University Of Leicester

Computer Science Department

MSc. Web Applications and Services

HiPerVison Projects on Shelton Vision's Textile Inspection Systems: Backend web-based interface

Malavika Reghunathan Nair

29 September 2014

An Interim report submitted in fulfilment of the requirements for Course CO7501- Individual Project, Department of Computer Science, University of Leicester.

Supervised by: Dr. Reiko Heckel

Summary:

This project involves working on the Shelton WebSPECTOR surface inspection system which is currently used in various industries to inspect materials for defects. The system uses a combination of vision hardware (lights, cameras and electronics) and software. There are two primary software platforms. The first is the front-end (written in C#) where the operator interface, defect analysis and system co-ordination is done. The second is the back-end (written in C++) where the image processing and defect detection is carried out.

The outline aim of the project is to explore the possibilities of using existing web technologies and services to provide a web-based interface to allow engineers to interrogate the back-end remotely to troubleshoot any faults in identifying defects and adjust parameters.

List of Acronyms:

HTTP	Hyper Text Transfer Protocol
XML	eXtensible Markup Language
REST	REpresentational State Transfer
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier
CGI	Common Gateway Script
MIME	Multi-Purpose Internet Mail Extensions
AJAX	Asynchronous JavaScript + XML
IDE	Integrated development environment
JAX-RS	Java API for RESTful Services
PHP	Hypertext Preprocessor
JSON	JavaScript Object Notation
SV	Shelton Machines Ltd

Table of Contents:

List of Figures:

List of Tables:

Chapter 1

1. Introduction

1.1 Overall aim of the project

'Shelton Vision' (SV) a company that works on textile inspection systems currently has a desktop based application for their engineers to access and write information from and to a product database. The database holds information viz. material images, process graphs and other displays.

The company's current vision is to have a back-end web based interface to allow its engineers to interrogate the backend remotely to troubleshoot any faults in identifying defects and adjust parameters.

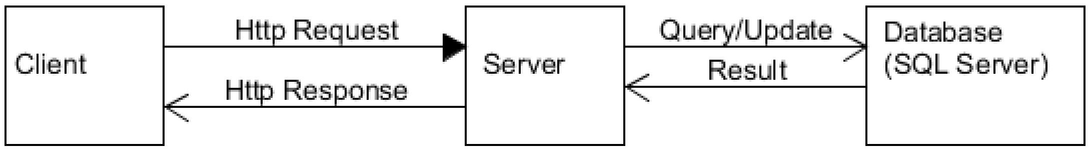


Figure 1: High-Level Architecture

The aim of this project is to do a feasibility study of current web technologies available in the market and then decide on the best technology to implement a web application to replace the existing desktop based application. Upon discussion with SV, for study purposes only a few core functionalities of the application are taken into consideration at this stage.

The performance and quality of the implemented web application will be tested by using the current desktop application as a baseline.

1.2 Objectives

The project assesses the feasibility of implementing a Web-based interface for the engineers to access the information linked to a product database. The information the engineer sees is a mixture of material images, process graphs and other displays. The existing system works under the Microsoft Windows XP operating system and a MS SQL database. It has been developed over 15 years with a mixture of Visual Basic 6, C and assembler libraries, SQL procedures, and WiT(visual programming package).

The project is supervised in cooperation with SV, who provides the requirements and technical support.

The main objective of this project is to obtain a basic understanding of client-server programming, obtain a good grasp of client-side scripting techniques and finally draw a conclusion to use a specific scripting to achieve quality results.

After discussing with SV, the functionalities to be implemented have been decided as follows:

i. Live streaming of images at client side. Images are of size 4MB stored at a location both client and server can access. Image paths are stored in the database.

ii. Update parameters in the database

iii. Display graph at client side from vector data

1.3 Minimum Requirements

- Successful implementation of a database to store the test data
- Successful implementation of an IDE
- Successful implementation of a web server and servlet container
- A framework for developing web services
- Evaluate the performance and quality by comparing the functionality with the existing desktop application

1.4 Schedule

A Gantt chart was prepared to manage the allocation time of the project tasks. The schedule starts from June 1st 2014 and each column represents end of week. There were regular meetings with the supervisor and SV to discuss on the progress and for technical support. It helped to decide on how to go about the solution. Sufficient time was allocated to do the background research of available technologies and to set up the environment.

Below is the embedded Gantt chart file.

Gantt Chart.xlsm

1.5 Report Structure

The following is the structure of the report.

- Chapter 2 describes the background research on client-server architecture, basics of web technology, specifications used for web services, technologies available for client-side scripting and a brief description of a framework that can be used to build web services.

- Chapter 3 describes the setting up of environment essential to perform a client-server task. Basically, the IDE, database, Operating system, Server etc and the obstacles faced during the progress

- Chapter 4 describes the implementation of a RESTful web service

- Chapter 5 describes the implementation of client and client using JavaScript, JQuery and AJAX

1.6 Reflective Analysis

According to the initial plan, a web service which is able to communicate to C++ applications was a requirement. But it was later decided to give more focus to the Client-side browser application at this stage.

Chapter 2

2. Literature Review

2.1 Introduction

The web is a distributed, dynamic and large information repository [3]. Communication over the web or internet can be broken down to two interested parties: *Clients and Servers*. The machine providing services are servers. Clients are the machine used to connect to those services [10]. *Services* are self-contained modules-deployed over standard middleware platforms- that can be described, published, located, orchestrated and programmed using technologies over a network [4].

A web browser is the web client which acts on behalf of the user. The browser contacts the web server and sends a request for the information and receives the information and displays it on the user's computer [10]. Fig 2.1a shows how a basic web technology works.

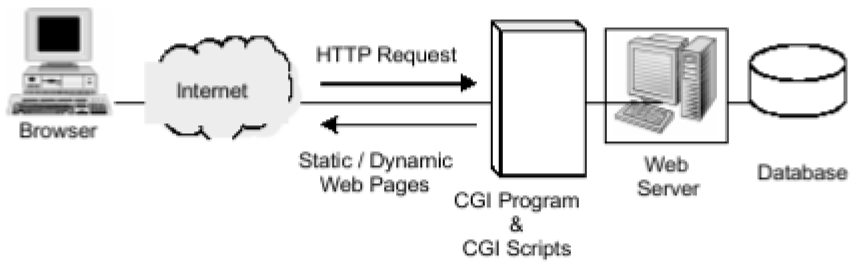


Figure 2: Web Technology [10]

The figure 2.1b illustrates the steps for a web page to access the database. A web browser cannot directly access a database. Most of the cases, browsers are a program running on the web server which acts as an intermediary to the database [10].

When the user hits a URL or clicks a submit button on the web page, the browser sends the request to the web server, which passes it to the Common Gateway Script (CGI). The CGI loads a library which sends the SQL commands to the SQL database server. The database server then executes the query and sends the result to the CGI script. The CGI script generates an HTML document and writes it to the web server. The web server sends the HTML page back to the remote user [10].

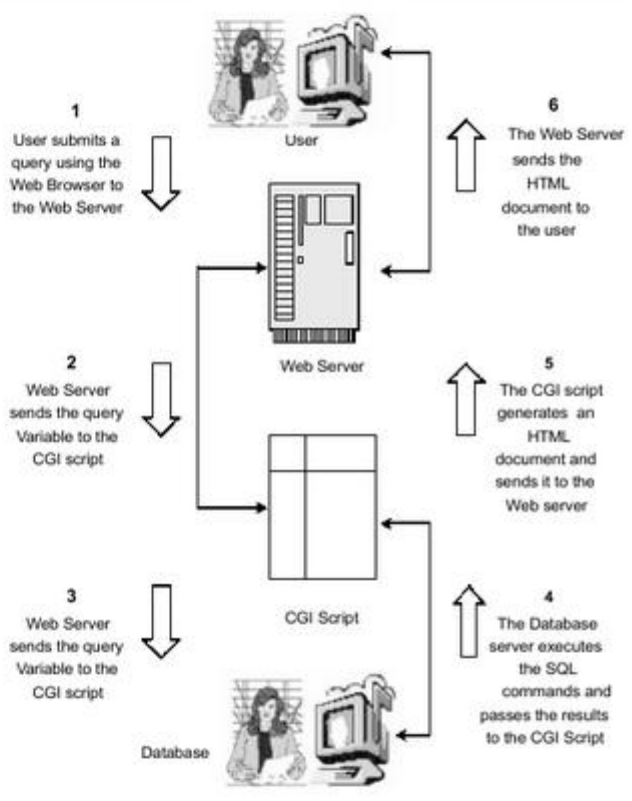

1
User submits a query using the Web Browser to the Web Server
User
6
The Web Server sends the HTML document to the user
Web Server
2
Web Server sends the query Variable to the CGI script
5
The CGI script generates an HTML document and sends it to the Web server
3
Web Server sends the query Variable to the CGI script
CGI Script
4
The Database server executes the SQL commands and passes the results to the CGI Script
Database

Figure 3: Communication between a user and web-based database [10]

The main goal of web service is to exchange information among applications in a standard way [9]. Two most widely used approaches for web service development are SOAP and REST (Representational State Transfer).REST has been accepted widely as a simpler alternative to SOAP and WSDL based web services [8].

2.2 SOAP, REST and JSON

SOAP, REST and JSON are three specifications of web services.

SOAP defines a communication protocol for web services. WSDL enables service providers to describe their applications. UDDI offers a registry service that allows advertisement and discovery of web services [3]. XML is used to define Simple Object Access Protocol (SOAP).

REST (Representational Stat Transfer) defines a set of architectural principles by which you can design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages [8].Basically, web services are viewed as resources and can be identified by their URLs. Client and server communicate by sending and receiving representation of resources. Resources are commonly represented using JSON rather than XML because it is more compact than XML and it can be used with almost all programming languages including JavaScript [5]. JAX-RS uses annotations to simplify RESTful web service development. By adding annotations, we can define resources and can define the operations or actions to be performed on those resources.

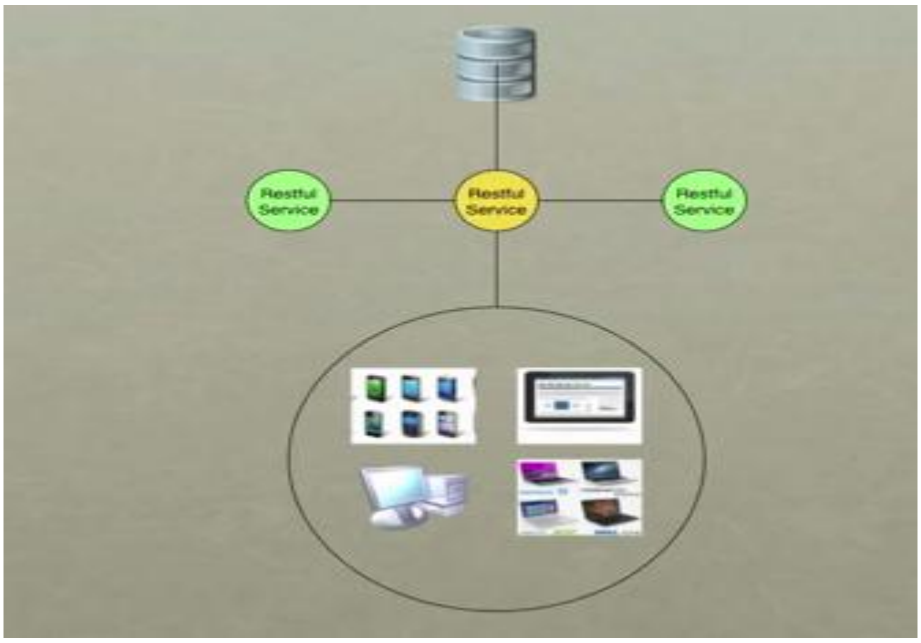


Figure 4: RESTful Web Service

JSON (JavaScript Object Notation) is an open standard format that uses a non-strict subset of JavaScript. Information is exchanged using data objects in the form of attribute-value pair. The MIME type for JSON text data is “application/json”.

JSON is much simpler than XML and is a better data exchange format.

JSON Sample:

```
{
   “id”: 1,
   “name”: “Dave”,
   “city”: “London”
  “gender” : { “type” : “male”
                  },
  “phone number” :{ “type” : “work”,
```

“number” : “000 007 131”

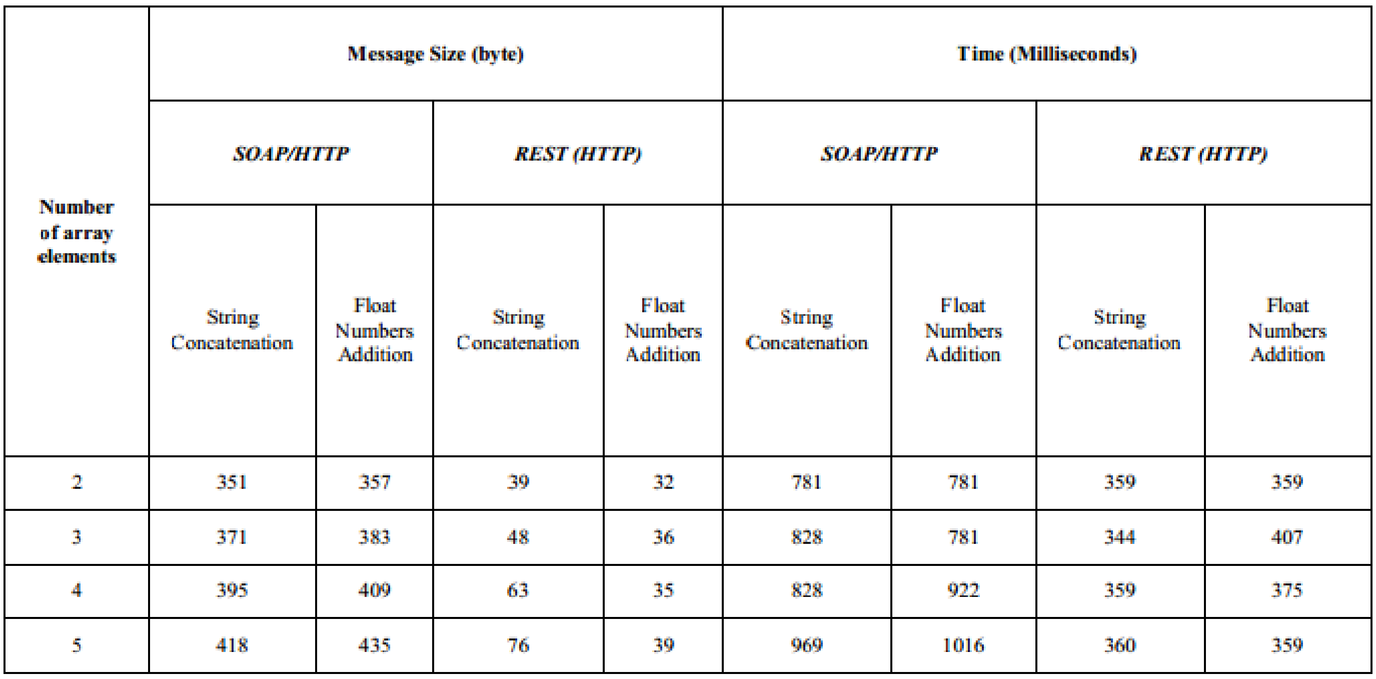
}

}

We could use a combination of web service specifications in order to obtain a better performance.

Advantages of REST over SOAP

Web services performance is an important factor. SOAP communications causes network traffic, higher latency and processing delays. To overcome this limitations the RESTful architecture is used. REST is a lightweight, easy and better alternative for the SOAP [9]. Table 1 illustrates a performance comparison of SOAP and REST in terms of message size and time.



Number of array elements	**Message Size (byte)**				**Time (Milliseconds)**			
	SOAP/HTTP		***REST (HTTP)***		***SOAP/HTTP***		***REST (HTTP)***	
	String Concatenation	Float Numbers Addition	String Concatenation	Float Numbers Addition	String Concatenation	Float Numbers Addition	String Concatenation	Float Numbers Addition
2	351	357	39	32	781	781	359	359
3	371	383	48	36	828	781	344	407
4	395	409	63	35	828	922	359	375
5	418	435	76	39	969	1016	360	359

Table 1: Performance result of SOAP and REST [11]

2.3 Client-Side scripting techniques

2.3.1 JavaScript, jQuery and AJAX

jQuery is a free, open-source and cross-platform JavaScript library used to simplify the client-side scripting of HTML. It has become very popular today and mostly used to develop dynamic web pages. Using jQuery library eliminates cross-browser incompatibilities.

The library provides a general-purpose abstraction layer for common web scripting [15] by taking a lot of common tasks and wrapping them into methods that can be invoked by a single line of code. In addition the framework comes with various plug-ins that are constantly being developed to add new features [15].

The use of JQuery has several advantages over several other JavaScript libraries. Some of them are listed below:

- Ease of use: This is one of the primary advantages of using JQuery. As mentioned above, the level of abstraction that the framework provides means that a task may be performed more easily with lesser lines of code than when using most alternatives.
- Library Size: The large library that JQuery provide allows performing more functions in comparison to other JavaScripts. In addition a compressed version of the library is only around 90 k, which is very small.
- Documentation and Tutorials: JQuery’s dedicated website provides ample information, tutorial and examples to demonstrate the use of the library. In addition it has got a large developer community [17].
- Ajax support: The jQuery library has a full suite of Ajax capabilities that can access by making use of the provided APIs. Actions can be performed on pages without requiring the entire page to be reloaded. [16]

However with these come certain disadvantages as well. They are listed below:

- Limited Functionality: Since jQuery is a framework that provides an abstraction over JavaScript, there may be inevitable cases where the raw JavaScript might have to be used depending on the customization required for example on a webpage.
- jQuery javaScript file: The jQuery file is required to run jQuery commands. Though the size of the file is relatively small, it is still an overhead on the client computer and as well as the web server in certain cases [17].

It can be summarized that the advantages of using the jQuery library clearly out-weighs its disadvantages and hence is clearly a potential candidate for use in this project.

AJAX is the ability of a webpage to send and retrieve data asynchronously from a server, without interfering with the display and actions on the webpage. JSON is mostly used in AJAX instead of XML. Ajax apps are browser and platform independent.

2.3.2 PHP

PHP is a widely used open source programming language that can run on various platforms (Windows, Linux, Unix, MAC OSX etc.) and is executed on the server. Though it is mainly used as a server-side scripting language, its use as a general purpose programming language is common. In more detail, PHP can be used for three primary purposes:

- Server-side scripting
- Command-line scripting
- Client-side GUI applications

PHP also provides a library to perform common tasks such as data abstraction, error handling etc. with the PHP Extension and Application Repository (PEAR) framework [14].

2.3.3 ASP.NET

ASP.Net was developed by Microsoft to develop dynamic web sites, web application and web services. It is a free, open-source framework for building web sites and web applications using HTML, CSS and JavaScript [18].

2.4 Jersey Framework

“Jersey RESTful Web Services framework is an open source, production quality JAX-RS reference implementation for building RESTful Web Services in Java” [6].Jersey produces and consumes RESTful web services. In Jersey framework, the following methods are used which are defined in the HTTP/1.1 protocol.

- @GET is used to retrieve data or perform a query on a resource. The data returned from the web service is a representation of the requested resource [7]. This is the most-used, read only , public access method
- @POST is used to create a new resource. The web service may respond with data or status indicating success or failure [7]. With HTTPS, we can protect data
- @PUT is used to update existing resources or data [7]. But it can also be used for inserting or adding data.
- @DELETE is used to remove a resource or data [7].
- @HEAD used to return meta-data of the resource

Sample code:

```
@Path("/service/")
public class Rest_Service {

    @Path("/database/imagepath/*")
    @GET // HTTP verb is required to access this method
    @Produces(MediaType.TEXT_HTML)
    public String getImagePath() throws Exception {

            // code goes here
        }
}
```

2.5 Conclusion

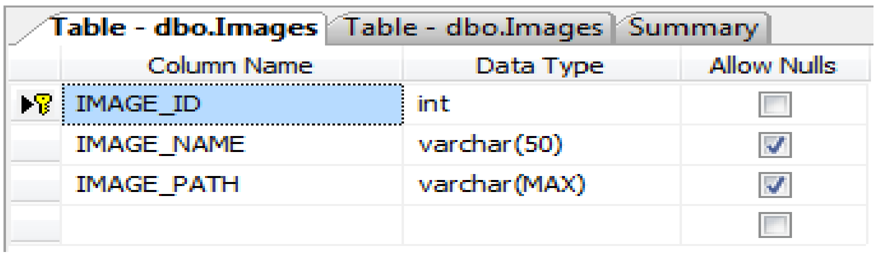
Based on the background research, it was decided to create a Java RESTful service and do a feasibility study of different client-side scripting techniques.

Chapter 3

3. Environment Set Up

3.1 Database – MS SQL

The database used for this project is MS SQL 2003 as recommended by the client (Shelton Machines Ltd). A test database was created whose specifications are shown in figure5.



Column Name	Data Type	Allow Nulls
IMAGE_ID	int	☐
IMAGE_NAME	varchar(50)	☑
IMAGE_PATH	varchar(MAX)	☑
		☐

Figure 5: Test Database Specification

The images are of size 4MB.For time being, the images are stored in the C drive of the machine.

3.2 IDE - Eclipse

Using Eclipse IDE for development of Java RESTful service simplifies the task significantly and offers several tools. Eclipse Indigo which comes along with Oracle Weblogic 12c installation in used in this project.

With a new Dynamic Web Project created in the eclipse IDE for Java EE developers, it is possible to create both service and client in the same package through a web design view. Still the coding needs to be done manually. Also it makes the deployment of the code to the server easy.

In order to make the Java RESTful service work, the following Java library files were added:

- Jar file from http://Jersey.java.net/
- Jar file from http://Jackson.codehaus.org/

3.3 Server – Oracle WebLogic

This project makes use of a Java application Server to deploy the service. “Oracle WebLogic Server 12c is the industry's best application server for building and deploying enterprise Java EE applications with support for new features for lowering cost of operations, improving performance, enhancing scalability and supporting the Oracle Applications portfolio “[12].

Oracle Weblogic 12c was installed with the default settings and Oracle WebLogic 12.1.1 is used for all work in this project. It operates by default on port 7001. In this project, service is deployed to it using Eclipse IDE. A new domain is created with a username and password so that we can login to admin console of the server. In order to connect the server with the database, a data source needs to be created. An SQL data source is created from admin console of the server and a connection is established to the database created in MS SQL.

3.4 Obstacles faced during set up

Problem: Installation of MSSQL 2003 in Windows 8 Operating System unsuccessful.

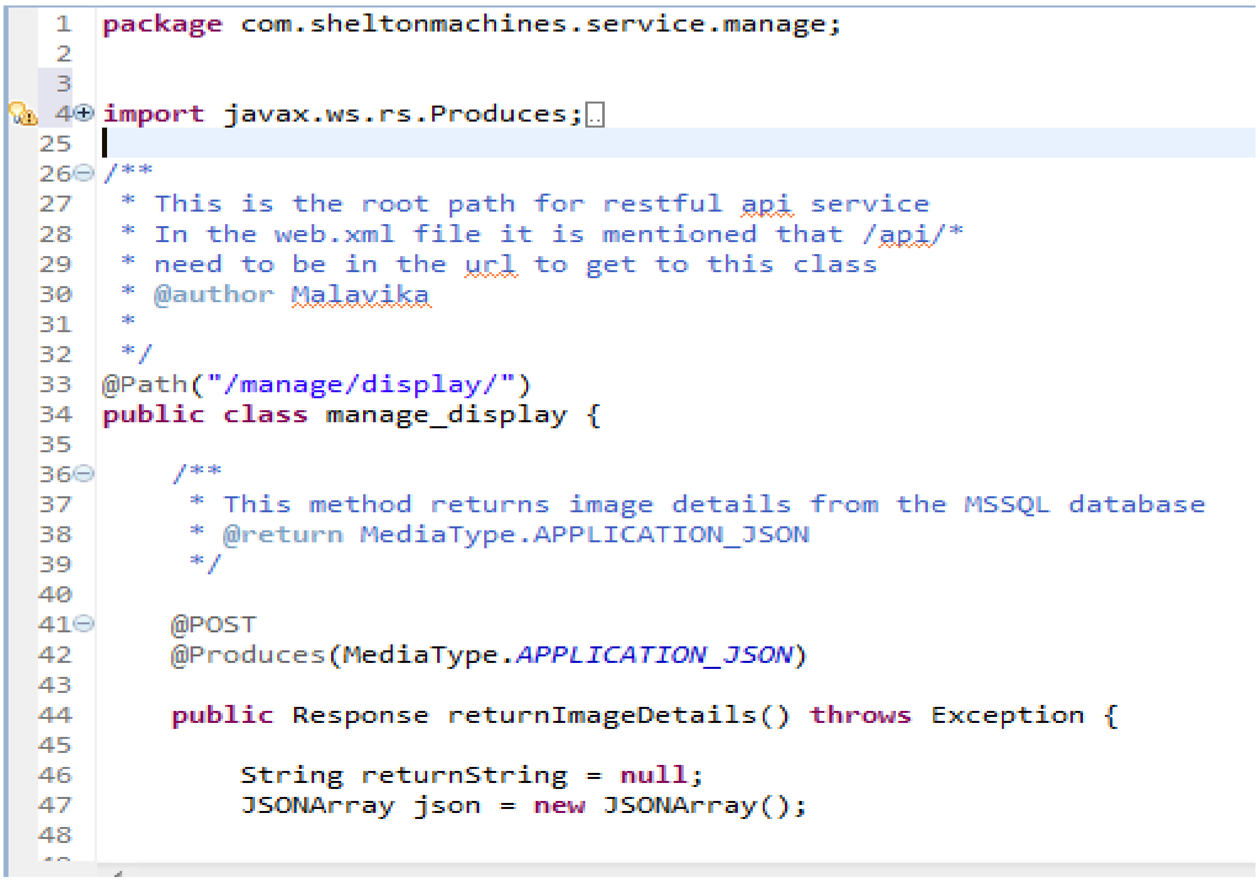
Solution: After discussing with SV, ordered Windows 7 Home Premium OS online as it was not physically available to purchase. Following the installation of Windows 7, MS SQL 2003 was successfully installed.

Chapter 4

4. Implementation of a RESTful web service

The REST Service is written in Java and runs on Oracle WebLogic Server.The service is written in the template provided by Eclipse when a new dynamic web project is created. It loads the necessary libraries and provides options such as configuring the server. Oracle WebLogic server is then added as a new server to the project. The rest of the coding for the service should be done manually.

Figure 6 shows a code snippet of the RESTful web service. As seen in line 41, the HTTP verb “POST” is used to invoke the service method.



```
 1  package com.sheltonmachines.service.manage;
 2
 3
 4⊕ import javax.ws.rs.Produces;
25
26⊖ /**
27   * This is the root path for restful api service
28   * In the web.xml file it is mentioned that /api/*
29   * need to be in the url to get to this class
30   * @author Malavika
31   *
32   */
33  @Path("/manage/display/")
34  public class manage_display {
35
36⊖     /**
37       * This method returns image details from the MSSQL database
38       * @return MediaType.APPLICATION_JSON
39       */
40
41⊖     @POST
42      @Produces(MediaType.APPLICATION_JSON)
43
44      public Response returnImageDetails() throws Exception {
45
46          String returnString = null;
47          JSONArray json = new JSONArray();
48
```

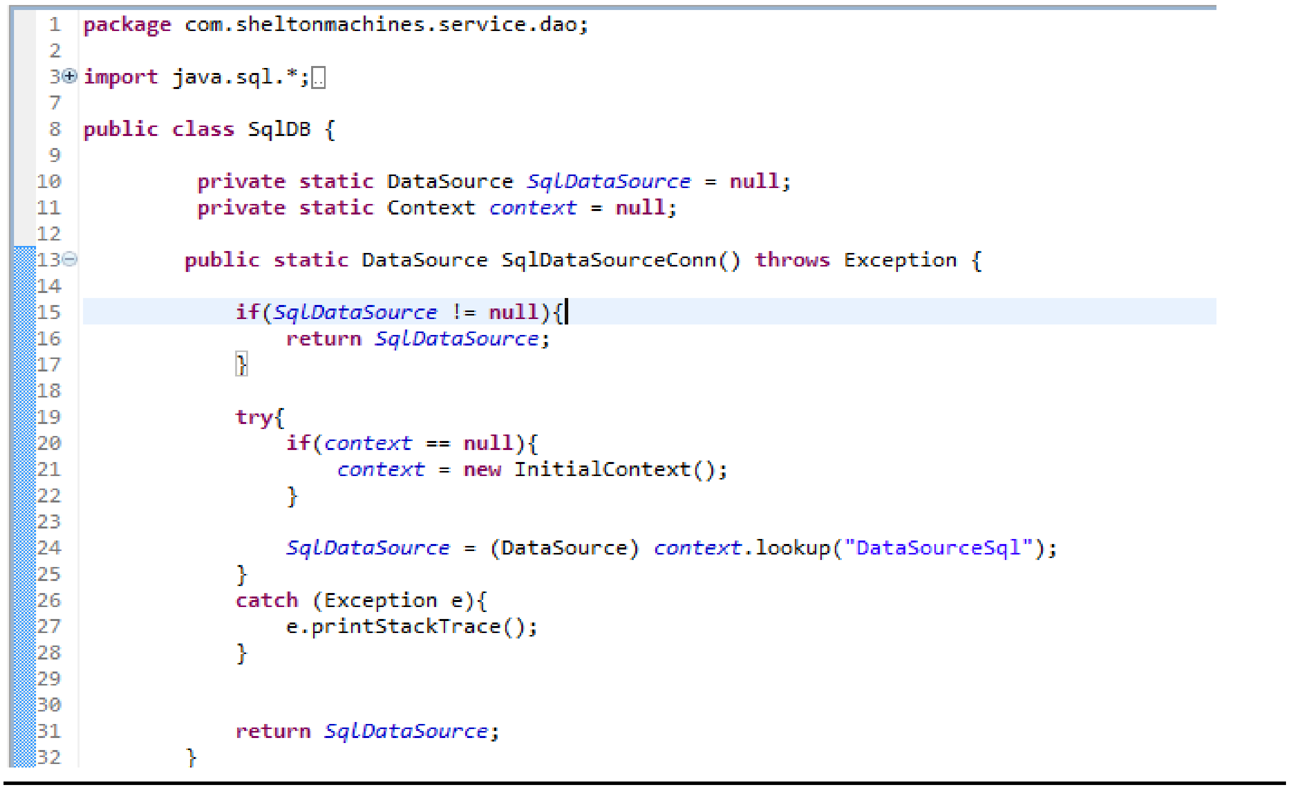
Figure 6: Code snippet of Java RESTful Service Code snippet

The method retrieves the SQLDataSource of WebLogic which was created as explained in section 3.3 and returns a connection. The code snippet is showed in Figure 7. The method response is the image path to the client as a JSON object. Code snippet of the method doing the SQL queries is shown in Figure 8.

Three packages were used to separate the RESTful service from SQL Logic and JSON object mapping. The advantage of this is that if the REST service needs changing, there is no risk of breaking the SQL connection and logging packages. The structure of the packages is shown in figure 9.

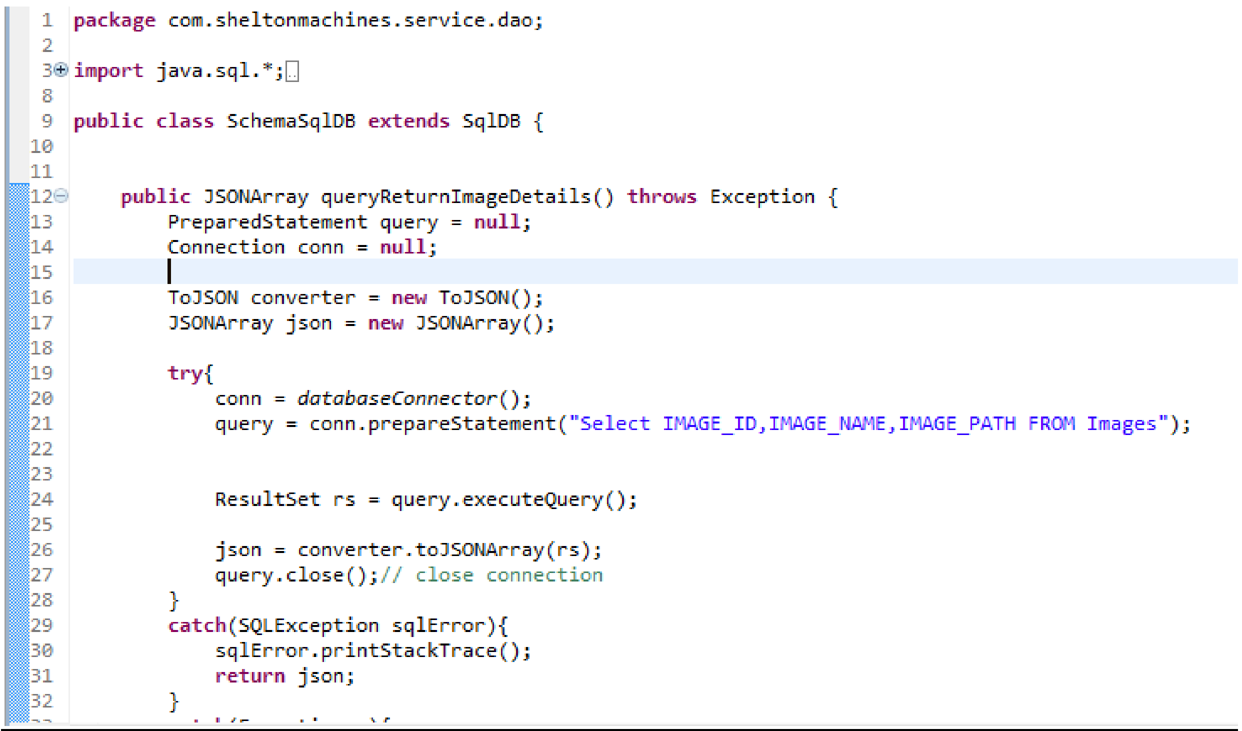
The service is then published to the WebLogic server. The service can be tested by entering the endpoint URL in the browser (for GET methods) or by using a client application. It can also be tested by using REST console apps available for each browser.

The resultset of the database query is converted to JSON object using a separate class. The code snippet for the same is shown in figure 10.



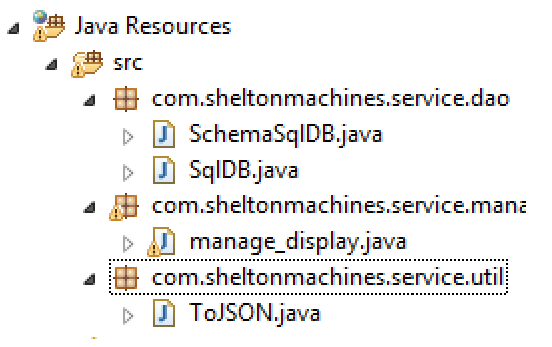
```
package com.sheltonmachines.service.dao;

import java.sql.*;

public class SqlDB {

        private static DataSource SqlDataSource = null;
        private static Context context = null;

       public static DataSource SqlDataSourceConn() throws Exception {

           if(SqlDataSource != null){
               return SqlDataSource;
           }

           try{
               if(context == null){
                   context = new InitialContext();
               }

               SqlDataSource = (DataSource) context.lookup("DataSourceSql");
           }
           catch (Exception e){
               e.printStackTrace();
           }


           return SqlDataSource;
       }
```

Figure 7: Code snippet of the Java method to retrieve the DataSource created in WebLogic



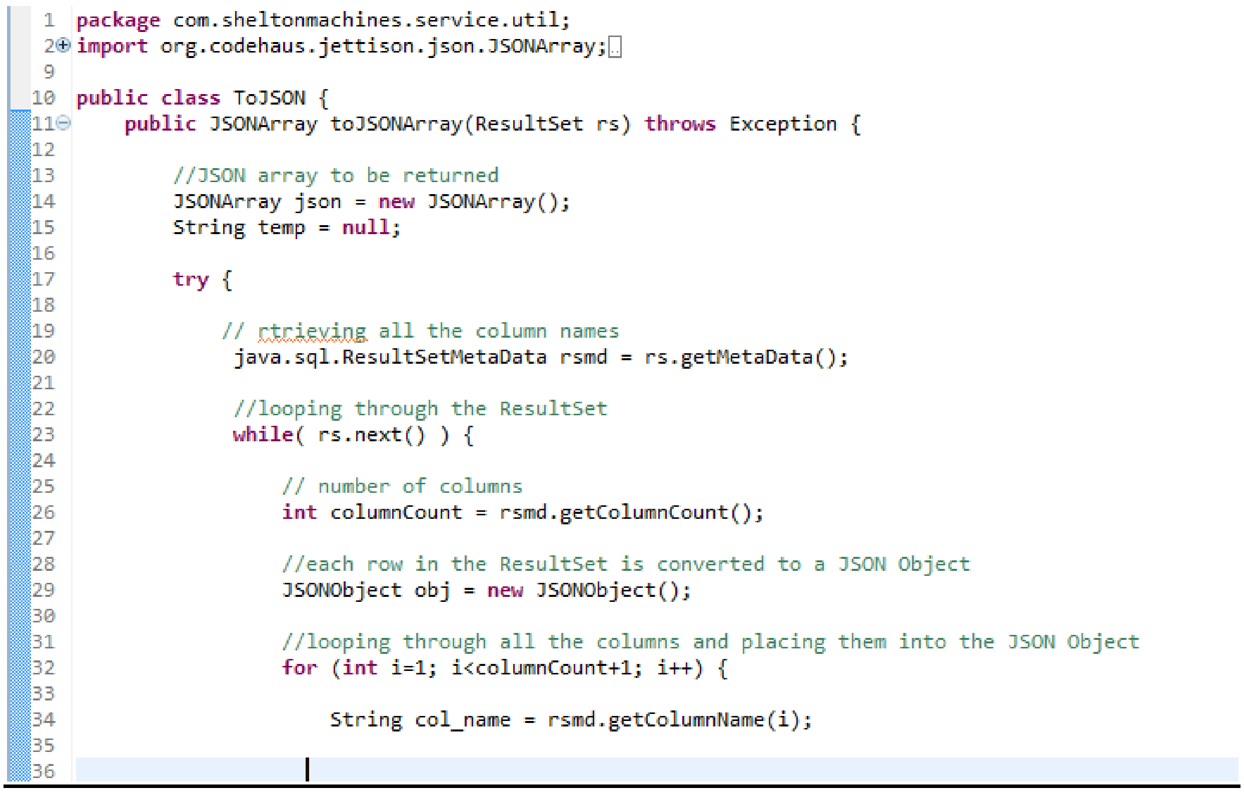
```
package com.sheltonmachines.service.dao;

import java.sql.*;

public class SchemaSqlDB extends SqlDB {


    public JSONArray queryReturnImageDetails() throws Exception {
        PreparedStatement query = null;
        Connection conn = null;

        ToJSON converter = new ToJSON();
        JSONArray json = new JSONArray();

        try{
            conn = databaseConnector();
            query = conn.prepareStatement("Select IMAGE_ID,IMAGE_NAME,IMAGE_PATH FROM Images");


            ResultSet rs = query.executeQuery();

            json = converter.toJSONArray(rs);
            query.close();// close connection
        }
        catch(SQLException sqlError){
            sqlError.printStackTrace();
            return json;
        }
```

Figure 8: Code snippet of the method which performs the SQL query to the database



```
Java Resources
  src
    com.sheltonmachines.service.dao
      SchemaSqlDB.java
      SqlDB.java
    com.sheltonmachines.service.mana
      manage_display.java
    com.sheltonmachines.service.util
      ToJSON.java
```

Figure 9: REST Service Package



```java
package com.sheltonmachines.service.util;
import org.codehaus.jettison.json.JSONArray;

public class ToJSON {
    public JSONArray toJSONArray(ResultSet rs) throws Exception {

        //JSON array to be returned
        JSONArray json = new JSONArray();
        String temp = null;

        try {

            // rtrieving all the column names
             java.sql.ResultSetMetaData rsmd = rs.getMetaData();

             //looping through the ResultSet
             while( rs.next() ) {

                 // number of columns
                 int columnCount = rsmd.getColumnCount();

                 //each row in the ResultSet is converted to a JSON Object
                 JSONObject obj = new JSONObject();

                 //looping through all the columns and placing them into the JSON Object
                 for (int i=1; i<columnCount+1; i++) {

                     String col_name = rsmd.getColumnName(i);

```

Figure 10: Code snippet for converting the database ResultSet to JSON Object

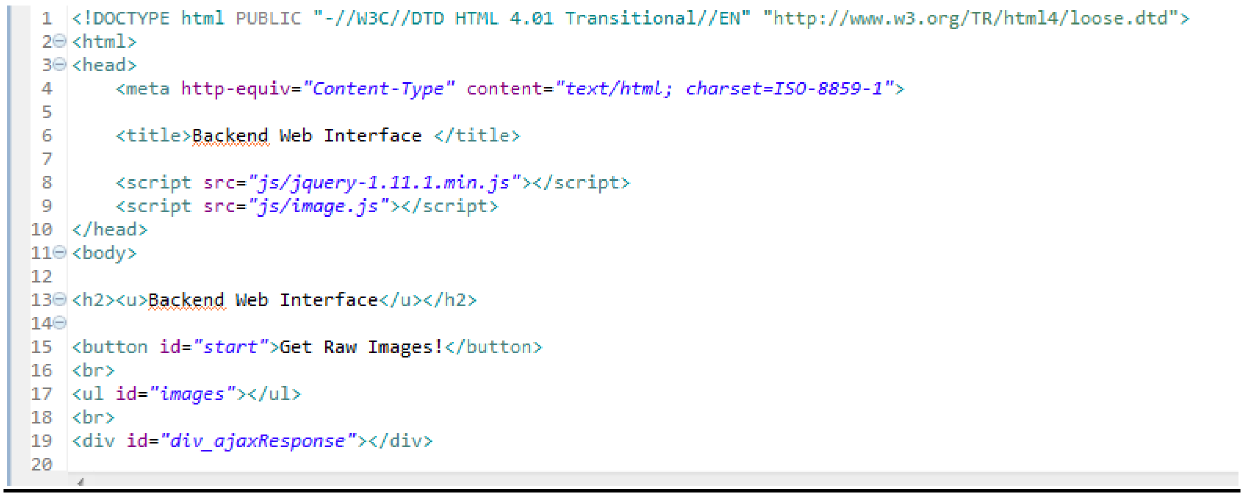
Chapter 5

5. Implementation of client using JavaScript, jQuery and Ajax

The client is written in Javascript, jQuery and Ajax and the client access the service on the WebLogic server.

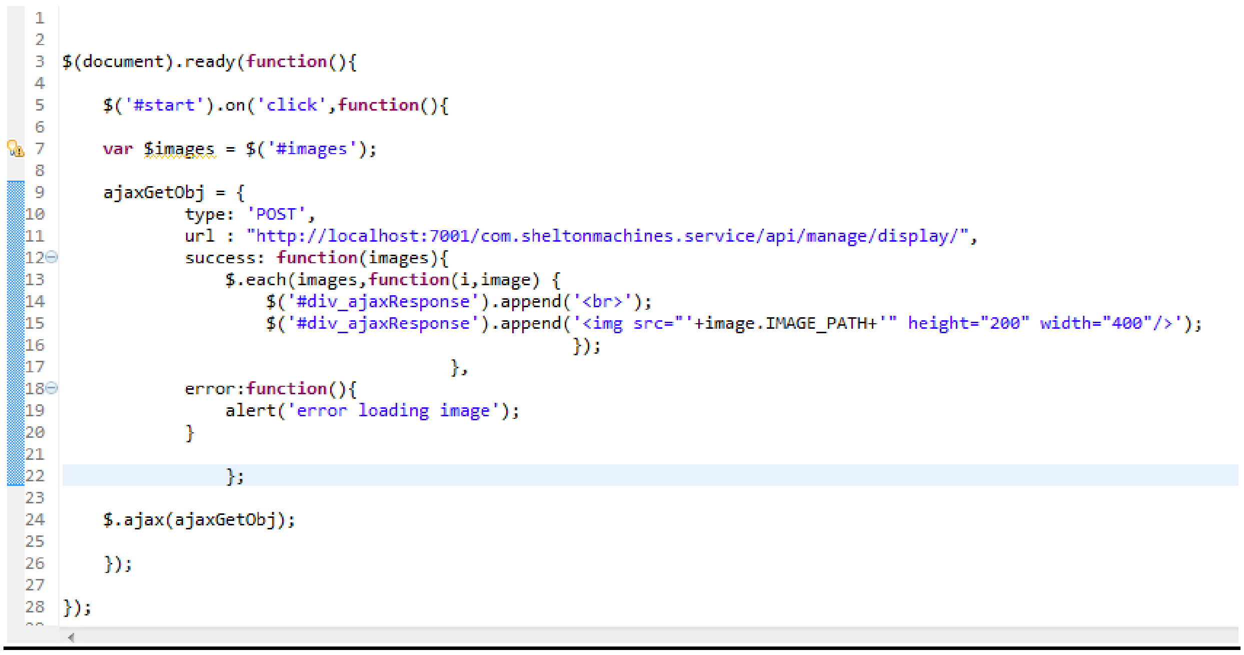
Objective i) Live streaming of images at client side. Images are of size 4MB stored at a location both client and server can access. Image paths are stored in the database.

The service implemented in chapter 4 returns the image details as a JSON Object. Here the client makes Ajax calls using JQuery library. The code snippet is shown in figure 12. Specify the REST service URL in the Ajax call and the request is send to the server. The server then responds with the image details. On success of the Ajax call, the images are displayed in the browser.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

    <title>Backend Web Interface </title>

    <script src="js/jquery-1.11.1.min.js"></script>
    <script src="js/image.js"></script>
</head>
<body>

<h2><u>Backend Web Interface</u></h2>

<button id="start">Get Raw Images!</button>
<br>
<ul id="images"></ul>
<br>
<div id="div_ajaxResponse"></div>

```

Figure 11: Code snippet of the html page for the client

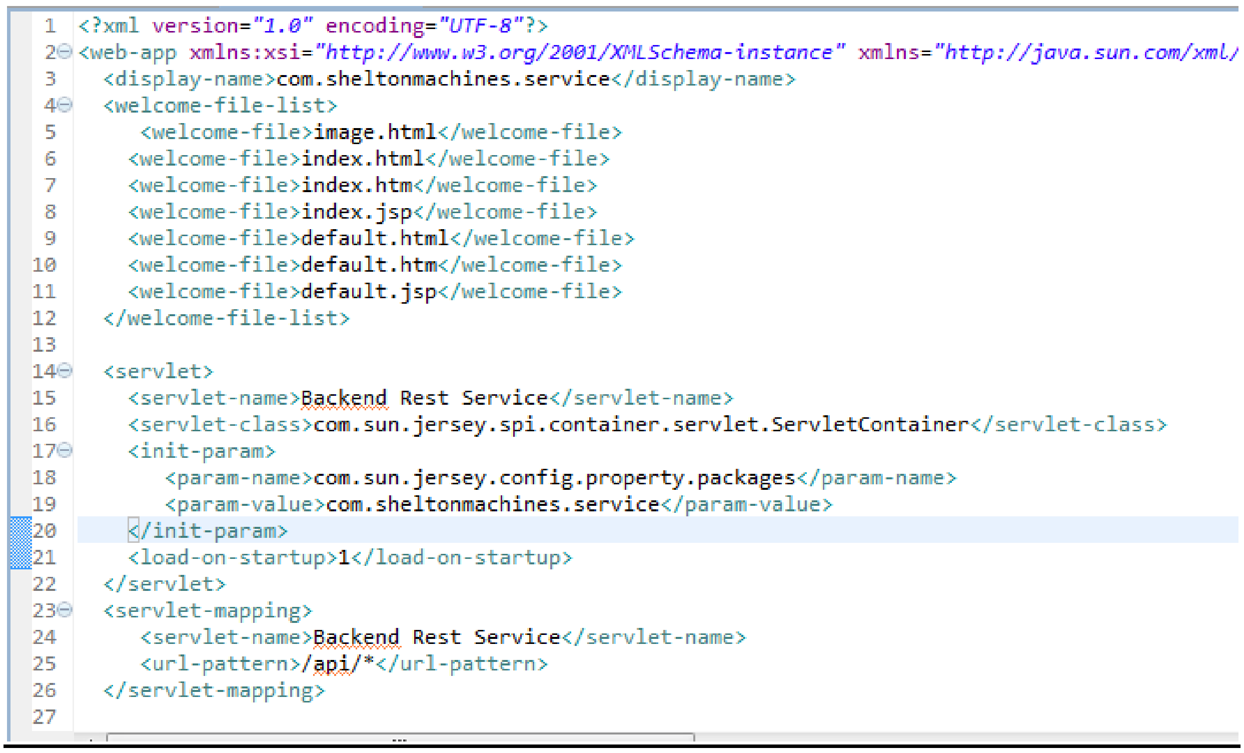


```


$(document).ready(function(){

    $('#start').on('click',function(){

    var $images = $('#images');

    ajaxGetObj = {
            type: 'POST',
            url : "http://localhost:7001/com.sheltonmachines.service/api/manage/display/",
            success: function(images){
                $.each(images,function(i,image) {
                    $('#div_ajaxResponse').append('<br>');
                    $('#div_ajaxResponse').append('<img src="'+image.IMAGE_PATH+'" height="200" width="400"/>');
                                            });
                                },
            error:function(){
                alert('error loading image');
            }

                };

    $.ajax(ajaxGetObj);

    });

});
```

Figure 12: Code snippet of JQuery + Ajax consuming the REST service

When an empty ‘Dynamic Web Project’ is created in Eclipse IDE for Java EE, a template for web application deployment descriptor (web.xml) gets generated in the WEB-INF directory of the web application project. It provides the configuration and deployment information for the web components of a specific web application (eg: servlets, servlet mappings, URL mapping) .When we hit the client URL , the starting point will be this web.xml. It acts a gateway to the application.

The servlet needs to be configured manually. A servlet can have multiple servlet-mapping. The web.xml for this project is shown in figure 13. When a request comes for a particular URL pattern, the servlet container matches the servlet-name with the url-pattern, finds the corresponding servlet-path and pass the control to the service.



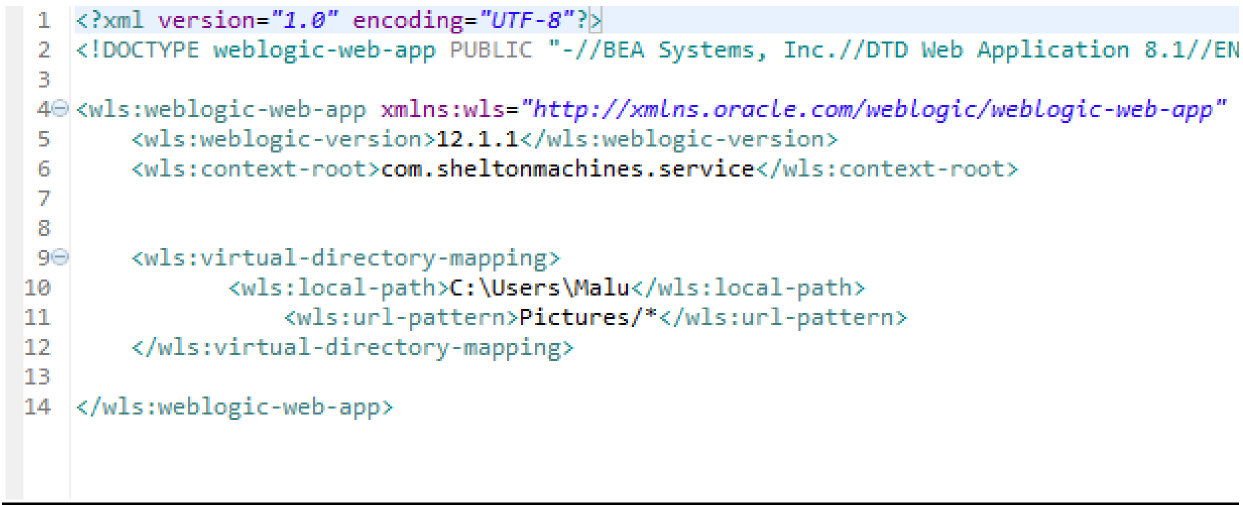
```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/
  <display-name>com.sheltonmachines.service</display-name>
  <welcome-file-list>
     <welcome-file>image.html</welcome-file>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <servlet>
    <servlet-name>Backend Rest Service</servlet-name>
    <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
    <init-param>
       <param-name>com.sun.jersey.config.property.packages</param-name>
       <param-value>com.sheltonmachines.service</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
     <servlet-name>Backend Rest Service</servlet-name>
     <url-pattern>/api/*</url-pattern>
  </servlet-mapping>

```

Figure 13: Code snippet of Web Application deployment descriptor (web.xml)

Web application deployment descriptor elements for all the application in a user domain are defined in weblogic.xml file which also resides in WEB-INF directory of the web application (figure 14). Lines 9 to 11 in figure 14 shows virtual directory mapping.

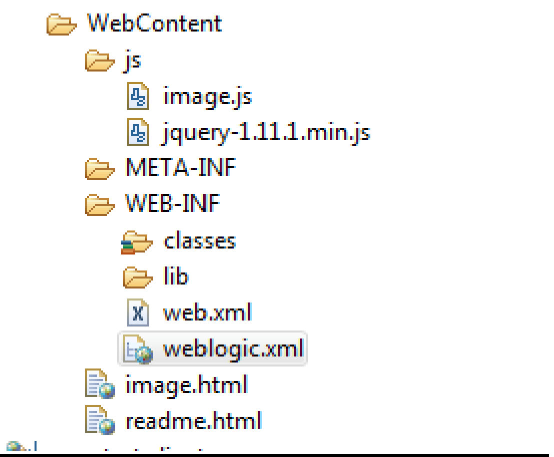
“Use the *virtual-directory-mapping* element to specify document roots other than the default document root of the Web application for certain kinds of requests, such as image requests” [13]. It can be used to include images from another location other than the physical directory that is mapped to application’s root virtual directory. If we specify a directory name, it gets mapped to a physical directory on a local or remote server.

The Client package structure is shown in figure 15. All JavaScript files are in a spate folder.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE weblogic-web-app PUBLIC "-//BEA Systems, Inc.//DTD Web Application 8.1//EN

<wls:weblogic-web-app xmlns:wls="http://xmlns.oracle.com/weblogic/weblogic-web-app"
    <wls:weblogic-version>12.1.1</wls:weblogic-version>
    <wls:context-root>com.sheltonmachines.service</wls:context-root>


    <wls:virtual-directory-mapping>
            <wls:local-path>C:\Users\Malu</wls:local-path>
                <wls:url-pattern>Pictures/*</wls:url-pattern>
    </wls:virtual-directory-mapping>

</wls:weblogic-web-app>
```

Figure 14: code snippet of Web application deployment descriptor elements (weblogic.xml)



```
WebContent
    js
        image.js
        jquery-1.11.1.min.js
    META-INF
    WEB-INF
        classes
        lib
        web.xml
        weblogic.xml
    image.html
    readme.html
```

Figure 15: Client Package

Please find the client page and the corresponding result in figure 16 and 17 respectively. Right now, all the images for the image paths in database are displayed in the browser on a button click. But it should be changed as explained below to achieve the first objective.

1) Client will send a ‘get raw images’ instruction to the server .

2) The server will then send an image file path to the client which then displays on its browser. This image path is of the first image in a list of images.

3) The client sends an acknowledgement it has displayed the image. It may have a small delay before sending the acknowledgement to not have the images change too fast.

4) The server would send the next file path (back to step 2). This would result in a 'live' changing of images and would look similar to streaming.

5) The client will eventually want to stop the image changing and click a button 'stop'. The sequence would also stop if the server came to the end of its image list.

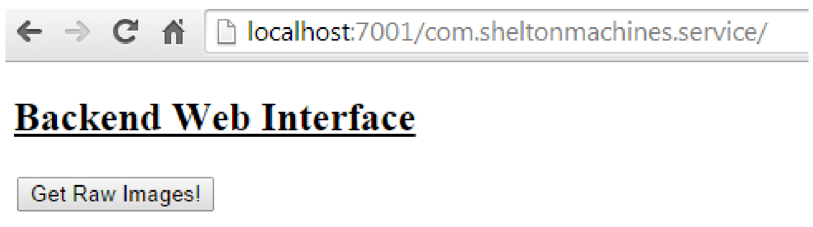


Figure 16: The client home page

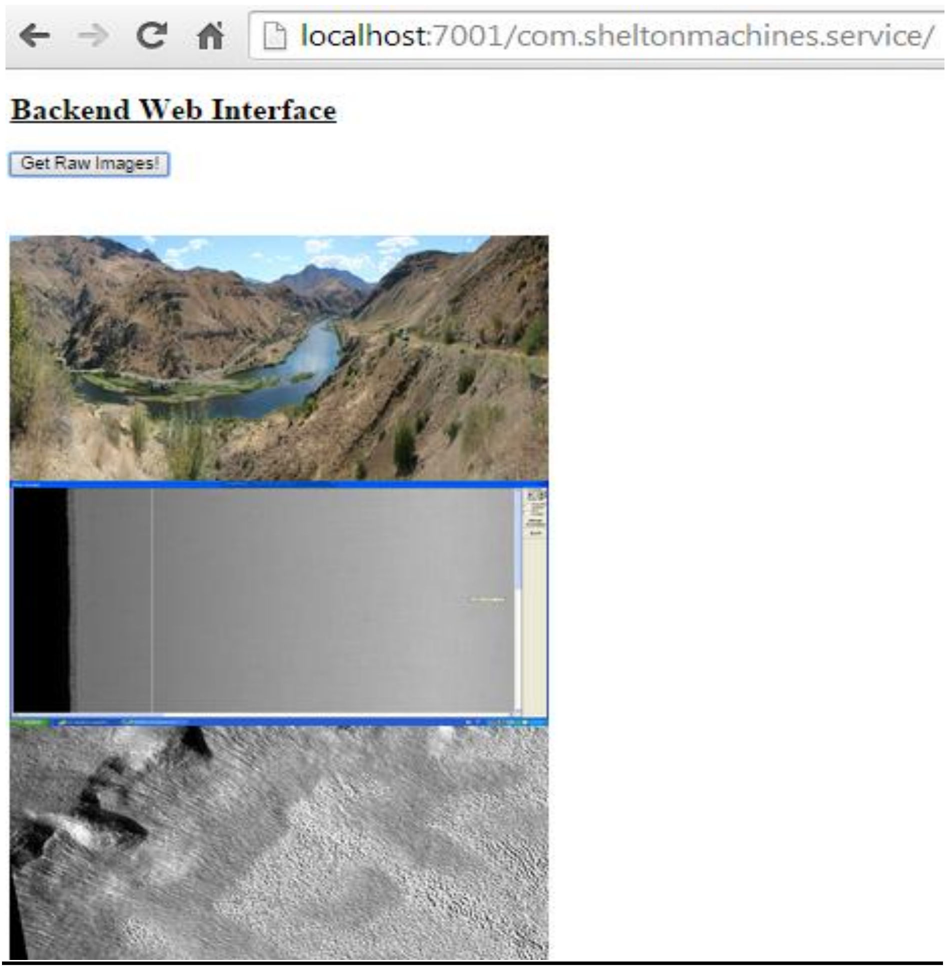


Figure 17: Client page after processing the request

6. Next Steps

- Implementation of the remaining functionalities using jQuery, Javascript and Ajax.
- Implementation of client using PHP and AJAX
- Implementation of client using ASP.NET and jQuery
- Evaluation of results to check the performance, to do a comparison of all the above implementations in order to decide on the best technology that suits the project
- Overall impact of the project

References:

[1] "JQuery UI 1.8. The User Interface Library for JQuery",by Dan Wellman

[2]"Apache CXF Web Service Development-Develop and deploy SOAP and RESTful web services", by Naveen Balani and Rajeev Hathi,PACKT PUBLISHING, BIRMINGHAM-MUMBAI

[3]"Foundations for Efficient Web service Selection"by Qi Yu, Athman Bouguettaya, SPRINGER US, 2010

[4]"Web Services & SOA Principles and Technology", by Michael.P.Papazoglou, SECOND EDITION,PEARSON

[5] Part 1: Introduction to Jersey—a Standard, Open Source REST Implementation [online], Accessed date [27/09/2014]

Available at: http://www.oracle.com/technetwork/articles/javaee/jersey-part1-159891.html

[6] "RESTful Web Services with Jersey" by Scott Leberknight , Software Developer at Near Infinity Corporation, Jun 07, 2014

[7] RESTful Web Services [online], [Accessed date: [27/09/2014]

Available at: http://www.oracle.com/technetwork/articles/javase/index-137171.html

[8] RESTful Web services: The basics [online], [Accessed date: 14/07/2014]

Abailable at: http://www.ibm.com/developerworks/webservices/library/ws-restful/

[9] Web Services based on SOAP and REST Principles [online], [Accessed date: 25/09/2014]

Available at : http://www.ijsrp.org/research-paper-0513/ijsrp-p17115.pdf

[10] “Introduction to Client Server Computing” by Yadav, Subash Chandra, Singh, Sanjay Kumar, New Age International, 2009

[11] Hatem Hamad, Motaz Saad, and Ramzi Abed “Performance Evaluation of RESTful Web Services for Mobile Devices” Computer Engineering Department, Islamic University of Gaza, Palestine International Arab Journal of e-Technology, Vol. 1, No. 3, January 2010.

[12] **Oracle WebLogic Server** [online], [Accessed date: [27/09/2014]
Available at:
http://www.oracle.com/technetwork/middleware/weblogic/overview/index.html

[13] **virtual-directory-mapping** [online], [Accessed date: [29/09/2014]
Available at:
http://docs.oracle.com/cd/E11035_01/wls100/webapp/weblogic_xml.html#wp1039396

[14] “ Programming PHP”, 3rd edition, By Kevin Tatroe, Peter MacIntyre, RasmusLerdorf (Page: 1)

[15] “Learning JQuery”, By Jonathan Chaffer, Karl Swedberg

[16] Category: Ajax [online], [Accessed date: [29/09/2014]

Available at: http://api.jquery.com/category/ajax/

[17]” jQuery:Advantages and Disadvantages” [online], [Accessed date: [29/09/2014]

Available at: http://www.jscripters.com/jquery-disadvantages-and-advantages/

[18] “Get Started with ASP.NET” [online], [Accessed date: [29/09/2014]

Available at: http://www.asp.net/get-started