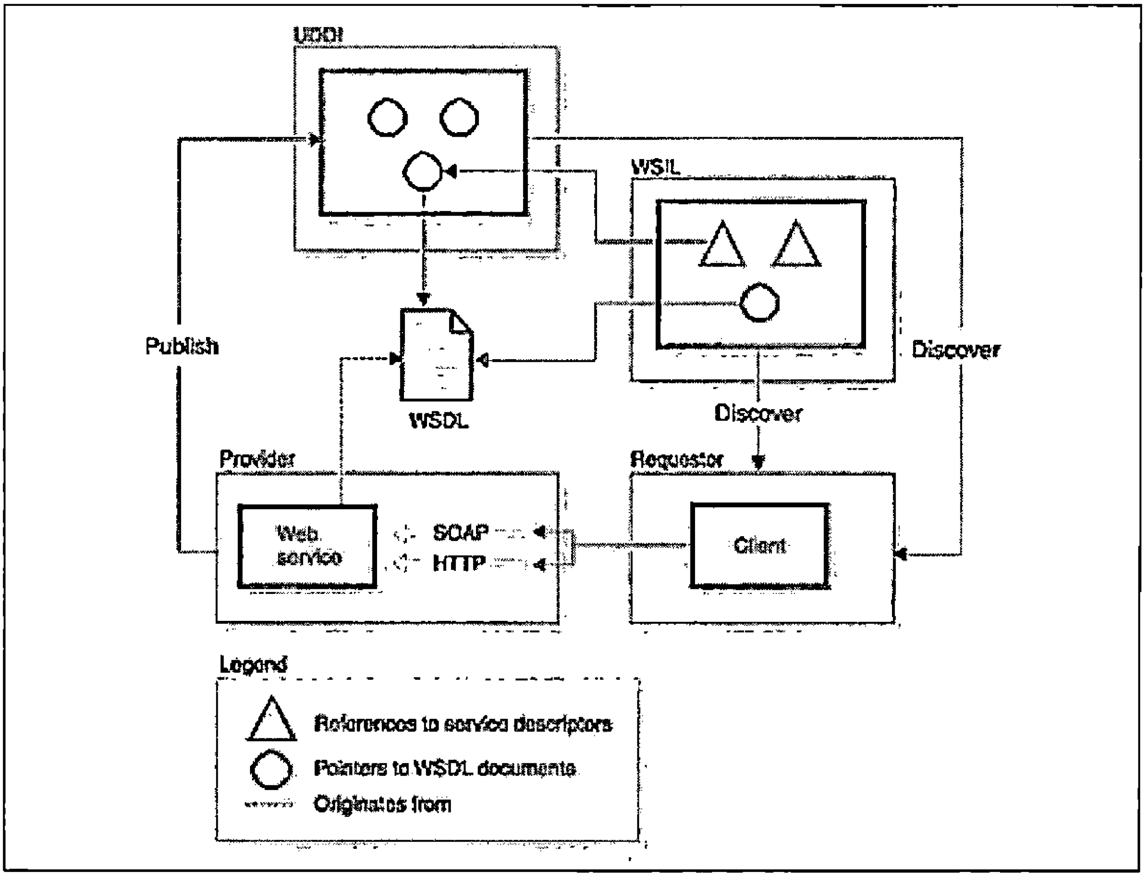


Figure 2. Service Oriented Architecture Approach [20]

2.2 Initial Research

The initial research started out by looking for appropriate web services delivering the information identified in section 1.2. The following sections provide more detailed overview of this process.

The flow diagram in Figure 3 below illustrates the high level steps during initial phase of the project.

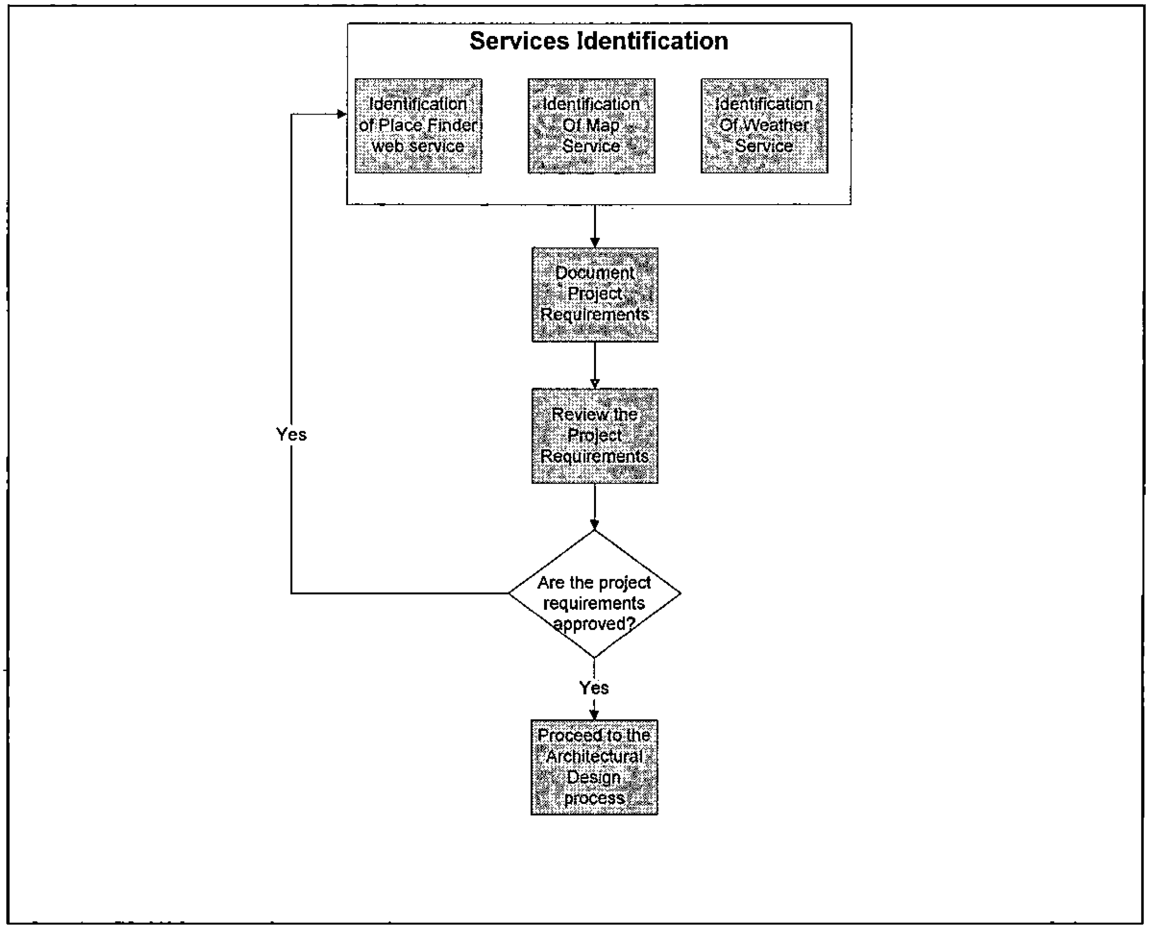

Services Identification
Identification of Place Finder web service
Identification Of Map Service
Identification Of Weather Service
Document Project Requirements
Review the Project Requirements
Yes
Are the project requirements approved?
Yes
Proceed to the Architectural Design process

Figure 3. Requirements and Design

2.2.1 Identify Existing Web Services

Since one of the primary purposes of this project is to use existing web services, extensive research was done on identifying suitable web services that meet the requirements.

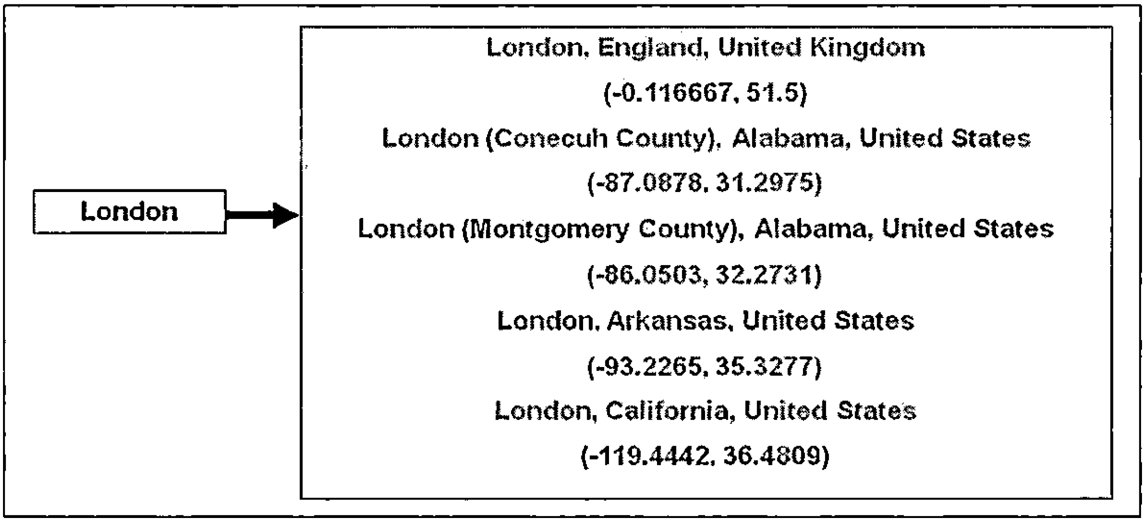
The following sub tasks were performed as part of this task.

Initially research started with the search for existing web services that provide the capability to locate millions of places in the world such as a city, a river, or an airport. Along with the places it should also determine the x, y coordinates of a place which can be used to locate the place on a map using a different service.

The following place finder services were evaluated for these purposes.

- Place Finder Web Service from Arc Web services provided by ESRI
- OpenStreetMap Name Finder by Free Wiki World Map

Out of the above mentioned two, Place Finder web service from ArcWeb services by Environmental Systems Research Institute (ESRI) has the best features and availability of the other one. There was also sufficient information on its website on how to use the service which made the implementation easy. Figure 4 below shows an example of the options it gives when a user types London.



London, England, United Kingdom
(-0.116667, 51.5)
London (Conecuh County), Alabama, United States
(-87.0878, 31.2975)
London (Montgomery County), Alabama, United States
(-86.0503, 32.2731)
London, Arkansas, United States
(-93.2265, 35.3277)
London, California, United States
(-119.4442, 36.4809)

Figure 4. ArcWeb Place Finder Search

The Arc web services provided from ESRI has a sample web service place finder which does not require any authentication steps and can be freely accessed once the license agreement is accepted. This service takes a place name as an argument and returns a list of locations for places that match this argument and also returns the location co-ordinates (x-coordinate, y-coordinate).

The next important step is the identification of existing map service that has coverage for the entire world which can be used to show the map location of a place selected by the user. The following map services are evaluated:

- Yahoo
- Google

- Map Quest
- Arc Web services

Of the above mentioned map services, Google map service was selected for the following reasons.

- Most current data up to 1 to 3 years old
- Better coverage for the entire world
- Better performance than others
- Good documentation and reliability of service
- Availability of other useful services like Traffic and Driving Directions along with the mapping service.
- Supports most of the browsers

Google maps are a web based mapping service created by Google to facilitate developers to integrate maps in their own web site. By requesting an API key and agreeing to its their terms of use, a map is embedded into the page. JavaScript functions are available for manipulating and adding content to the map which can be populated on the client side and passed in along with the request.

Once the place finder and map services are identified, the next task was to find a suitable weather service that can be easily integrated. The following weather services were evaluated for these purposes.

- Netweather from Accuweather
- Weatherbug from Weather
- Weather XML Data Feed from Weather
- XML RSS Feed from National Weather Service (NOAA)
- Global Weather Web Service from Webservicex

Out of all, weather XML data feed service from weather.com was the best source for this information for the following reasons.

- Better documentation
- Provides global weather based on Location ID
- Information available in XML format and free to use
- Current weather and forecast for the week data available

This weather service provides current weather conditions which are updated approximately every hour and include information such as temperature, humidity, wind, pressure, and cloud cover. It provides the current conditions for over 30,000 U.S. and over 7,900 international Location IDs which are updated every hour. It also provides four day forecast which is updated at least three times a day as per weather.com.

2.3 Identification of Programming Environment

After identifying the most suitable web services providing necessary content the next task was to select the programming environment. The flow chart in Figure 5 below identifies the important steps of this process.

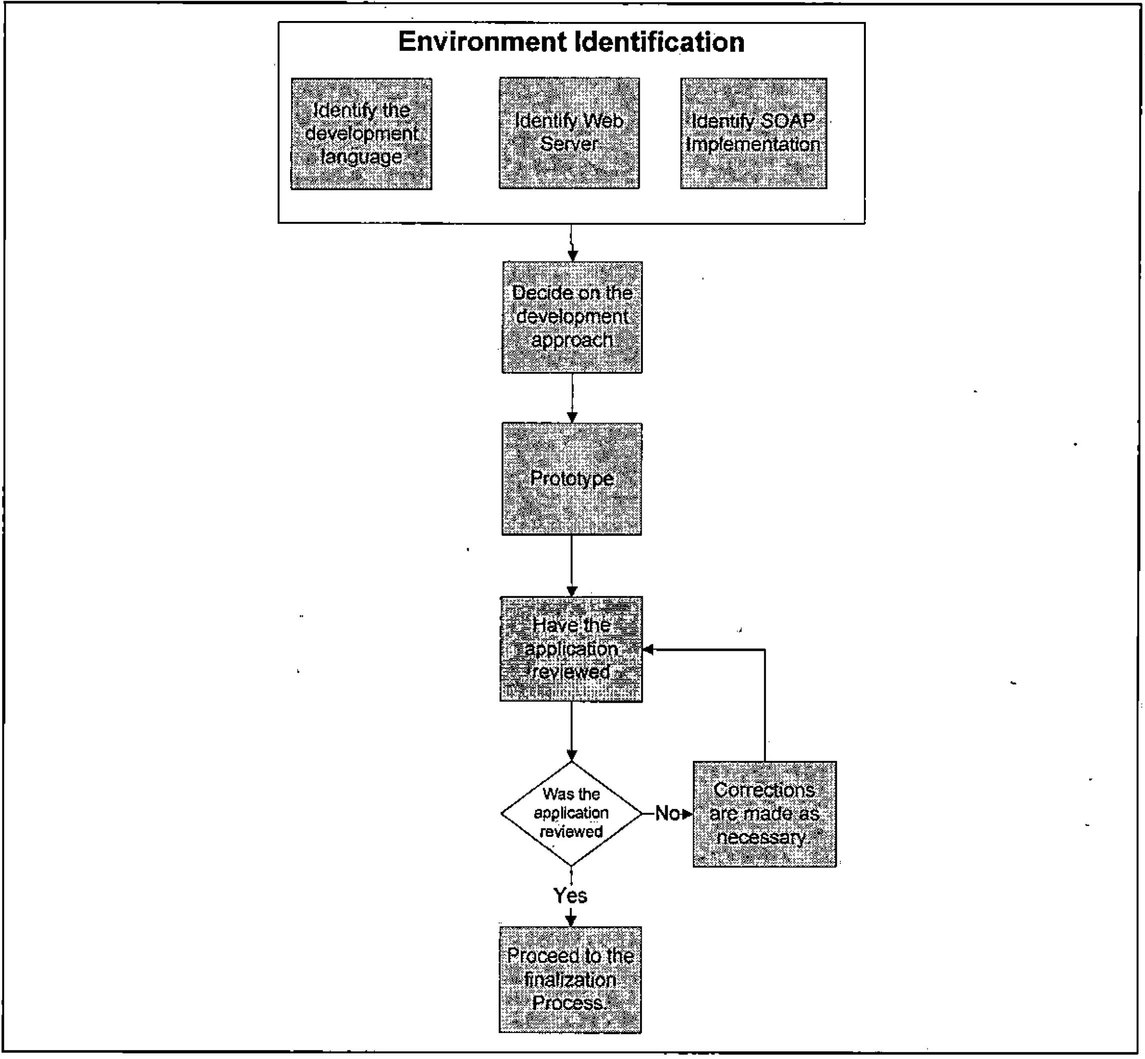

Environment Identification
Identify the development language
Identify Web Server
Identify SOAP Implementation
Decide on the development approach
Prototype
Have the application reviewed
Was the application reviewed
No
Corrections are made as necessary
Yes
Proceed to the finalization Process

Figure 5. Identify Development Environment

There are many programming languages like Java, .Net and C++ that we can chose from. Out of all Java was selected for the following reasons.

- Being familiar with Java programming I could start coding without spending time on research.
- It has extensive libraries like XML data parsing.
- Most of the web services are written in Java so integration with my application will be easy.

2.3.1 Java Language

Java programming language is a high-level language which can be characterized by the following characteristics. [12]

- Simple: Very easier to learn and understand for developer.
- Object Oriented: Treats everything as objects having properties and behavior.
- Distributed: Can access code on a remote machine by objects.
- Multi Threaded: Executes multiple processes independently and concurrently.

- Dynamic: Loads all the non compiled classes required for a class to compile by the compiler at runtime.
- Architecture neutral: Compiler generates object files based on bytes codes which can run on any machine.
- Portable: Can be developed for many cross platform systems.
- High performance: The java file can be compiled to byte's code on fly.
- Robust: Allocating and freeing the memory for the objects will be done runtime machine automatically.
- Secure: Secures all the files in the system.

To briefly explain how Java works, the source code is written in text with .Java extension. The compiler compiles the .Java into .class files which contains bytes codes. Bytes codes are machine language of Java Virtual Machine (JVM) which is machine independent. An instance of JVM runs the .class file which provides the required results. Figure 6 represents the steps explained above.

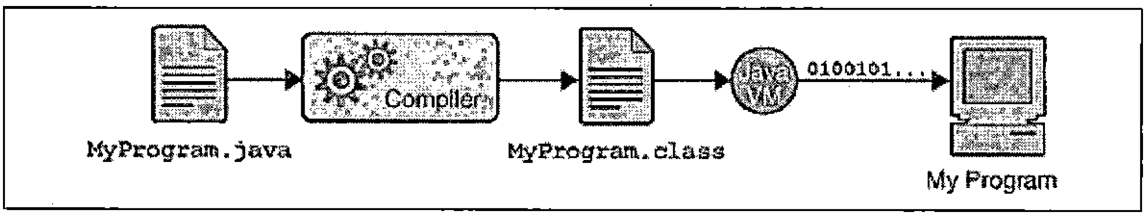


Figure 6. How Java Works [12]

The following are some of the advantages that Java offers.

1. Java Programs can be compiled once and run anywhere irrespective of the operating system used like Linux, Microsoft Windows etc.
2. There are many integration libraries available which can be used for Database operations, manipulation of remote objects and web application API.
3. Simpler, easier to read programs and more efficient reuse of code

2.3.2 Java Web Technologies

Since this is a Java based application, it is required to use a Java based web technology. Used Java Server Pages (JSP) because of the following reasons.

1. It is easier to write a JSP page since it contains HTML tags and Java code thus taking

care of presentation and business logic in one file.

2. Once loaded into the web server it takes less time to load the page on every request.

Once the server installs the JSP page the request is sent to the JSP container which translates the HTML and Java code into a Java file. Then the JVM compiles the file into .class file known as JSP implementation class. JSP's are built on servlet API, so all the methods and objects provided by servlet API can be used implicitly without any initialization. The request from the client is being processed by the page object and sends back the response to the client page. Figure 7 represents the steps explained above.

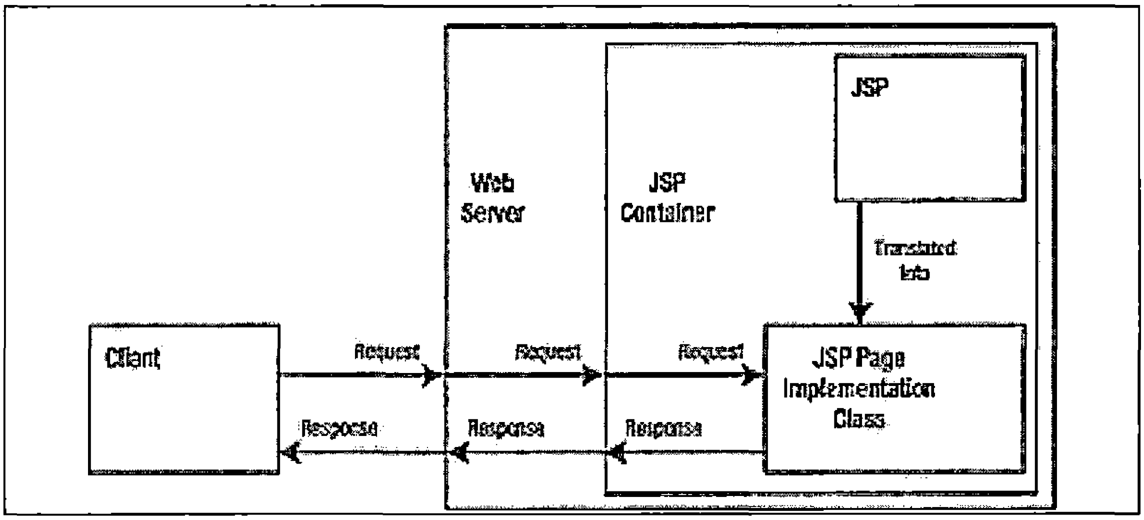


Figure 7. How does Java Server Pages Work [20]

2.3.3 Web Server

Since this application contains Java Server Pages which requires a web server to run the pages. After considering various web servers, Tomcat web server was used in this project. Apache Tomcat is a Servlet container that implements the Java Servlet and the Java Server Pages (JSP) and provides a "pure Java" HTTP web server environment for Java code to run.[13]

On the startup of the server it creates an instance of the server and port name the default being 8080 for tomcat. The service module groups up the connector with the engine component. The engine component processes the request from the web browser and maps it to the corresponding context or host. On parsing the request the context passes back the response to the web browser with the help of connectors. Figure 8 represents the steps explained above.

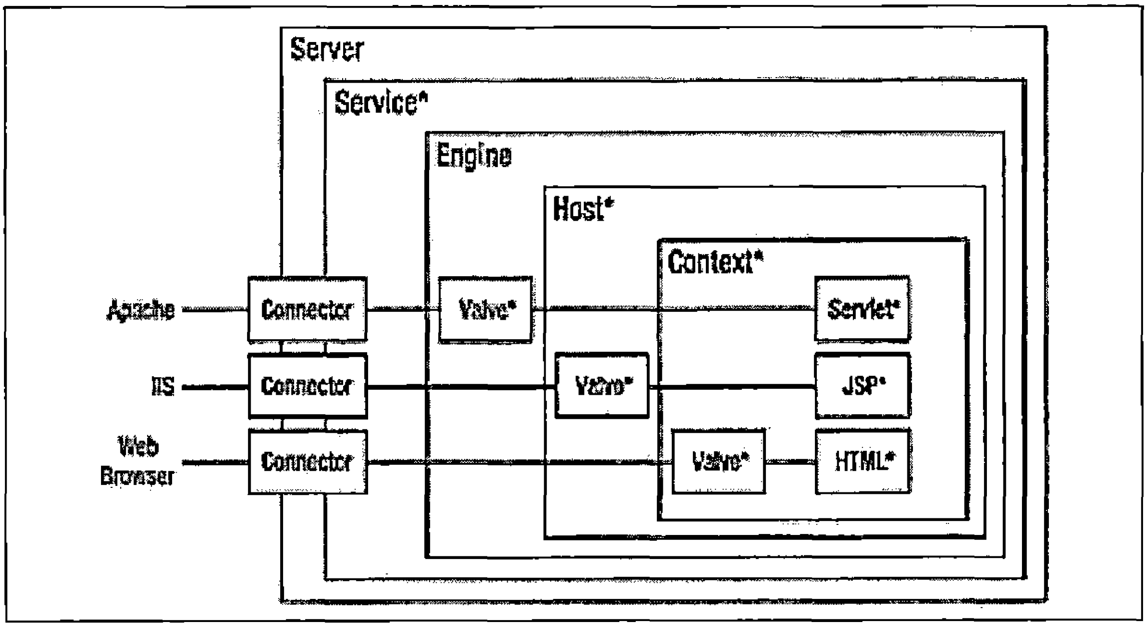


Figure 8. How does Tomcat Work [13]

2.3.4 Simple Object Access Protocol Implementation

To consume a web service we need a SOAP engine which will construct a client and a server and encode and decode data in SOAP format.

After performing analysis, decided to use Apache Axis because of the following reasons.

- Easily integrated with Tomcat web server
- Support for Web Service Description Language
- Integration with Java is easy

Apache Axis is an open source toolkit for creating and consuming Web services. [1]

It accepts the request from the service requester and sends the response from the service provider, thus shielding the developer from the logic to decode the messages between communications. It has two utilities Java2WSDL and WSDL2Java which are used to generate a WSDL file from java file and vice versa.

The Axis engine runs when the service consumer invokes an operation on the service provider, a SOAP message are built. It traverses through service handler chains which process the message to the format of specification. The global handler chains process the message for the Axis engine runtime. After being processed it is sent to the service provider through transport layer. After the SOAP message is decoded data in the message is converted to java objects which access the service methods and produce a result. The result or response is again passed back through the Axis engine to the service consumer. The Figure 9 below shows the architecture of Axis engine.

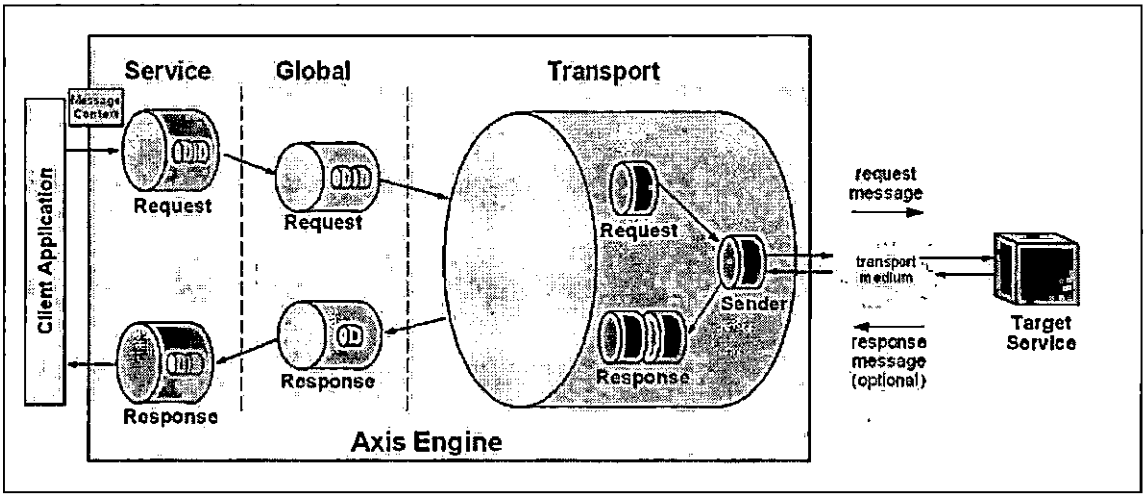


Figure 9. Axis Architecture [1]

2.3.5 Asynchronous JavaScript Library

Usually a response is sent back from the server when the user submits the page and the server processes the information and rebuilds the page. The user has to wait for the page to load until then thus wasting the time. Due to a pursuit for more interactive and faster web pages AJAX programming was born. It is used to create a more user friendly and interactive web application by retrieving the data from the server without actually refreshing or submitting the page. Thus, making the web application more user friendly and interactive.

With AJAX enabled web applications, all the requests to the server are handled by the AJAX engine which is written in JavaScript language. This engine instantiates the XMLHttpRequest object which allows sending, receiving and processing HTTP requests to and from the server. So when a request in the form of JavaScript variables is sent to the engine it makes a call to the server asynchronously and sends a response in XML format. The AJAX engine processes the server data and updates that section of the page without refreshing the entire page.

The Figure 10 below shows how AJAX's calls are sent to the server.

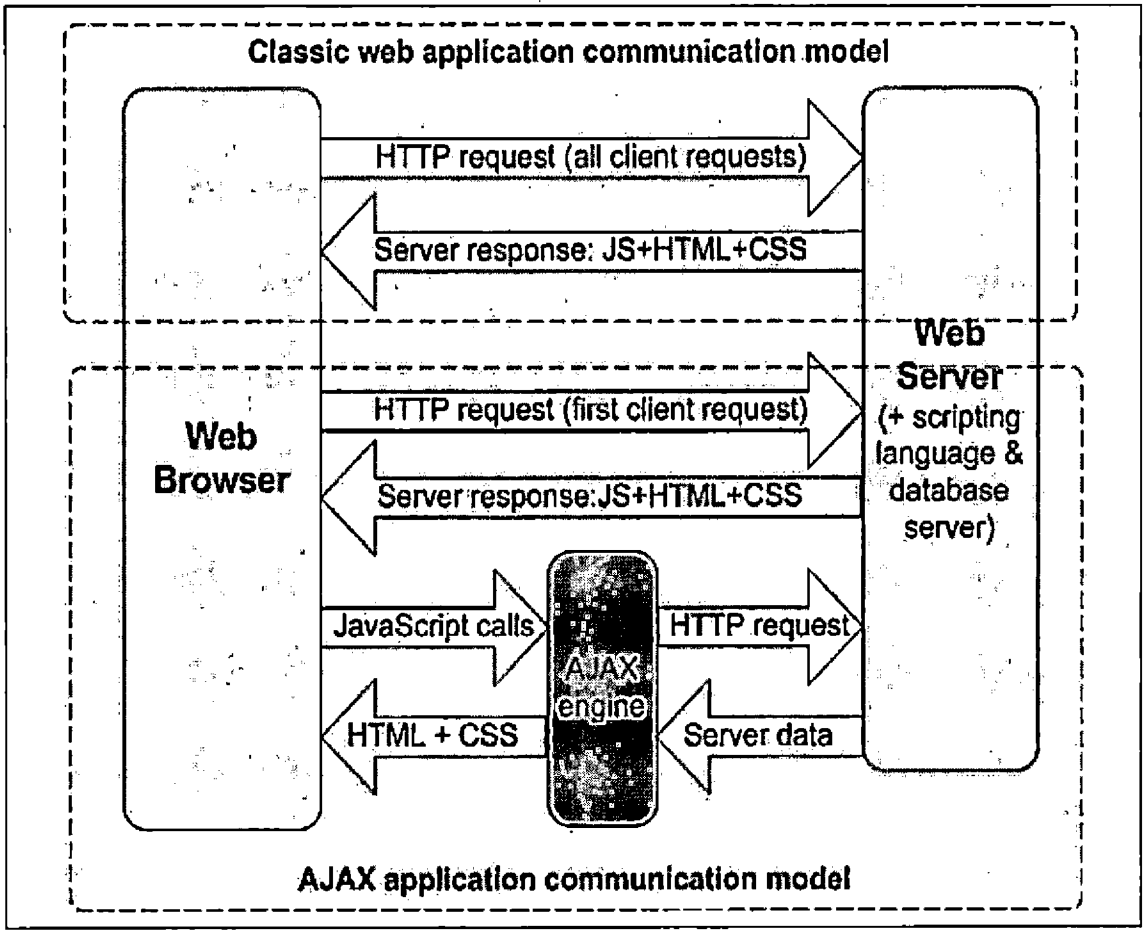

Classic web application communication model
HTTP request (all client requests)
Server response: JS+HTML+CSS
Web Browser
Web Server (+ scripting language & database server)
HTTP request (first client request)
Server response:JS+HTML+CSS
JavaScript calls
AJAX engine
HTTP request
HTML + CSS
Server data
AJAX application communication model

Figure 10. Classic and Asynchronous JavaScript Communication Models [8]

Use of AJAX libraries reduces creation of browser specific JavaScript code for sending the request and handling the response back.

After evaluating some libraries Scriptaculous library was determined right for this application because of the following reasons.

- Support handling results as XML

- Distinct separation between client and server code
- Easier implementation and documentation
- Integrated with several libraries

Scriptaculous library helps to develop visual, cross browser and easy to use JavaScript libraries which adhere to Web 2.0 programming. [32] The use of this library is to reuse code for common functions for AJAX.

After all the necessary services and software environment was finalized, I initially worked on the place finder web service to help define or refine requirements, to prove or demonstrate capability of intended functionality, and to learn more on the best implementation approach.

CHAPTER THREE

METHODOLOGY

The following sections outline the methodology of the implementation of the final application that meets the functional requirements as identified in section 2. The flexibility of the technology adopted as part of the proposed solution offers numerous architecture deployment and configuration options.

3.1 Setting Up the Development Environment

The programming platform has been discussed in the above sections. To develop with all the above technologies, an Integrated Development Platform Eclipse was used.

Eclipse software platform was used for building integrated web and application development in this project.

This section provides the details of the software and hardware environment used in this project.

3.1.1 Software Environment

The following software was installed in the order listed below.

Java 6.0 Installation:

1. Downloaded SDK from http://Java.sun.com/Javase/ downloads/index.jsp under C:\Program Files\Java.
2. It will download 2 files in the folder C:\Program Files\Java\jre1.6.0_03 and C:\Program Files\Java\jdk1.6.0_03

Tomcat 5.5 Installation:

1. Downloaded the binary distribution file named jakarata-tomcat-5.5.0.zip from Jakarta Apache home http://tomcat.apache.org/ download-55.cgi website.
3. Created a new folder Tomcat 5.5 under C:\keerthi_software\Tomcat 5.5 and unzip the downloaded file.

Apache Axis 1.4 Installation:

4. Downloaded the axis-bin-1_4.zip file from http://ws.apache.org/axis/.
5. Created a new folder axis-1_4 under C:\keerthi_software\Downloads\axis-1_4 and unzip the downloaded file.

Scriptaculous Framework 1.6 Installation:

6. Downloaded the scriptaculous-js-1.8.1.zip file from http://script.aculo.us/downloads.

7. Unzipped the file into C:\keerthi_software\Scriptaculous folder.

Eclipse 3.3 Installation:

8. Downloaded the .zip file from Eclipse home from http://download.eclipse.org/eclipse/downloads/drops/R-3.3.2-200802211800/

9. Created a new folder C:\keerthi_software\eclipse.

3.1.2 Hardware Environment

Performed the development on 32 bit Dell Windows Vista Business operating system laptop with 3 GB of memory. The model used is Vostro 1500 with Intel(R) Core(TM)2 Duo CPU 1.40GHz of processor speed.

3.2 Setting up the Project

This section provides the details of configuring the project in Eclipse and adding all the dependencies required.

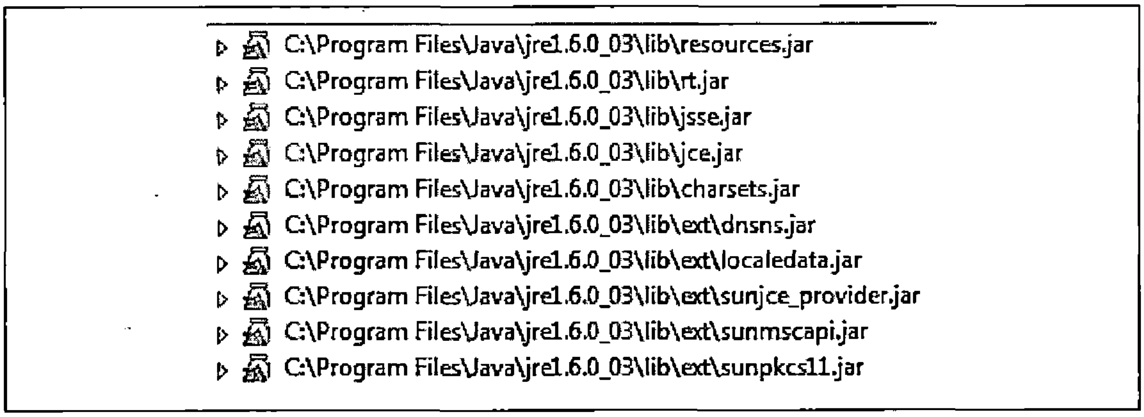
By clicking on eclipse.exe under C:\keerthi_software\eclipse, Eclipse is launched. Initially it shows a welcome screen with tutorials, help, overview and workbench options. Click on the workbench options and follow the configuration steps below.

3.2.1 Configure Java in Eclipse

This section details the steps used to configure Java in Eclipse.

1. Go to windows/preferences
2. Expand the Java option on the left window and click on Installed JREs.
3. Configure JRE by clicking on Add.
4. Attached a screen shot of the configuration of Java and click OK.
5. Go to Build Path under Java option and configure the JRE library and click Apply and OK.

The Figure 11 below shows all the Java Dependencies added in Eclipse.



```
C:\Program Files\Java\jre1.6.0_03\lib\resources.jar
C:\Program Files\Java\jre1.6.0_03\lib\rt.jar
C:\Program Files\Java\jre1.6.0_03\lib\jsse.jar
C:\Program Files\Java\jre1.6.0_03\lib\jce.jar
C:\Program Files\Java\jre1.6.0_03\lib\charsets.jar
C:\Program Files\Java\jre1.6.0_03\lib\ext\dnsns.jar
C:\Program Files\Java\jre1.6.0_03\lib\ext\localedata.jar
C:\Program Files\Java\jre1.6.0_03\lib\ext\sunjce_provider.jar
C:\Program Files\Java\jre1.6.0_03\lib\ext\sunmscapi.jar
C:\Program Files\Java\jre1.6.0_03\lib\ext\sunpkcs11.jar
```

Figure 11. Java Dependencies in Eclipse

Details of the above dependencies are explained in Table 1 below:

Table 1. List of Java Dependencies

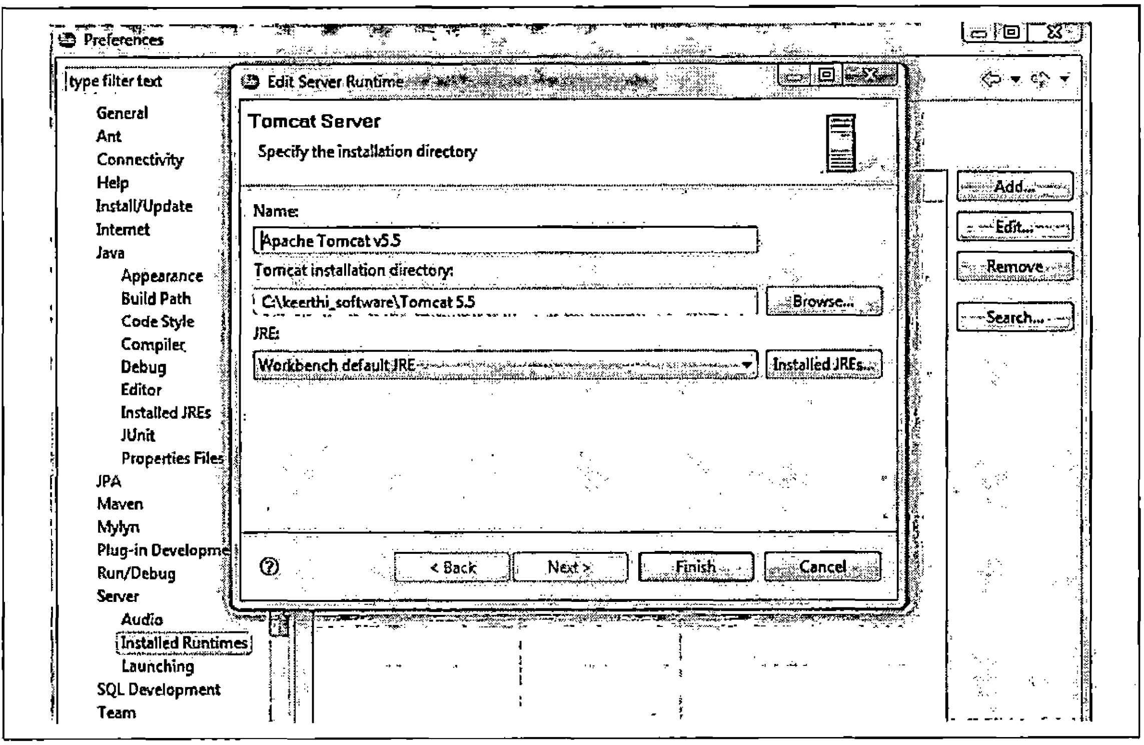
Rt.jar	Include all the Java packages used to run JDK
Resources.jar	Support the static web resources
Jsse.jar	Enable secure internet communications
Jce.jar	Framework for encryption, key generation and key agreement.
Charsets.jar	Mapping between Unicode characters and bytes
Dnsns.jar	Wrapper for JNDI dns provider
Localedata.jar	Used for locale specific messages
Sunjce_provider.jar	Used for cryptography
Sunmscapi.jar	Cryptography for windows based applications
Sunpkcs11.jar	Standards for encryption

3.2.2 Configure Apache Tomcat Web Server in Eclipse

This section details the steps used to configure Tomcat web server in Eclipse.

1. To configure tomcat server in Eclipse go to windows/preferences.
2. Open Server option and select Installed Runtimes.
3. Add Tomcat home to installation home and click Finish.

The Figure 12 below shows how to configure Tomcat in Eclipse.



Tomcat Server

Specify the installation directory

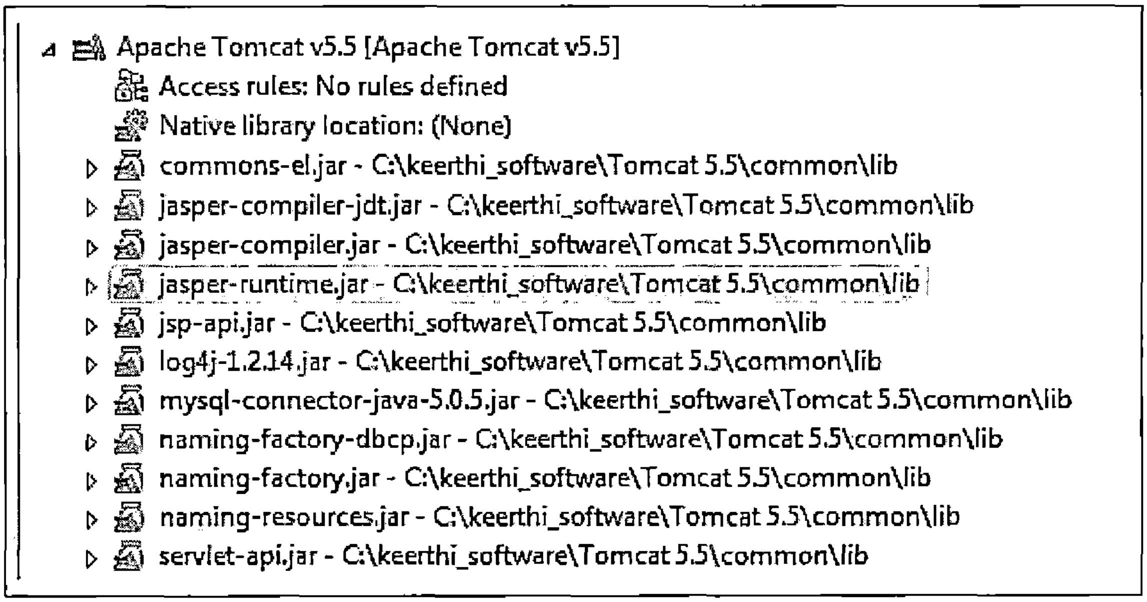
Name: Apache Tomcat v5.5

Tomcat installation directory: C:\keerthi_software\Tomcat 5.5

JRE: Workbench default JRE

Figure 12. Tomcat Configuration

The Figure 13 below shows all the Java Dependencies added in Tomcat.



```
Apache Tomcat v5.5 [Apache Tomcat v5.5]
    Access rules: No rules defined
    Native library location: (None)
  ▷ commons-el.jar - C:\keerthi_software\Tomcat 5.5\common\lib
  ▷ jasper-compiler-jdt.jar - C:\keerthi_software\Tomcat 5.5\common\lib
  ▷ jasper-compiler.jar - C:\keerthi_software\Tomcat 5.5\common\lib
  ▷ jasper-runtime.jar - C:\keerthi_software\Tomcat 5.5\common\lib
  ▷ jsp-api.jar - C:\keerthi_software\Tomcat 5.5\common\lib
  ▷ log4j-1.2.14.jar - C:\keerthi_software\Tomcat 5.5\common\lib
  ▷ mysql-connector-java-5.0.5.jar - C:\keerthi_software\Tomcat 5.5\common\lib
  ▷ naming-factory-dbcp.jar - C:\keerthi_software\Tomcat 5.5\common\lib
  ▷ naming-factory.jar - C:\keerthi_software\Tomcat 5.5\common\lib
  ▷ naming-resources.jar - C:\keerthi_software\Tomcat 5.5\common\lib
  ▷ servlet-api.jar - C:\keerthi_software\Tomcat 5.5\common\lib
```

Figure 13. Tomcat Dependencies

Details of the above dependencies are explained in the Table 2 below:

Table 2. List of Tomcat Dependencies

Commons-el.jar	Provides an interpreter for expression language
Jasper-compiler-jdt.jar	Support eclipse to compile JSP Java source code
Jasper-compiler.jar	Compiles JSP class
Jasper-runtime.jar	Helps to run the JSP class file
Jsp-api.jar	Classes for JSP API
Log4j-1.2.14.jar	Enable logging at runtime for debugging
Mysql-connector-Java-5.0.5.jar	Helps to build database applications
Naming-factory-dbcp.jar	
Naming-factory.jar	Contains factories used by naming resources
Naming-resources.jar	Manages the naming resources associated with JNDI context
Servlet-api.jar	Contracts between servlet class and runtime environment

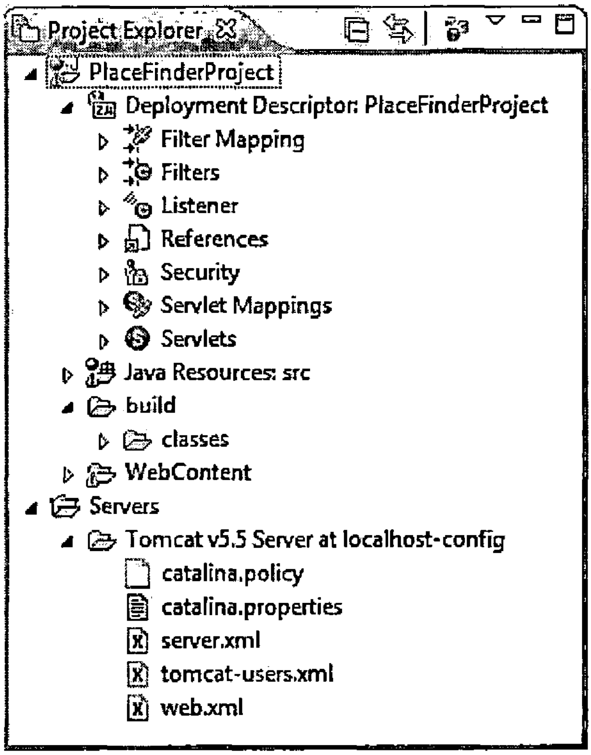
3.2.3 Create PlaceFinderProject in Eclipse

This section details the high level steps followed for creating the Place Finder application.

- Create a new Java project PlaceFinderProject under File/New/Other.
- Now a project is created with source and web Content folders.

- Under Open File/New/Other Server, select Tomcat v5.5 Server and click finish.
- Under Windows/Show View select Servers. A window is opened with server tomcat configured in it with start, stop and debugging options.
- Right click on the tomcat server and open Add and Remove Project. Add the PlaceFinderProject jar file to tomcat. Now the project is configured with the server.
- Select Window/Open Perspective/other and select Java EE Option. The following views are opened with Project Explorer, Server, Console and Progress options.
- Right Click on the project PlaceFinderProject and properties. The structure of the PlaceFinderProject will have the following files.

The Java Resources: src folder should have all the Java class files. The Web Content folder will have all the JSP files, HTML files, image files and configuration files like web.xml. The Server folder will have all the server configuration files like server.xml where the port address is mapped. The Figure 14 below show the place finder project folder.



```
Project Explorer
PlaceFinderProject
    Deployment Descriptor: PlaceFinderProject
        Filter Mapping
        Filters
        Listener
        References
        Security
        Servlet Mappings
        Servlets
    Java Resources: src
    build
        classes
    WebContent
Servers
    Tomcat v5.5 Server at localhost-config
        catalina.policy
        catalina.properties
        server.xml
        tomcat-users.xml
        web.xml
```

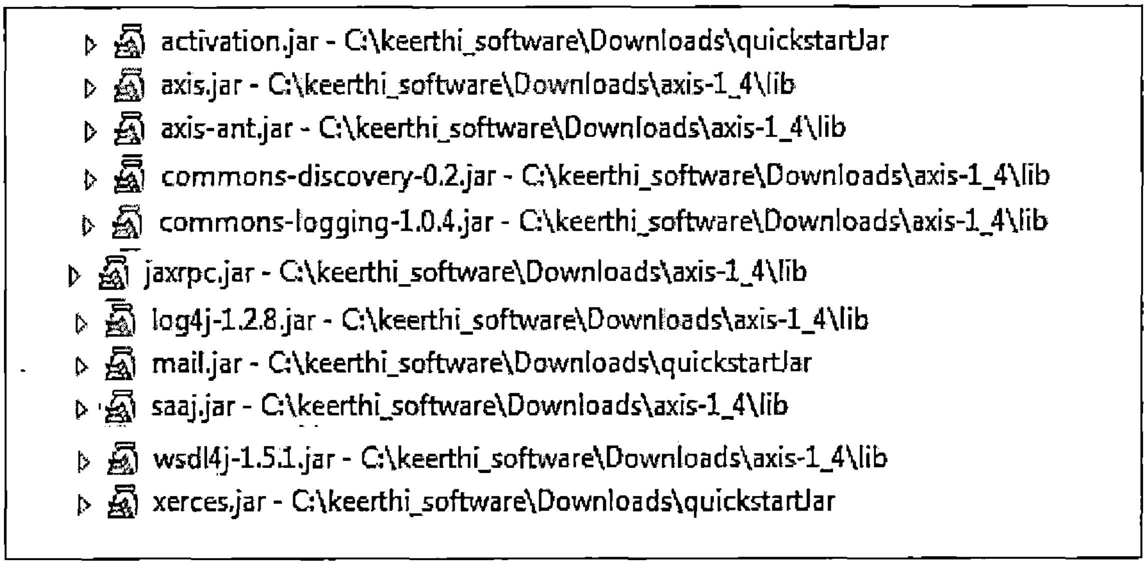
Figure 14. Place Finder Project Folder

3.2.4 Configure Apache Axis to Project

To configure Apache Axis, added the following Jar files to the Java build path of the project. Click on Add External JARs and browse to C:\keerthi_software\Downloads\axis-1_4\lib and select all the jar files.

Also added activation.jar, mail.jar, xerces.jar from downloading from internet.

Added all the jars listed in the Figure 15 below to the project.



```
activation.jar - C:\keerthi_software\Downloads\quickstartJar
axis.jar - C:\keerthi_software\Downloads\axis-1_4\lib
axis-ant.jar - C:\keerthi_software\Downloads\axis-1_4\lib
commons-discovery-0.2.jar - C:\keerthi_software\Downloads\axis-1_4\lib
commons-logging-1.0.4.jar - C:\keerthi_software\Downloads\axis-1_4\lib
jaxrpc.jar - C:\keerthi_software\Downloads\axis-1_4\lib
log4j-1.2.8.jar - C:\keerthi_software\Downloads\axis-1_4\lib
mail.jar - C:\keerthi_software\Downloads\quickstartJar
saaj.jar - C:\keerthi_software\Downloads\axis-1_4\lib
wsdl4j-1.5.1.jar - C:\keerthi_software\Downloads\axis-1_4\lib
xerces.jar - C:\keerthi_software\Downloads\quickstartJar
```

Figure 15. Apache Axis Jars

The basic functionality of these jars is shown in Table 3 below.

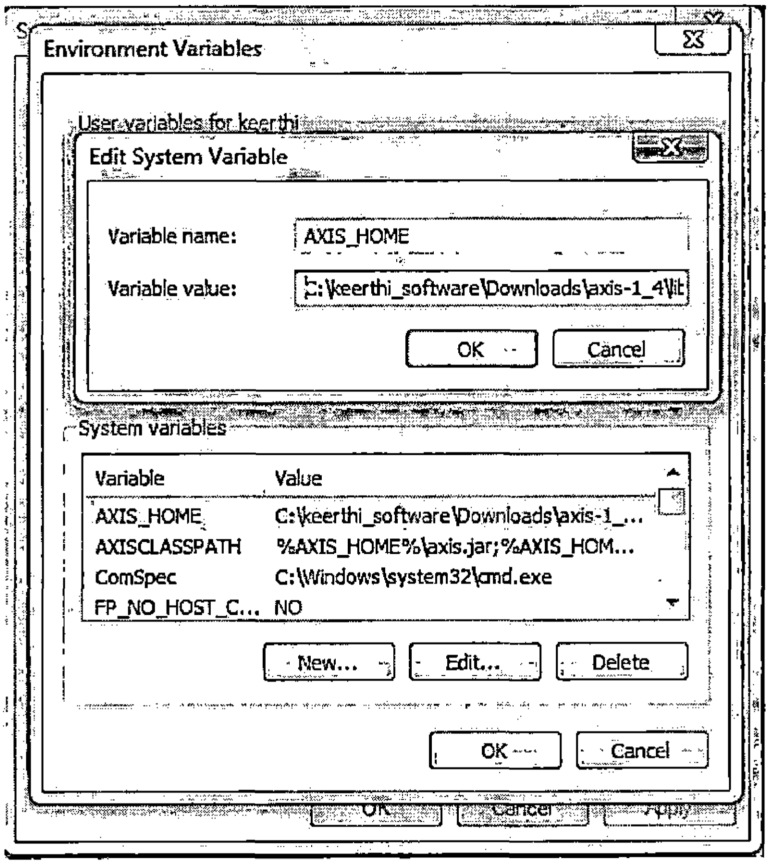
Table 3. List of Apache Axis Dependencies

Activation.jar	Instantiate the appropriate bean for an arbitrary piece of data in Java
Axis.jar	Compiled Axis
Axis-ant.jar	To automate building processes inside ant
Commons-discovery-0.2.jar	Locate resources by mapping service/reference names
Commons-logging-1.0.4.jar	Wrapper around logging API
Jaxrpc.jar	API for XML based RPC
Log4j-1.2.8.jar	Enable logging at runtime for debugging
Mail.jar	Build mail and messaging functionality
Saaj.jar	Creates and sends SOAP messages
Wsdl4j-1.5.1.jar	Creates, represents and manipulates WSDL documents
Xerces.jar	Java parser

Once the jars are added to the project build path we need to add them to system path in windows.

To set environment variables, select Start/Settings/Control Panel, and choose the System option. Select the Advanced system settings and click the Environment Variables button.

Add %AXIS_HOME% to system variables so that every user has access to it, click the New button then enter AXIS_HOME as the variable name, and enter the directory where Apache Axis was installed as the value.



Variable name: AXIS_HOME

Variable value: C:\keerthi_software\Downloads\axis-1_4\lit

Variable	Value
AXIS_HOME	C:\keerthi_software\Downloads\axis-1_...
AXISCLASSPATH	%AXIS_HOME%\axis.jar;%AXIS_HOM...
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO

Figure 16. Setting Environmental Variable for Apache Axis

Also add AXISCLASSPATH as variable and add all the jar files to the class path.

Variable Name: AXISCLASSPATH

```
Variable Value:
%AXIS_HOME%\axis.jar;%AXIS_HOME%\jaxrpc.jar;
%AXIS_HOME%\saaj.jar;
%AXIS_HOME%\commons-logging-1.0.4.jar;
%AXIS_HOME%\commons-discovery-0.2.jar;
%AXIS_HOME%\wsdl4j-1.5.1.jar;
C:\keerthi_software\Downloads\quickstartJar\activation.jar;
C:\keerthi_software\Downloads\quickstartJar\mail.jar;
C:\keerthi_software\Downloads\quickstartJar\xerces.jar;
```

3.3 Implementation of the Project

This section explains the high level steps that were executed during the implementation of the each individual component.

3.3.1 Place Finder

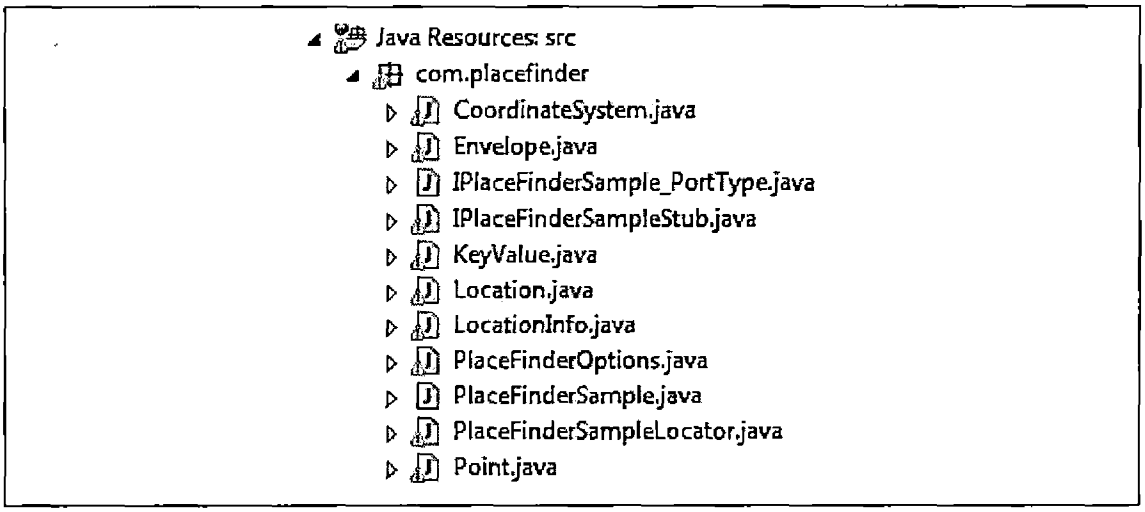
To consume the place finder service of ESRI, we need a WSDL document which describes the web services and the methods to access it. Below is a link to the document location.

http://arcweb.esri.com/services/v2/PlaceFinderSample.WSDL.

To access all the methods of the web service we need to run WSDL2Java tool against the WSDL location in the command prompt. Include AXISCLASSPATH which contains all the jars to solve all the dependencies.

```
C:> Java -cp %AXISCLASSPATH% org.apache.axis.wsdl.WSDL2Java
http://arcweb.esri.com/services/v2/PlaceFinderSample.WSDL -p
esri.arcwebservices
-v
```

It creates a new package named esri/arcwebservces/v2 which contains all the methods required to access this service. Copy these files into eclipse project in a new package com.placefinder. Given below in Figure 17 is the list of all the classes generated by WSDL location.



```
◢ Java Resources: src
  ◢ com.placefinder
    ▷ CoordinateSystem.java
    ▷ Envelope.java
    ▷ IPlaceFinderSample_PortType.java
    ▷ IPlaceFinderSampleStub.java
    ▷ KeyValue.java
    ▷ Location.java
    ▷ LocationInfo.java
    ▷ PlaceFinderOptions.java
    ▷ PlaceFinderSample.java
    ▷ PlaceFinderSampleLocator.java
    ▷ Point.java
```

Figure 17 Place Finder Classes

By understanding the WSDL file the methods and parameters needed to access these classes to get the data is understood.

Below is the Figure 18 showing UML class diagram of all the classes, methods and relationship between them.

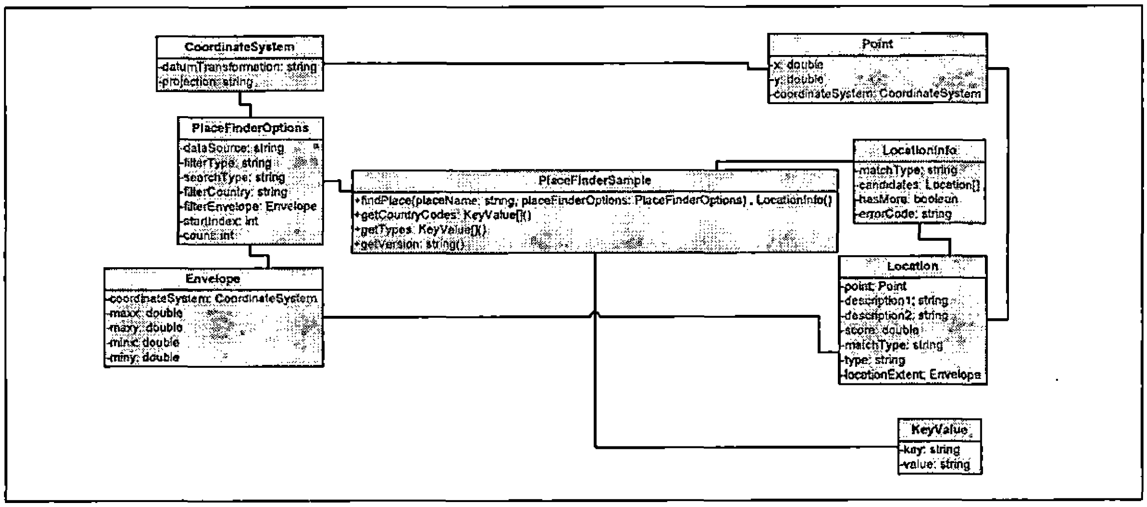
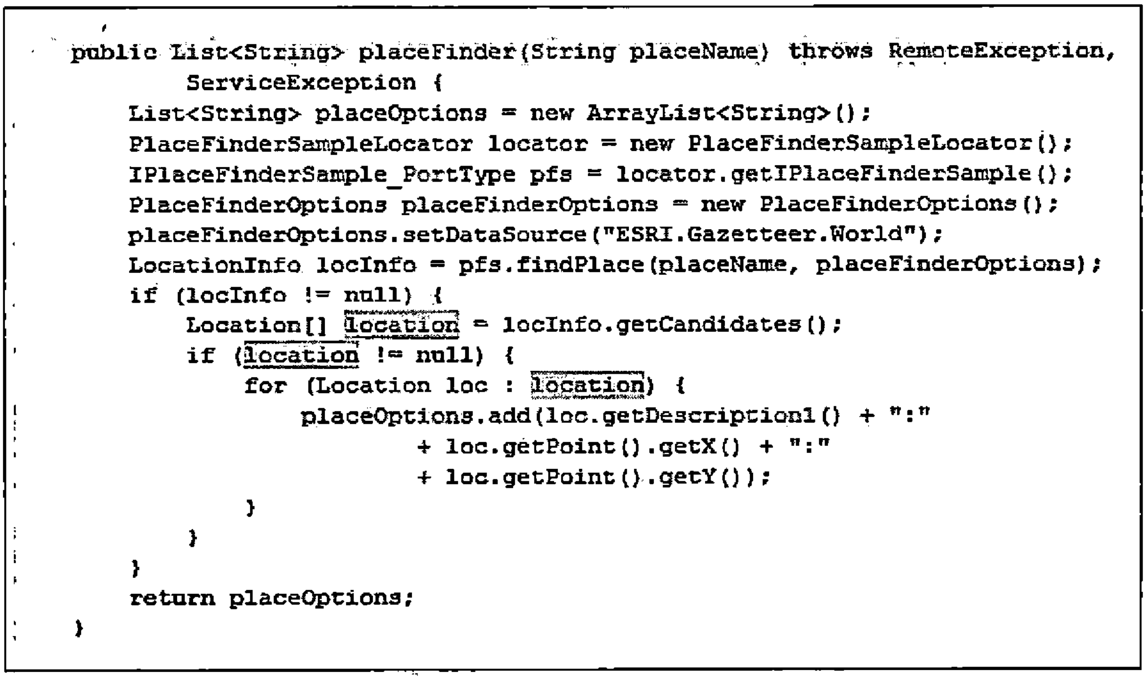


Figure 18. Place Finder Class Diagram

Created an ImplementPlace.Java class in com.project package which takes in the place name and return list of place names and coordinates. The Figure 19 below shows a code snippet on how to access the methods of the place finder web service and retrieve the list of matching place names and coordinates.



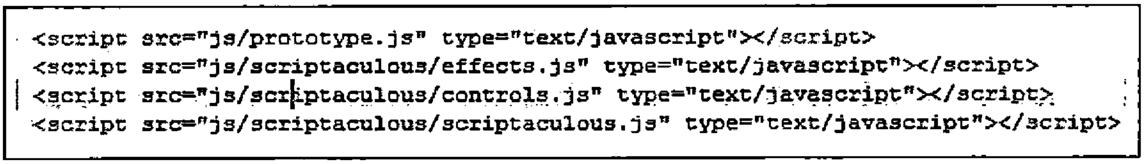
```
public List<String> placeFinder(String placeName) throws RemoteException,
        ServiceException {
    List<String> placeOptions = new ArrayList<String>();
    PlaceFinderSampleLocator locator = new PlaceFinderSampleLocator();
    IPlaceFinderSample_PortType pfs = locator.getIPlaceFinderSample();
    PlaceFinderOptions placeFinderOptions = new PlaceFinderOptions();
    placeFinderOptions.setDataSource("ESRI.Gazetteer.World");
    LocationInfo locInfo = pfs.findPlace(placeName, placeFinderOptions);
    if (locInfo != null) {
        Location[] location = locInfo.getCandidates();
        if (location != null) {
            for (Location loc : location) {
                placeOptions.add(loc.getDescription1() + ":"
                        + loc.getPoint().getX() + ":"
                        + loc.getPoint().getY());
            }
        }
    }
    return placeOptions;
}
```

Figure 19. Code Snippet of the Place Finder

Created a testFinder.jsp in WebContent folder page added 3 input fields for place name, x-coordinate and y-coordinate.

Created js folder in WebContent of the project and add scriptaculous library.

The Figure 20 below shows the Scriptaculous libraries are added to the jsp page.



```
<script src="js/prototype.js" type="text/javascript"></script>
<script src="js/scriptaculous/effects.js" type="text/javascript"></script>
<script src="js/scriptaculous/controls.js" type="text/javascript"></script>
<script src="js/scriptaculous/scriptaculous.js" type="text/javascript"></script>
```

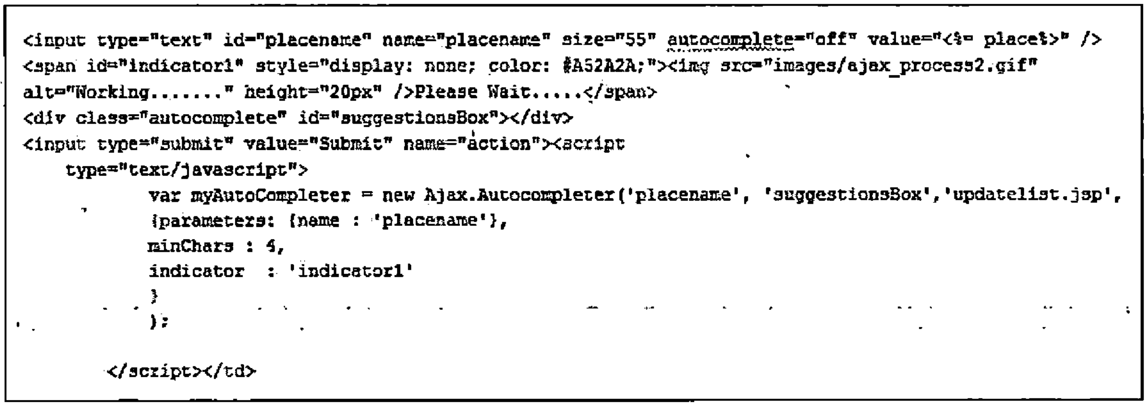
Figure 20. Scriptaculous Libraries

Instead of the user typing in the full place name and submitting the form for the list I implemented Ajax.AutoCompleter method for the place name for auto completing the place name.

The Autocompleter method needs some parameters to be passed in at the time of request.

- Name of the parameter typed on the autocompletion field.
- Update element id used to populate the list of the results.
- Url of the page which converts the XML data to HTML tags.
- Value of the parameter to be passed in the jsp page.
- Minimum number of characters that must be entered in the input field before an AJAX request can be made.
- While the request is processed by the server this element can be shown.

Figure 21 below shows the implementation of the Ajax.Autocompleter with all the parameters defined.

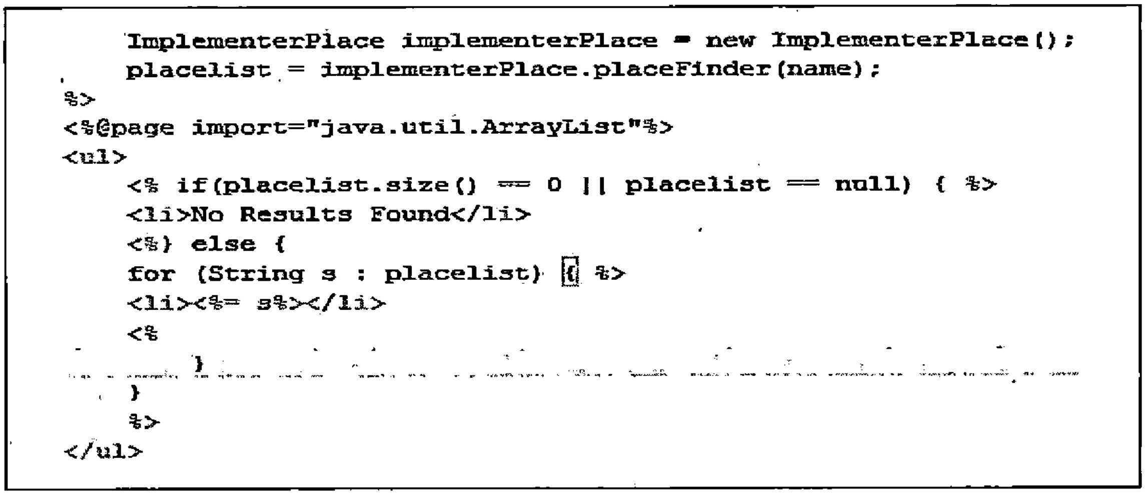


```
<input type="text" id="placename" name="placename" size="55" autocomplete="off" value="<%= place%>" />
<span id="indicator1" style="display: none; color: #A52A2A;"><img src="images/ajax_process2.gif"
alt="Working......." height="20px" />Please Wait.....</span>
<div class="autocomplete" id="suggestionsBox"></div>
<input type="submit" value="Submit" name="action"><script
    type="text/javascript">
            var myAutoCompleter = new Ajax.Autocompleter('placename', 'suggestionsBox','updatelist.jsp',
            {parameters: {name : 'placename'},
            minChars : 4,
            indicator  : 'indicator1'
            }
            );

        </script></td>
```

Figure 21. Asynchronous JavaScript Auto Completer

The results of AJAX are in XML format so in order to display to the user we need to convert it into HTML data. Updatelist.jsp removes the XML tags and displays the results in the form of a list. Below in Figure 22 is the code snippet of how the results from the web service are intercepted to a list.



```
        ImplementerPlace implementerPlace = new ImplementerPlace();
        placelist = implementerPlace.placeFinder(name);
    %>
    <%@page import="java.util.ArrayList"%>
    <ul>
        <% if(placelist.size() == 0 || placelist == null) { %>
        <li>No Results Found</li>
        <%} else {
        for (String s : placelist) { %>
        <li><%= s%></li>
        <%
          }
      }
        %>
    </ul>
```

Figure 22. Place Names Interception

To add color and display to the results included a css file to the jsp page.

3.3.2 Google Maps

With the place finder web service locating a place along with coordinates, we need to implement Google maps which will show the map location of the place in a nice map.

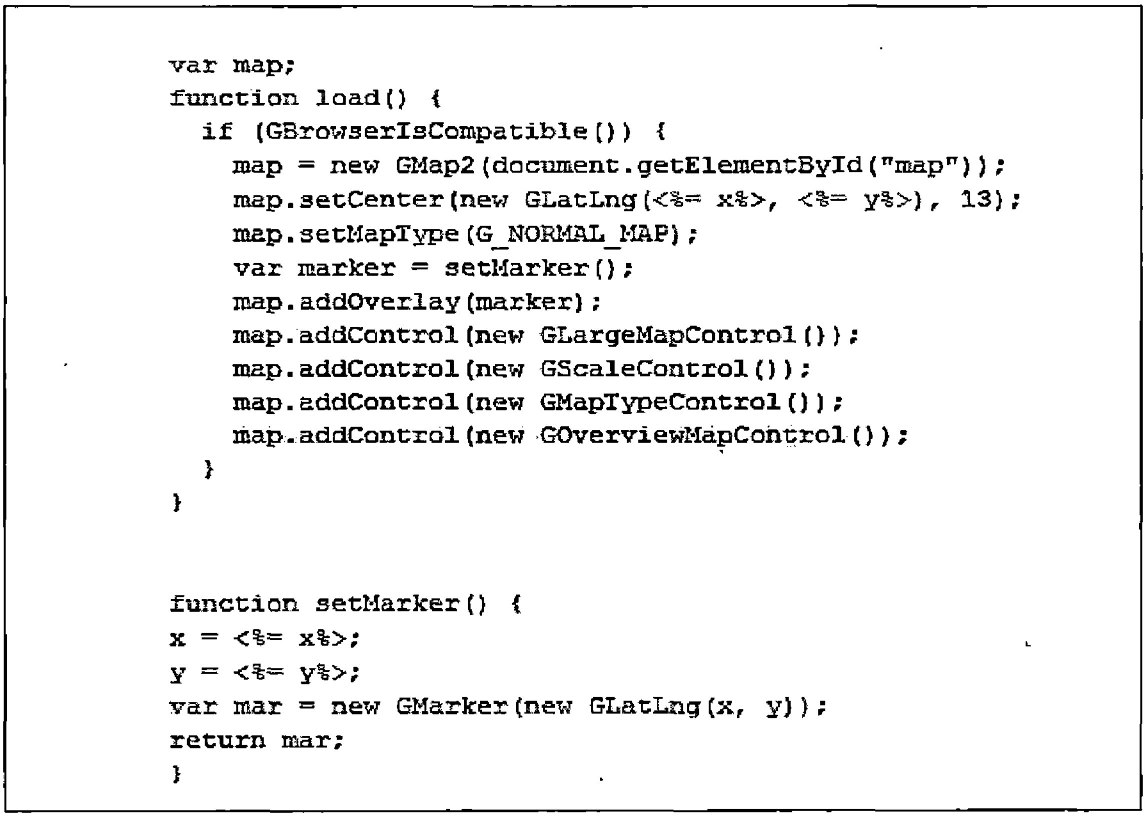
For accessing Google maps API go to http://code.google.com/apis/maps/. Sign up for Google Maps API by agreeing to terms and conditions to their site and providing the URL of the application.

Login to the website using your Google account login and password. Once logged in we get access to the key and the JavaScript methods used to get the map. Add the map to testFinder.jsp with a <script> tag as shown in Figure 23.

```
<script
     src="http://maps.google.com/maps?file=api&amp;v=2&amp
;key=ABQIAAAAUviqLvpFV2xOFpymSvH1BBSEwHm_qL4TlW-2-
tntSIcYoqMV7xSJGJihbuIniJnyG_ZTV_Psn7xRBg"
     type="text/Javascript"></script>
```

Figure 23: Adding Google Maps to Page

The parameters passed in to the maps.google are the version of the API used and the unique key to access the map. The basic function to get the map and manipulating the coordinates is shown below in Figure 24.



```
var map;
function load() {
  if (GBrowserIsCompatible()) {
    map = new GMap2(document.getElementById("map"));
    map.setCenter(new GLatLng(<%= x%>, <%= y%>), 13);
    map.setMapType(G_NORMAL_MAP);
    var marker = setMarker();
    map.addOverlay(marker);
    map.addControl(new GLargeMapControl());
    map.addControl(new GScaleControl());
    map.addControl(new GMapTypeControl());
    map.addControl(new GOverviewMapControl());
  }
}


function setMarker() {
x = <%= x%>;
y = <%= y%>;
var mar = new GMarker(new GLatLng(x, y));
return mar;
}
```

Figure 24. Accessing Google Map Functions

The value of x and y in the code are x-coordinate and y-coordinate from the place finder web service which is mapped as longitude and latitude.

Create a map object in the map. Create an OnLoad() event on the body of the page to load the map. In JavaScript define the load function. Initialize the map by creating GMap2 object passing in the div id of the map element. A new instance of the map is created. The map object can be initialized by invoking setCenter() method. The method takes in GLatLng object which passes in the longitude, latitudes and zoom level. The mapType() method takes a map type which are defined below.

- G_NORMAL_MAP- the default view
- G_SATELLITE_MAP - showing Google Earth satellite images
- G_HYBRID_MAP - showing a mixture of normal and satellite views
- G_DEFAULT_MAP_TYPES - an array of these three types, useful for iterative processing

Used NORMAL_MAP parameter in this application as the default view. A new marker object is created to mark the result on the map based on the longitude and latitude. The marker icon can be customized for each map. Also, used the DEFAULT_ICON to display the marker provided by the Google API. The marker is added by overlay() method. Now more control is added to the map that helps in user navigation and changing the map view.

- GLargeMapControl() is added to the map to facilitate the zoom level to the map.

- GScaleControl() is added to the map to show the scale control.
- GMapTypeControl() is added to the map to give an option to the user to select the map type.
- GOverviewMapControl() is added to the map to give a small overview of the map.

3.3.3 Accessing Weather XML Data Feed

With the map and place finder implemented, the weather service is accessed using Weather XML Data Feed by clicking http://www.weather.com/services/xmloap.html.

Registering up for the service can be done by creating a user profile with providing name, password and email address. XML partner ID and License key are sent in email on registration access the data from weather.com we need sent in the email.

PartnerID = "1072042464"

LicenceKey = "ea6c74f0fa4072bf"

Also included in the email is a zip file which consists of terms and conditions surrounding the use of the Service, graphical icons that are necessary to represent the sky conditions associated with our current conditions and forecast data, and weather logo.

In the XML data feed following data in Table 4 can be intercepted for Current Conditions.

Table 4. Current Condition Data from Weather Extensible Markup Language Data Feed

Location Id	Wind Gust
City name (mixed case) to be displayed	Direction Wind is blowing
Time and date of observation	Wind Direction Phrase
Observation Station Name	Relative humidity
Temperature	Visibility
"Feels Like" Temperature	UV Index
Weather Description Phrase	UV Index Description
Weather Icon Code	Dew point
Barometric Pressure	Sunrise
Wind Speed	Sunset
Moon Phase Description	Moon Icon Code

The following data in Table 5 is included for Five-Day Forecast.

Table 5. Forecast Data from Weather Extensible Markup Language Data Feed

Location Id	Daypart Day Code (day or night)
City name (mixed case) to be displayed	Weather Icon Code
Time and date of last update	Weather Description Phrase
Day Number of forecast	Weather Description Short
Day of Week	Phrase
Date	Wind Speed
Forecast High Temperature	Wind Gust
Forecast Low Temperature	Wind Direction
Sunrise	Wind Direction Phrase
Sunset	Chance of Precipitation
	Humidity

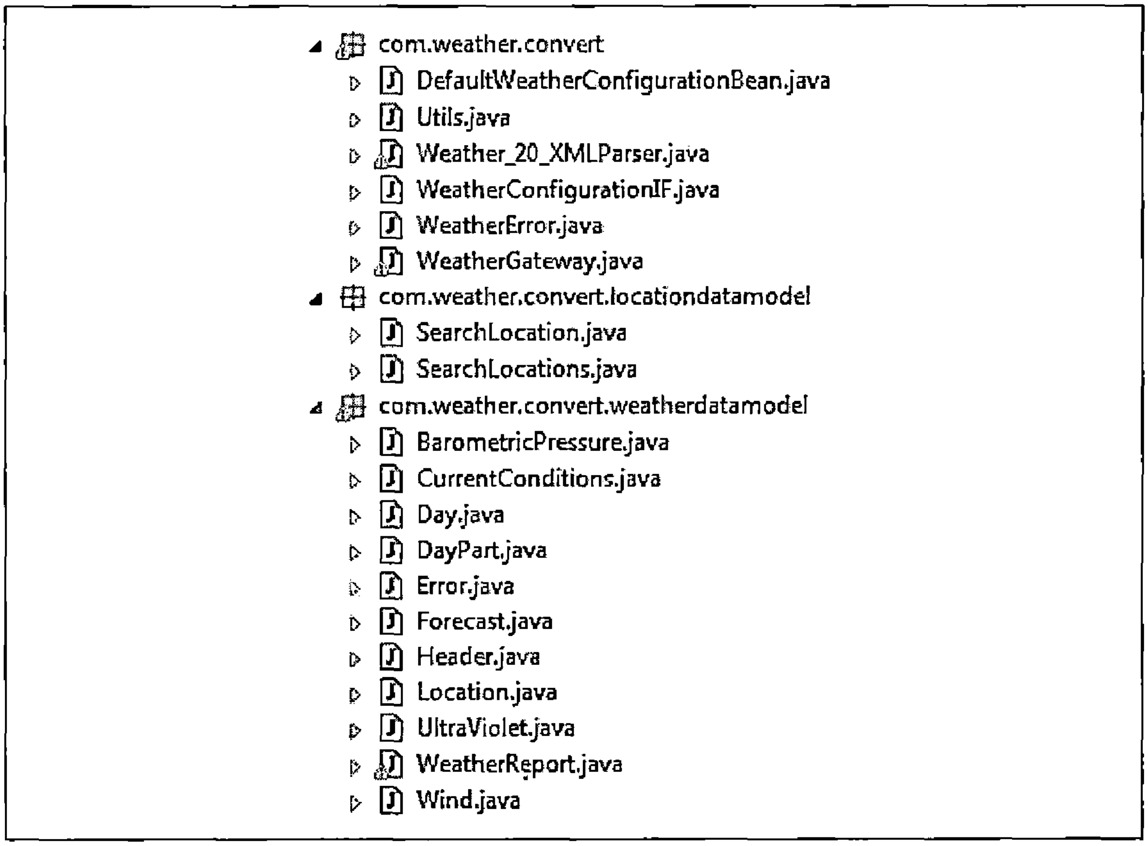
The entire request can be made to access the XML feed over the internet through invoking this hostname xoap.weather.com.

With the place finder web service we get the name of the place we are looking for. But Weather data must be requested for a specific location by a valid locID. The XML Feed works with two types of LocIDs, Type 1 locations are city identifiers and Type 4 locations are 5-digit U.S. postal codes. Since my application looks for places all over the world the Type 4 locations with postal codes does not work. So used the Type 1 locations which get the location ID based on the city name. The location Id can be looked up using this search function at this URL -http://xoap.weather.com/search/search.

After the location ID is retrieved, this ID is sent to weather.com with partner ID and license key to access the current weather conditions and a five-day forecast. The XML Feed will appear as follows:

> http://xoap.weather.com/weather/local/[locID]?cc =*&dayf=5&link=xoap&prod=xoap&par=[PartnerID]&key =[LicenseKey] where dayf is 5-day forecast, [PartnerID] is the unique Partner ID and [LicenseKey] is the unique License Key.

The weather data gives us back a XML document which we need to parse to get the data. It is tedious to parse the response for every request so used Java classes which were provided by www.DeveloperLife.com. Add these classes to PlaceFinderProject in com.weather.convert, com.weather.convert.locationdatamodel and com.weather.convert.weatherdatamodel packages respectively. All the classes used for the weather data conversion is shown in Figure 25 below.



- com.weather.convert
  - DefaultWeatherConfigurationBean.java
  - Utils.java
  - Weather_20_XMLParser.java
  - WeatherConfigurationIF.java
  - WeatherError.java
  - WeatherGateway.java
- com.weather.convert.locationdatamodel
  - SearchLocation.java
  - SearchLocations.java
- com.weather.convert.weatherdatamodel
  - BarometricPressure.java
  - CurrentConditions.java
  - Day.java
  - DayPart.java
  - Error.java
  - Forecast.java
  - Header.java
  - Location.java
  - UltraViolet.java
  - WeatherReport.java
  - Wind.java

Figure 25. Classes for Parsing Weather

So now the data is transformed into a set of Java beans which can be easily integrated with my application. Since I am parsing the XML data into methods there are few dependencies that the code will require to work. Along with the zip file are the jar files which need to be added to the project for compiling the project. Below is the Figure 26 showing UML class diagram of all the classes, methods and relationship between them.

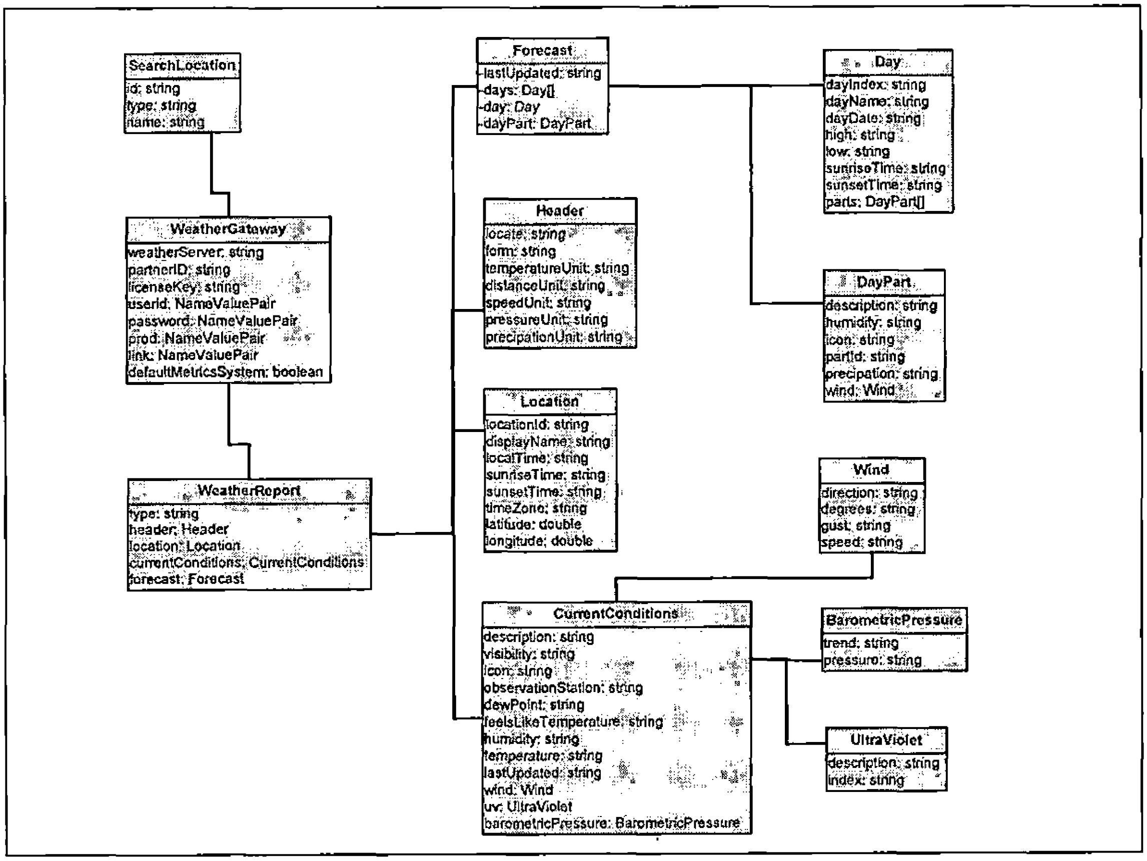


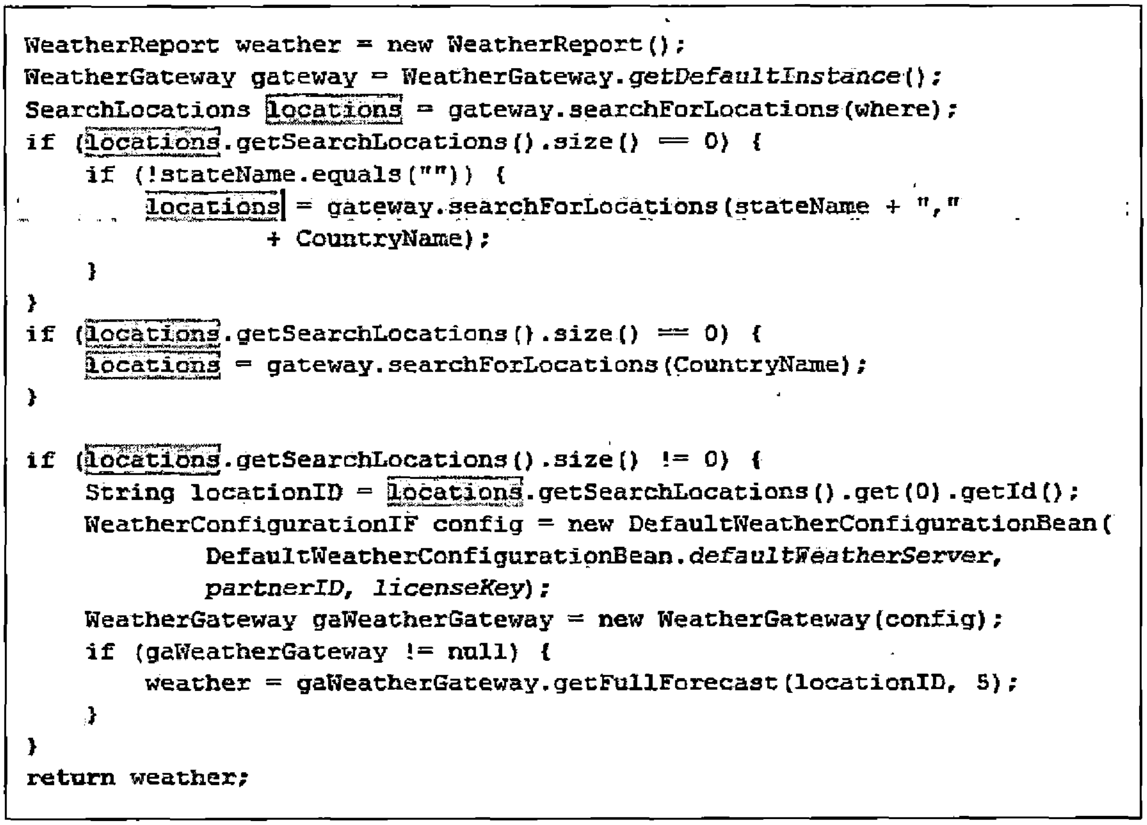
Figure 26. Class Diagram for Parsing Weather Data

Below in Table 6 is a list of Jar files added to the project to resolve errors.

Table 6. Weather Data Parsing Dependencies

jdom.jar	Used as a XML parser to parse the XML document
commons-codec-1.3.jar	Used to encode the request parameters
commons-httpclient-3.0.1.jar	Used to perform the HTTP communications
commons-io-1.2.jar	Used to provide collection of I/O utilities
commons-lang-2.2.jar	Used to provide extra functionality for Java classes
jaxen-core.jar	Used to treat JDOM trees as compiled Java byte code
jaxen-jdom.jar	Used to provide navigation for JDOM trees
saxpath.jar	Used as a event based parser and handle XPath expressions
taskapi.jar	Used to manage multiple tasks

Create a method getWeather() in implementerPlace.Java class. Below in Figure 27 is the code used to access the weather data based on the place name.



```
WeatherReport weather = new WeatherReport();
WeatherGateway gateway = WeatherGateway.getDefaultInstance();
SearchLocations locations = gateway.searchForLocations(where);
if (locations.getSearchLocations().size() == 0) {
    if (!stateName.equals("")) {
        locations = gateway.searchForLocations(stateName + ","
                + CountryName);
    }
}
if (locations.getSearchLocations().size() == 0) {
    locations = gateway.searchForLocations(CountryName);
}

if (locations.getSearchLocations().size() != 0) {
    String locationID = locations.getSearchLocations().get(0).getId();
    WeatherConfigurationIF config = new DefaultWeatherConfigurationBean(
            DefaultWeatherConfigurationBean.defaultWeatherServer,
            partnerID, licenseKey);
    WeatherGateway gaWeatherGateway = new WeatherGateway(config);
    if (gaWeatherGateway != null) {
        weather = gaWeatherGateway.getFullForecast(locationID, 5);
    }
}
return weather;
```

Figure 27. Accessing Weather Data

3.3.4 Directions and Traffic

Create a new directions.jsp page for adding directions and traffic conditions. Follow the steps in section 3.3.2 to add a map on the page using the same map key. For the directions add new GDirections() object which takes in the map and id of the div element that has to be populated with the directions.

gdir = new GDirections(map, document.getElementById("directions"));

Load the GDirections method with from and to addresses.

gdir.load("from: " + fromAddress + " to: " + toAddress);

Create a form with two input fields fromAddress and toAddress which on submit load the GDirections() method passing these values.

```
<form action=“#” onsubmit=“setDirections(this.from.value,
this.to.value); return false”>
 <table align=“center” border=“2” bordercolor=“#A52A2A”
width=“75%” height=“70”>
<tr><td colspan=“2” align=“center”><label style=“font-size:
18px”><b>Enter Directions</b></label>
<tr><td><span>From:</span>
<input type=“text” size=“45” id=“fromAddress” name=“from”
value=“<%= placename%>”/></td>
<td><span>To:</span>
<input type=“text” size=“45” id=“toAddress” name=“to” value=“<%=
placename%>”/>
 </td></tr>
<tr><td align=“right”>
<input name=“submit” type=“submit” value=“Get Directions”
/></td><td>
```

To add traffic to the map create a GTrafficOverlay() object and pass in the traffic options with incidents. addOverlay() method add a new layer to map with the traffic updates.

```
var trafficOptions = {incidents:true};
var trafficInfo = new GTrafficOverlay(trafficOptions);
var toggleState = 1;
function toggleTraffic() {
     if (toggleState == 1) {
       map.removeOverlay(trafficInfo);
     toggleState = 0;
} else {
     map.addOverlay(trafficInfo);
     toggleState = 1;
     }
}
```

3.4 Integration

This section discusses the integration of the place finder web service, Google maps API and weather XML data feed. From section 3.3.1 we get a list of place names from the web service which is shown on the input field of the form. Once a place name along with coordinates is selected from the list and the form is submitted, the request object in the jsp page intercepts the value of the input field. Based on the coordinates the place name can be looked up in the maps.

```
String place = request.getParameter("placename");
String xcordinate = request.getParameter("xcordinate");
String ycordinate = request.getParameter(“ycordinate”);
String action = request.getParameter(“action”);
```

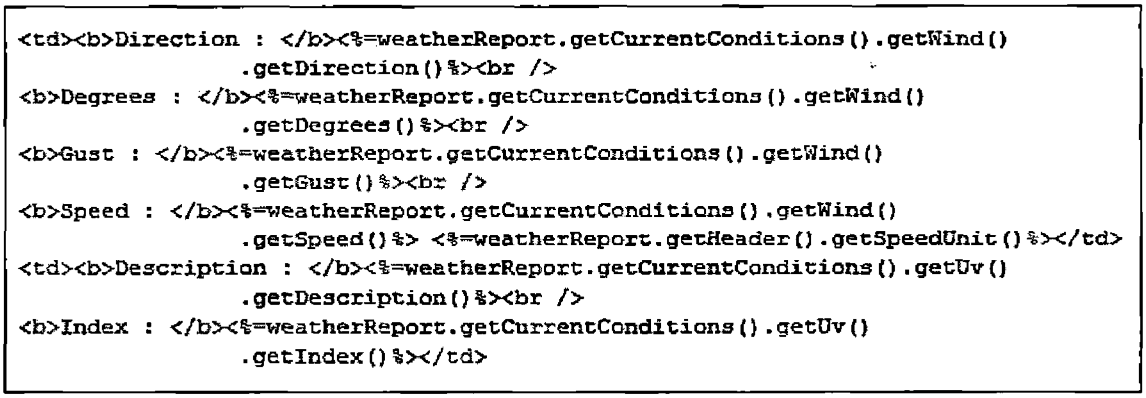
Based on the request the place name is formatted to three substrings place name, x-coordinate, y-coordinate.

```
placename = place.substring(0, place.indexOf(‘:’));
y = place.substring(place.indexOf(‘:’) + 1,
place.lastIndexOf(‘:’));
x = place.substring(place.lastIndexOf(‘:’) +
1,place.length());
```

Once the place name is found out the value is passed into the weather report to get the report.

```
weatherReport = implementerPlace.getWeather(placename);
```

Once the weatherReport is populated with weather and forecast, accessing the data fields on it is easy as shown in Figure 28.



```
<td><b>Direction : </b><%=weatherReport.getCurrentConditions().getWind()
                .getDirection()%><br />
<b>Degrees : </b><%=weatherReport.getCurrentConditions().getWind()
                .getDegrees()%><br />
<b>Gust : </b><%=weatherReport.getCurrentConditions().getWind()
                .getGust()%><br />
<b>Speed : </b><%=weatherReport.getCurrentConditions().getWind()
                .getSpeed()%> <%=weatherReport.getHeader().getSpeedUnit()%></td>
<td><b>Description : </b><%=weatherReport.getCurrentConditions().getUv()
                .getDescription()%><br />
<b>Index : </b><%=weatherReport.getCurrentConditions().getUv()
                .getIndex()%></td>
```

Figure 28. Accessing Methods to Get Weather Data

For mapping on the map the latitude and longitude are populated with x coordinate and y coordinate.

```
function setMarker() {
x = <%= x%>;
y = <%= y%>;
var mar = new GMarker(new GLatLng(x, y));
return mar;
}
```

The flow diagram below in Figure 29 shows how information is passed from one service to the other.

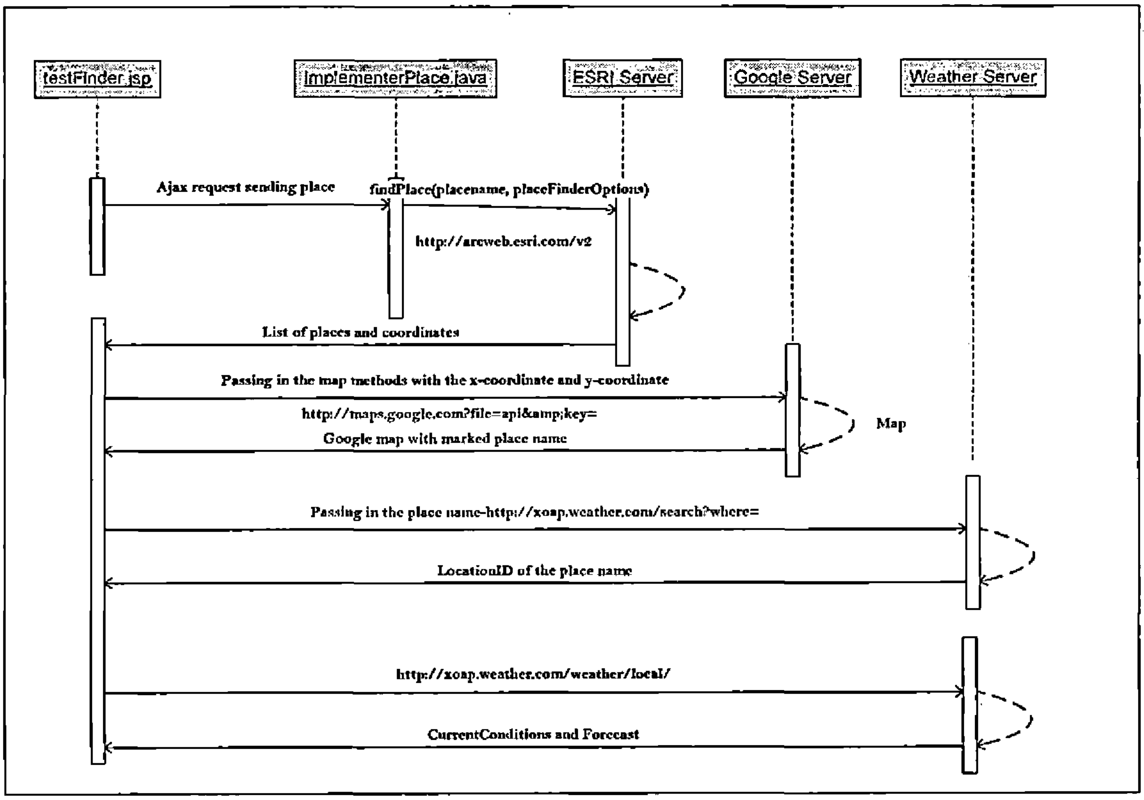

testFinder.jsp
ImplementerPlace.java
ESRI Server
Google Server
Weather Server
Ajax request sending place
findPlace(placename, placeFinderOptions)
http://arcweb.esri.com/v2
List of places and coordinates
Passing in the map methods with the x-coordinate and y-coordinate
http://maps.google.com?file=api&amp;key=
Google map with marked place name
Map
Passing in the place name-http://xoap.weather.com/search?where=
LocationID of the place name
http://xoap.weather.com/weather/local/
CurrentConditions and Forecast

Figure 29. Flow Diagram of Application

To integrate the directions traffic page to the place finder page, place name is passed in the url to the directions and traffic page which is intercepted with the request object on the page.

3.5 Demonstration and Enhancements

In the first demonstration to my advisor Dr. Gomez, suggested some minor enhancements listed below that were implemented in the final application.

1. Add a help page to the application to guide the users on how to use the application.
2. Allow users to input Latitude (X), Longitude (Y) co-ordinates and find the location on the map.
3. Default the place name that has been looked up in the map by the user to find directions and traffic page so as to provide continuity to the application.

Dr. Yu also suggested the following enhancements that have been implemented in the final application.

1. When a user types in the place name and as the matching place names are being fetched by AJAX request, display a message that the request is being processed by the application to convey the

user to wait until the candidate place names are retrieved and displayed for selection.

CHAPTER FOUR

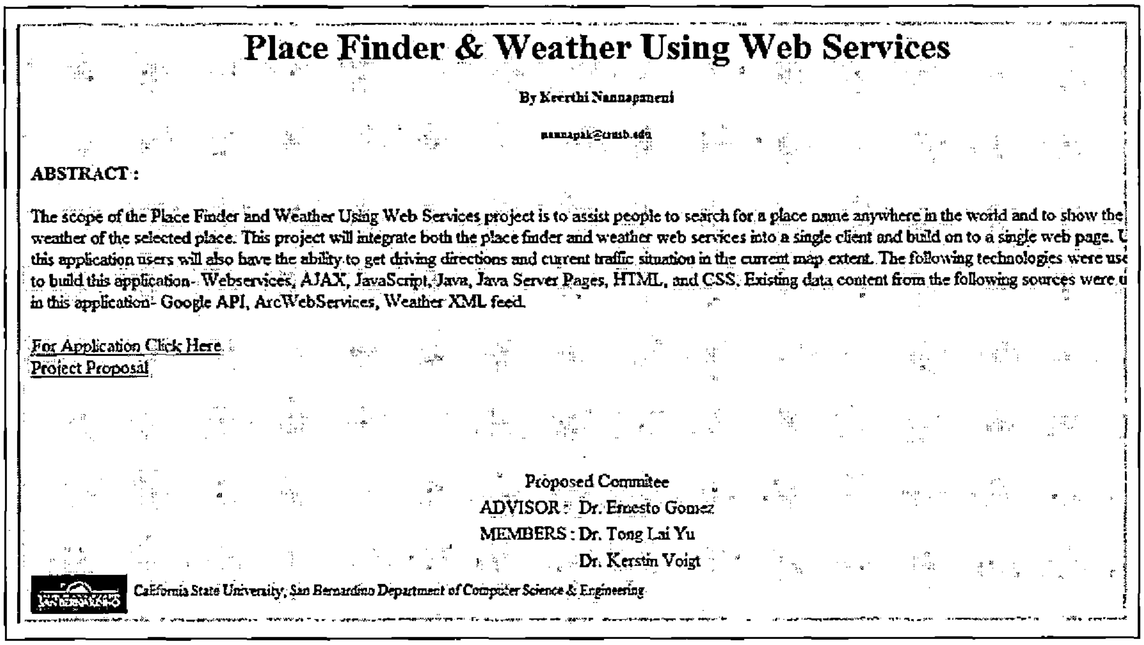
MAINTENANCE

4.0 Application Results

There are three web pages in this application.

1. Introduction to the application page
2. Place finder, weather conditions and forecast page
3. Directions and traffic conditions page

The first web page briefly explains the purpose, features and technologies used in the project. It consists of links to the project proposal and application. Figure 30 below is a screenshot of the introduction page.



Place Finder & Weather Using Web Services

By Keerthi Nannapaneni

nannapak@csusb.edu

ABSTRACT :

The scope of the Place Finder and Weather Using Web Services project is to assist people to search for a place name anywhere in the world and to show the weather of the selected place. This project will integrate both the place finder and weather web services into a single client and build on to a single web page. U this application users will also have the ability to get driving directions and current traffic situation in the current map extent. The following technologies were use to build this application- Webservices, AJAX, JavaScript, Java, Java Server Pages, HTML, and CSS. Existing data content from the following sources were u in this application- Google API, ArcWebServices, Weather XML feed.

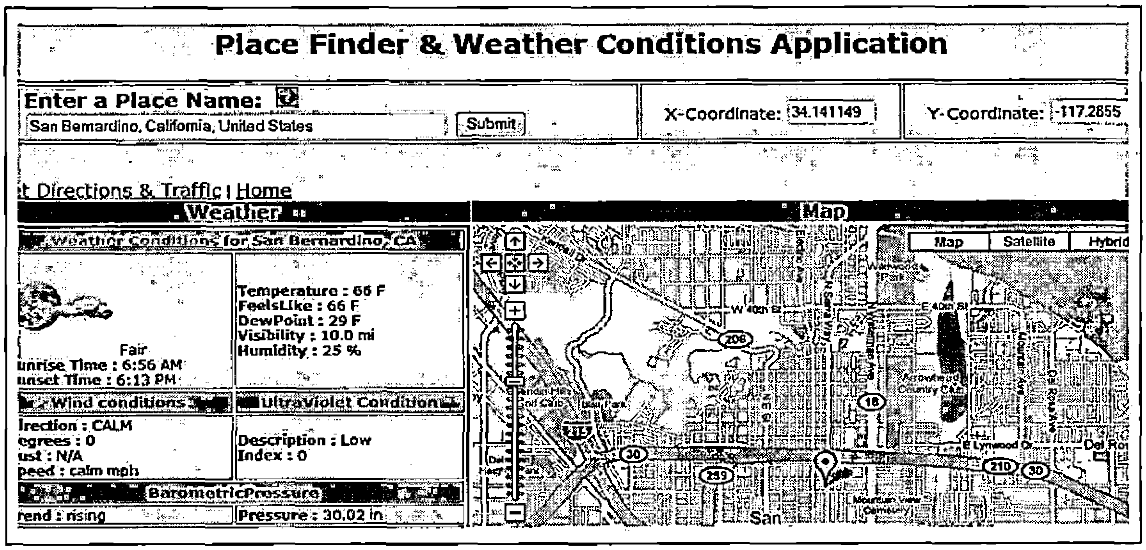
For Application Click Here
Project Proposal

Proposed Commitee
ADVISOR : Dr. Ernesto Gomez
MEMBERS : Dr. Tong Lai Yu
Dr. Kerstin Voigt

California State University, San Bernardino Department of Computer Science & Engineering

Figure 30. Introduction Page

To go to the application, please click on the *For Application* link, this takes the user to the place finder and weather conditions page as shown in Figure 31 below. The default location selected is San Bernardino, California, United States. On the right hand side are the coordinates (latitude, longitude) populated for this place. The current weather conditions, forecast and the map location are shown bottom of the page. By clicking on the 'Get Directions and Traffic' link user will go to Directions and traffic conditions page.



Place Finder & Weather Conditions Application

Enter a Place Name: San Bemardino, Califomia, Unilad States

X-Coordinate: 34.141149

Y-Coordinate: -117.2855

t Directions & Traffic | Home

Weather

Weather Conditions for San Bernardino, CA

Fair
unrise Time : 6:56 AM
unset Time : 6:13 PM

Temperature : 66 F
FeelsLike : 66 F
DewPoint : 29 F
Visibility : 10.0 mi
Humidity : 25 %

Wind conditions
Irection : CALM
egrees : 0
ust : N/A
peed : calm mph

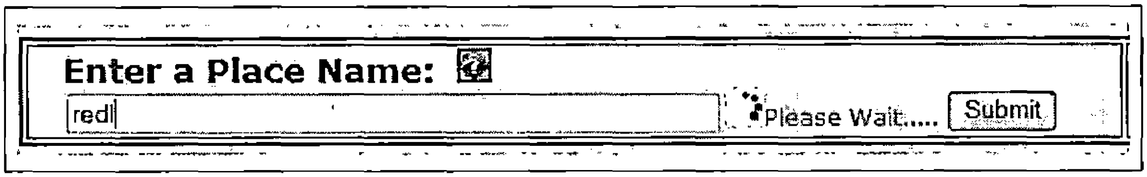
UltraViolet Condition
Description : Low
Index : 0

BaromotricProssure
rend : rising
Pressure : 30.02 in

Figure 31. Place Finder and Weather Conditions

A help icon is located near the search text. When a user clicks on it a window pops up with instructions on how to use the application.

Enter the first 4 characters of a place name box as shown in Figure 32 below and wait for AJAX to show a list of place name that match the first 4 characters.

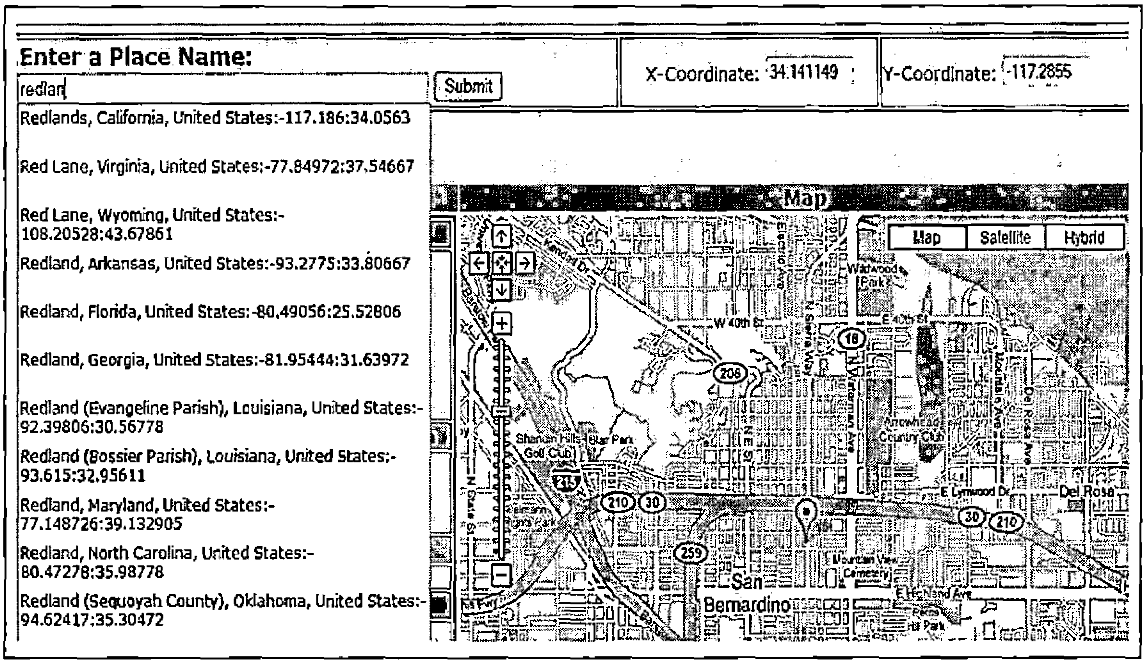


Enter a Place Name:

redl

Figure 32. Place Name

The user can scroll through the list by pressing down arrow or scroll down as shown in Figure 33 below. Once a place name is selected hit submit and the weather and map location will change to the place entered in the place name box.



Enter a Place Name: redlan

X-Coordinate: 34.141149

Y-Coordinate: -117.2855

Redlands, California, United States:-117.186:34.0563

Red Lane, Virginia, United States:-77.84972:37.54667

Red Lane, Wyoming, United States:-108.20528:43.67861

Redland, Arkansas, United States:-93.2775:33.80667

Redland, Florida, United States:-80.49056:25.52806

Redland, Georgia, United States:-81.95444:31.63972

Redland (Evangeline Parish), Louisiana, United States:-92.39806:30.56778

Redland (Bossier Parish), Louisiana, United States:-93.615:32.95611

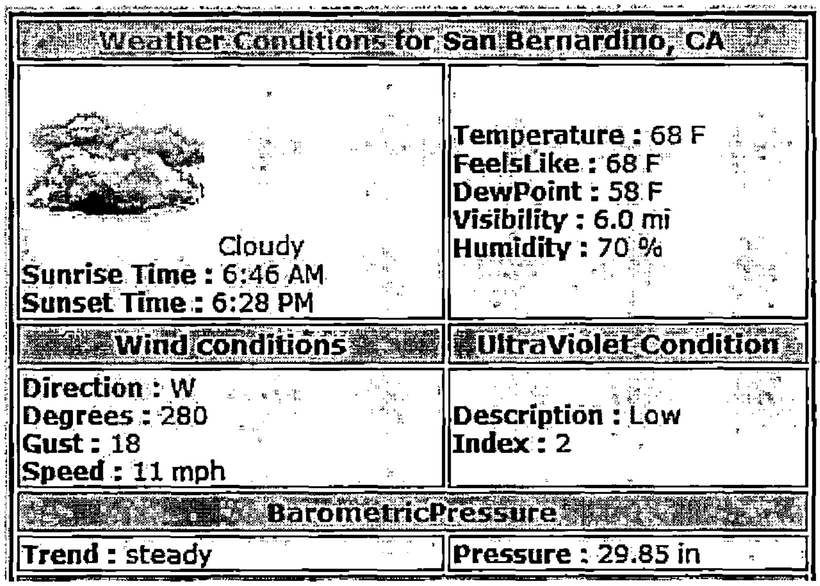
Redland, Maryland, United States:-77.148726:39.132905

Redland, North Carolina, United States:-80.47278:35.98778

Redland (Sequoyah County), Oklahoma, United States:-94.62417:35.30472

Figure 33. List of Place Names

For the place name selected, current and forecast weather conditions will be displayed as shown in Figure 34 below.



Weather Conditions for San Bernardino, CA

Cloudy
Sunrise Time : 6:46 AM
Sunset Time : 6:28 PM

Temperature : 68 F
FeelsLike : 68 F
DewPoint : 58 F
Visibility : 6.0 mi
Humidity : 70 %

Wind conditions

Direction : W
Degrees : 280
Gust : 18
Speed : 11 mph

UltraViolet Condition

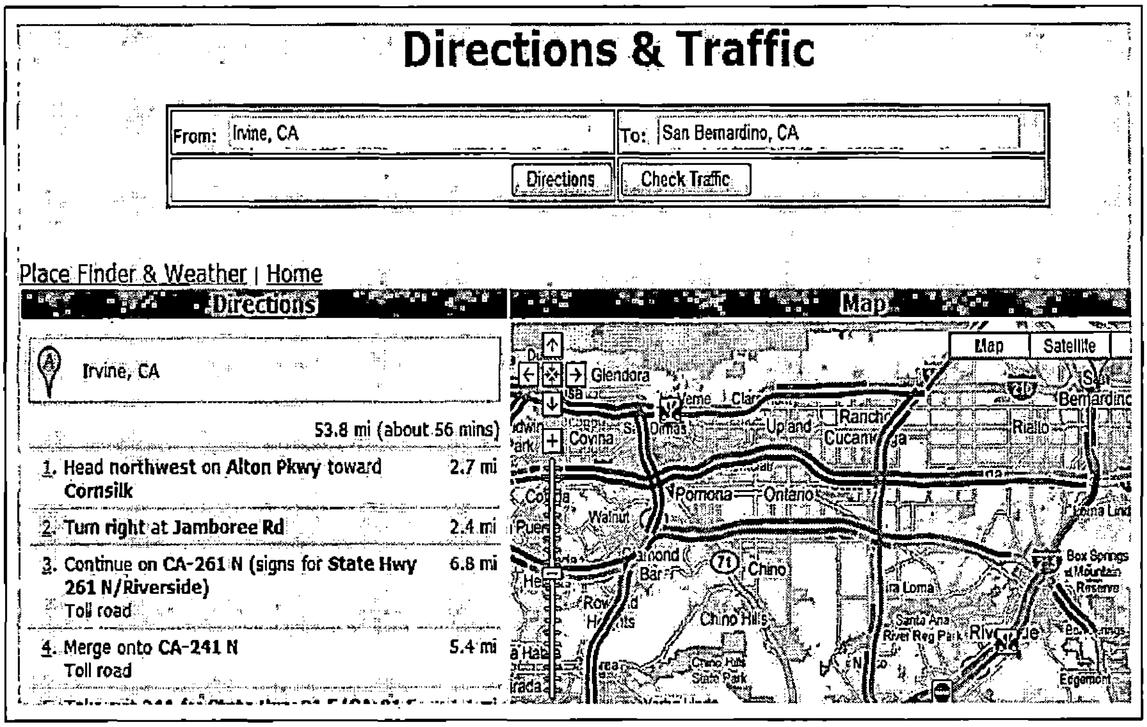
Description : Low
Index : 2

BarometricPressure

Trend : steady
Pressure : 29.85 in

Figure 34. Weather Conditions

To get driving directions, click on get Directions and traffic link. Change the destination or origination accordingly and click on the Directions button. A detailed textual direction along with a highlighted route is shown on the map space on the page. To get traffic update click on Check Traffic button and the current traffic conditions are shown on the current map extent as shown in Figure 35 below.



Directions & Traffic

From: Irvine, CA

To: San Bernardino, CA

Irvine, CA

53.8 mi (about 56 mins)

1. Head northwest on Alton Pkwy toward Cornsilk — 2.7 mi
2. Turn right at Jamboree Rd — 2.4 mi
3. Continue on CA-261 N (signs for State Hwy 261 N/Riverside) — 6.8 mi
   Toll road
4. Merge onto CA-241 N — 5.4 mi
   Toll road

Figure 35. Directions and Traffic

To go back to introduction page or place finder page click on Home or place finder and weather respectively.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

This section describes various conclusions as a result of completing the project and also provides some recommendations.

5.1 Conclusions

Through this project I tried to demonstrate how to research and find appropriate web service and easily make use of them. With the availability of Internet and rich data content that is served through Web services, users can leverage these existing services and tailor them to meet the needs of any particular user segment.

Through my application I tried to validate the following main advantages of using web services to provide valuable information to users by investing minimal resources.

- Outsourced - There is no need to store and maintain data, software and hardware infrastructure. The data used in the project is hosted by various companies and the software used to develop this application was minimal and free of cost.

- Built on Industry Standards - Web services are built using industry standards which makes it easy to build and create other dependent applications and find any help to support development efforts.
- Platform and Language Independent -Can be hosted or used by users from any OS platform or programming language making it transparent to the end user.
- Works of an Internet Connection - With internet penetration more than 70% in US and fast increasing around the world, Web services with rich content are accessible to users around the world.

With the above mentioned benefits come some risks as well. Since web services involve entities out of our control there is a risk of unavailability of service because of a network outage, server problems or even when there are minor changes in specifications of the web service components.

There has to be sufficient notification and communication mechanism between the service providers and consumers to quickly react to the changes made to the components so that the service is not interrupted.

Fortunately we have really reliable services that are available 24/7 and with good help resources to take advantage of the rich content and functionality. There is also a need to easily respond to the ever changing needs by making the Web services easy to configure and use.

5.2 Recommendations

This project is a good start for a novice programmer interested in Web services on how to start the research and implement web services. The documentation in this project can also help developers to choose the right tools and data resources. This project has been only limited to demonstrate a few important common tasks (Place Name, Weather, Driving Directions, etc). This application can be extended further very easily by adding features that complement the existing application. For example, the project can be extended to include fire alerts, flash flooding, air quality, road closures, etc. The code is provided with good documentation and is intended to be understood easily by a programmer with basic understanding of programming principles.

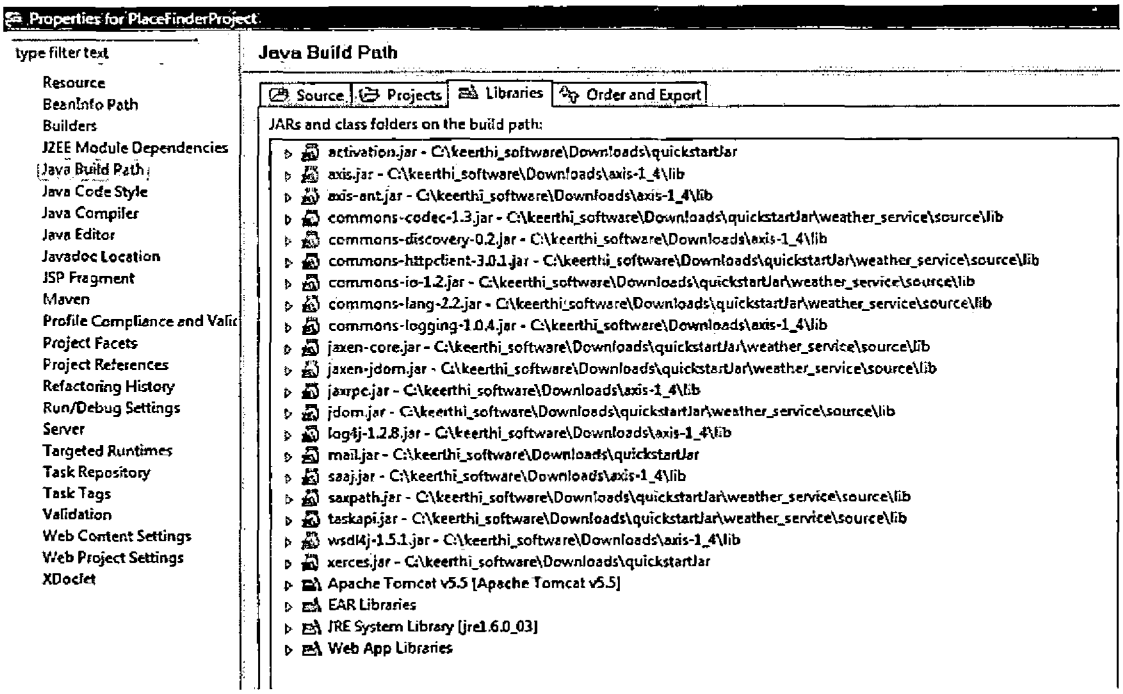
Based on the experience of implementing this project, I would like to recommend that the University create a list of topics or tools that they would like to be supported from the student community. This will allow students to match their interests and strengths with this list and to choose a topic of his or her interest. It is always very motivating to tackle an existing real requirement than a hypothetical situation.

By building this mechanism it is going to mutually benefit both the students and the university.

APPENDIX A

JAVA ARCHIVE FILES

Below is the screen shot of all the Jar Files that were used for setting up of this application.



Java Build Path

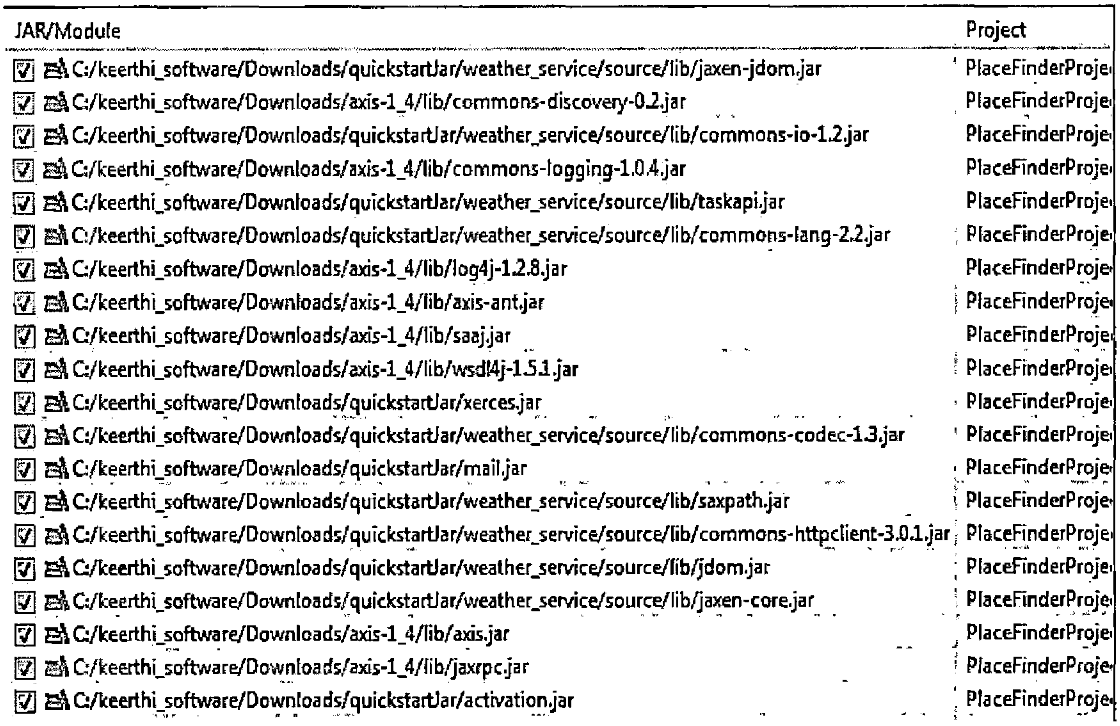
JARs and class folders on the build path:

```
activation.jar - C:\keerthi_software\Downloads\quickstartJar
axis.jar - C:\keerthi_software\Downloads\axis-1_4\lib
axis-ant.jar - C:\keerthi_software\Downloads\axis-1_4\lib
commons-codec-1.3.jar - C:\keerthi_software\Downloads\quickstartJar\weather_service\source\lib
commons-discovery-0.2.jar - C:\keerthi_software\Downloads\axis-1_4\lib
commons-httpclient-3.0.1.jar - C:\keerthi_software\Downloads\quickstartJar\weather_service\source\lib
commons-io-1.2.jar - C:\keerthi_software\Downloads\quickstartJar\weather_service\source\lib
commons-lang-2.2.jar - C:\keerthi_software\Downloads\quickstartJar\weather_service\source\lib
commons-logging-1.0.4.jar - C:\keerthi_software\Downloads\axis-1_4\lib
jaxen-core.jar - C:\keerthi_software\Downloads\quickstartJar\weather_service\source\lib
jaxen-jdom.jar - C:\keerthi_software\Downloads\quickstartJar\weather_service\source\lib
jaxrpc.jar - C:\keerthi_software\Downloads\axis-1_4\lib
jdom.jar - C:\keerthi_software\Downloads\quickstartJar\weather_service\source\lib
log4j-1.2.8.jar - C:\keerthi_software\Downloads\axis-1_4\lib
mail.jar - C:\keerthi_software\Downloads\quickstartJar
saaj.jar - C:\keerthi_software\Downloads\axis-1_4\lib
saxpath.jar - C:\keerthi_software\Downloads\quickstartJar\weather_service\source\lib
taskapi.jar - C:\keerthi_software\Downloads\quickstartJar\weather_service\source\lib
wsdl4j-1.5.1.jar - C:\keerthi_software\Downloads\axis-1_4\lib
xerces.jar - C:\keerthi_software\Downloads\quickstartJar
Apache Tomcat v5.5 [Apache Tomcat v5.5]
EAR Libraries
JRE System Library [jre1.6.0_03]
Web App Libraries
```

APPENDIX B

J2EE MODULE DEPENDENCIES

All the below jars are added to the build path of J2EE module to help the compiler to locate all the dependencies when running on tomcat server.



JAR/Module	Project
☑ C:/keerthi_software/Downloads/quickstartJar/weather_service/source/lib/jaxen-jdom.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/axis-1_4/lib/commons-discovery-0.2.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/quickstartJar/weather_service/source/lib/commons-io-1.2.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/axis-1_4/lib/commons-logging-1.0.4.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/quickstartJar/weather_service/source/lib/taskapi.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/quickstartJar/weather_service/source/lib/commons-lang-2.2.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/axis-1_4/lib/log4j-1.2.8.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/axis-1_4/lib/axis-ant.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/axis-1_4/lib/saaj.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/axis-1_4/lib/wsdl4j-1.5.1.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/quickstartJar/xerces.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/quickstartJar/weather_service/source/lib/commons-codec-1.3.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/quickstartJar/mail.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/quickstartJar/weather_service/source/lib/saxpath.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/quickstartJar/weather_service/source/lib/commons-httpclient-3.0.1.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/quickstartJar/weather_service/source/lib/jdom.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/quickstartJar/weather_service/source/lib/jaxen-core.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/axis-1_4/lib/axis.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/axis-1_4/lib/jaxrpc.jar	PlaceFinderProje
☑ C:/keerthi_software/Downloads/quickstartJar/activation.jar	PlaceFinderProje

REFERENCES

[1] Apache Web services (2008, Oct). About Web services - Axis. Apache, USA. [Online]. Available: http://ws.apache.org/axis/Java/user-guide.html

[2] Barry Burd, John (2005). Eclipse For Dummies, Wiley & Sons.

[3] Chris Ullman and Lucinda Dykes (2007). Ajax Resources—Frameworks and Libraries- Appendix B, Beginning Ajax, Wrox Press.

[4] CodeHaus (2008, Oct). Jaxex: Universal Java XPath Engine. CodeHaus, USA. [Online]. Available: http://jaxen.codehaus.org/

[5] Eclipse. (2008, Oct). Eclipse Home. Eclipse, USA. [Online]. Available: http://www.eclipse.org/

[6] ESRI. (2008, Oct). About ESRI- The GIS Software Leader, USA. [Online]. Available: http://www.esri.com/

[7] Google Maps. (2008, Oct). About Google Maps API. Google, USA. [Online]. Available: http://code.google.com/apis/maps/

[8] Interaktonline (2005, Nov). AJAX: Asynchronously Moving Forward. Interaktonline, Romania. [Online]. Available:http://www.interaktonline.com/Support/Articles/

[9] Java Home (2008, Oct). Glossary of Java Sun home, USA. [Online]. Available: http://Java.sun.com/ products/jlf/ed2/book/HIG.Glossary.html

[10] Java Home. (2008, Oct). About the Java Technology. Sun Java Home, USA. [Online]. Available: https://Java.sun.com/docs/books/tutorial/getStarted/intro/definition.html

[11] Java-Tips (2008, Oct). What is the difference between JDK and JRE, USA. [Online]. Available: http://www.Java-tips.org/Java-se-tips/Java.lang/what-is-the-difference-between-jdk-and-jre.html

[12] Kevin Mukhar, Chris Zelenak, James L. Weaver and Jim Crume. (2005, Oct). Beginning Java EE 5: From Novice to Professional

[13] Matthew Moodie (2005). Pro Jakarta Tomcat 5, Appress.

[14] Raghavan. (2008, Sep). What is Service-Oriented Architecture?. RoseIndia, India. [Online]. Available: http://www.roseindia.net/webservices/what_is_service_oriented_architecture.shtml

[15] Rose India (2008, Jun). Tutorials on Java Server Pages Tutorial. Roseindia, India. [Online]. Available: http://www.roseindia.net/jsp/Javaserverpagestutorial.shtml

[16] RSS Feed. (2007, Jul). What is RSS? RSS Explained, USA. [Online]. Available: http://www.whatisrss.com/

[17] SoapUser. (2001, Mar). SOAP Basics. SoapUser, USA. [Online]. Available: http://www.soapuser.com/basics1.html

[18] Software AG. (2000, Jun). XML Basics. SoftwareAG, USA. [Online]. Available: http://www.softwareag.com/xml/about/starters.htm

[19] Tomcat. (2008, Oct). Apache Tomcat Home. Apache, USA. [Online]. Available: http://tomcat.apache.org/

[20] Ueli Wahli. (2006, Sep). Web Services Handbook For WebSphere Application Server 6.1, IBM Redbooks.

[21] W3Schools (2008, Oct). CSS Tutorial. W3schools, USA. [Online]. Available: http://www.w3schools.com/css/

[22] W3Schools (2008, Oct). HTML Tutorial. W3schools, USA. [Online]. Available: http://www.w3schools.com/html/default.asp

[23] Weather (2008, Oct). About Weather XML Data Feed. Weather.com, USA. [Online]. Available: http://www.weather.com/services/xmloap.html

[24] Wikipedia. (2008, Oct). About Ajax Programming. Wikipedia, USA. [Online]. Available: http://en.wikipedia.org/wiki/AJAX

[25] Wikipedia. (2008, Oct). About Apache Tomcat. Wikipedia, USA. [Online]. Available: http://en.wikipedia.org/wiki/Apache_Tomcat

[26] Wikipedia. (2008, Oct). About Cascading Style Sheets. Wikipedia, USA. [Online]. Available: http://en.wikipedia.org/wiki/Cascading_Style_Sheets

[27] Wikipedia. (2008, Oct). About HTML. Wikipedia, USA. [Online]. Available:http://en.wikipedia.org/wiki/HTML

[28] Wikipedia. (2008, Oct). World Wide Web Consortium. Wikipedia, USA. [Online]. Available: http://en.wikipedia.org/wiki/World_Wide_Web_Consortium

[29] Wikipedia. (2008, Oct). About Application Programming Interface. Wikipedia, USA. [Online]. Available: http://en.wikipedia.org/wiki/API

[30] Wikipedia. (2008, Oct). About Object-oriented programming. Wikipedia, USA. [Online]. Available: http://en.wikipedia.org/wiki/Object_oriented

[31] Wikipedia. (2008, Oct). About HTTP. Wikipedia, USA. [Online]. Available: http://en.wikipedia.org/wiki/HTTP

[32] Wikipedia. (2008, Oct). About Script.aculo.us. Wikipedia, USA. [Online]. Available: http://en.wikipedia.org/wiki/Script.aculo.us

[33] Windows Live Developer Center (2008, Oct). What is WSDL?. MSDN, USA. [Online]. Available: http://msdn.microsoft.com/en-us/library/bb126207.aspx