ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

A SOA Approach to Build Remote Lab Services

T.V.Lakshmi Sukrutha

Assistant Professor, Dept. of Computer Science Engineering, Methodist College of Engineering and Technology,

Hyderabad, Telangana, India

ABSTRACT: Basically, a lab accommodates few students when the number of systems is limited. Each student will be allocated a particular system and assign to complete the experiment in a given time. In such situation a kind of asynchronous activity will be taking place in the lab. Considering this time constraint and limited resources this project mainly concentrates on building an application where huge number of students can perform their experiments at any time and use long hours which are convenient to them. In order to meet the above requirements, it is necessary that the designed application be web based which increases the number of lab users. The proposed project uses Service Oriented Architecture (SOA) rather than Client Server Architecture as this Architecture uses multiple servers for user interface, data validation and database which involve complex programming. SOA applications are easy to develop and their implementation depends on services and is essentially a collection of services. Each service implements one action. Services include unassociated, loosely coupled units of functionality that communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed. This application includes mainly two services, one is online booking and the other is remote accessing. Students can book *date* and *time* according to their convenience and remotely access lab by logging in. If a student blocks *date* and *time* for a particular usage, another student who blocks the same *time* and *date* is intimated that it is booked.

KEYWORDS: Service Oriented Architecture, Web services, Apache Axis.

I. INTRODUCTION

The improvements in communication network and the emergence of Internet have made revolutionary changes in the dissemination of information and the methods of learning. The Remote lab method is one of those practices being used by many through Internet facility. This facility has given an opportunity to practice to those people who cannot attend the traditional lab, due to some problems such as geographical distance i.e. living far away from the educational institutions, financial constraints, scheduling problems and the like. Within the disciplines of Engineering and Physical Sciences, lab work is considered to be the heart of learning and can have a strong impact on students' learning outcomes. Labs based sessions are widely used in order to provide physical evidence of theoretical principles and to teach practical skills. When used appropriately they can enthuse, motivate and inspire students [5]. In many cases, the theoretical knowledge is not enough and hence a practical knowledge is necessary. Remote labs represent an attractive solution to do experiments through online mode. Remote labs have been developed for the students to obtain this practical knowledge without attending the real lab or, in some cases, to reduce the time devoted to do real practices. Most of the remote labs focus on engineering lab as the engineering discipline contains the biggest portion of laboratory studies. In other words, engineering is an applied science. Remote labs are becoming widely accepted in universities and in the Institutions of higher learning for providing distance mode of education and for augmenting traditional laboratories.

II. RELATED WORK

The Remote lab project is implemented in three parts: The first part is the Student Front End (User Interface), second one is Back End Using Oracle and the third is connecting to remote computer using SOA. The user interface is organized in such a way that the students Register with their full name, username, password and student id and enters in

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

to next level of Online Booking. By using booking table, the student can reserve in advance the time when they want to execute remote experiments and can Login only in that reserved slot. At that particular time nobody else can access the remote experiment. The Student needs to give Remote Ip address as Input to access lab desktop.
Accessing the lab desktop also requires web browser and an Internet connection. After logging in to the remote desktop the student can practice Software programs. Each user interface contains a well defined set of actions, i.e. to Register, to Block Date and Time and to Login. The Backend Database is implemented using Oracle SQL commands and communicates with these actions. It Contains a Single Table tab which in turn consists of three other tables Sid , User Details and Accounts . The First Table Sid stores student ids. The Purpose of this table is to allow the stored ids to have permission to register and block strange ids. The security level is thus increased by this and no other third person can have access. The Second Table User Details is used to store the values inserted in to the Register User Interface. It stores and updates the values like Full name, Username and Password details. Third table Accounts stores Blocked date, time and the random id given after registering. Another important feature of this remote lab is the management of concurrent access to the same date by multiple users. The scheduling mechanism is strengthened by a proper locking system that tracks the dates that are in use. When a student blocks date the scheduler checks if the dates are available. If available the date and time along with the random id are given to the student and saved and updated in the database Accounts. If not, it shows error.
The third part is the core of this project, connecting to Remote desktop using SOA. This part is implemented using Web Services. The Axis server plays an important role in communicating between the client/student and the remote desktop. This server is first installed in remote desktop computer. After installing the server a web service program is written and deployed in to the axis server. The server must be started with the help of setting class path. This is explained in detail under AXIS section. Now the student/client has to connect through login User Interface by giving username, password, rid (random id), remote ip address and current Ip address. Then the connection will be established.

III. SERVICE ORIENTED ARCHITECTURE

Service-Oriented Architecture (SOA) is an evolution of distributed computing based on the request/reply design model for synchronous and asynchronous applications. An application's business logic or individual functions are modularized and presented as services for consumer/client applications. What's key to these services is their loosely coupled nature; i.e., the service interface is independent of the implementation [3]. Application developers or system integrators can build applications by composing one or more services without knowing the services' underlying implementations. For example, a service can be implemented either in .Net or J2EE, and the application consuming the service can be on a different platform or language. Service-oriented architectures have the following key characteristics:
1. SOA services have self-describing interfaces in platform-independent XML documents. Web Services Description Language (WSDL) is the standard used to describe the services.
2. SOA services communicate with messages formally defined via XML Schema (also called XSD). Communication among consumers and providers or services typically happens in mixed environments, with little or no knowledge about the provider. Messages between services can be viewed as key business documents processed in an enterprise.
3. SOA services are maintained in the enterprise by a registry that acts as a directory listing. Applications can look up the services in the registry and invoke the service. Universal Description, Definition, and Integration (UDDI) is the standard used for service registry.
4. Each SOA service has a quality of service (QoS) associated with it. Some of the key QoS elements are security requirements, such as authentication and authorization, reliable messaging, and policies regarding who can invoke services.
The reality in IT enterprises is that infrastructure is heterogeneous across operating systems, applications, system software, and application infrastructure. Some existing applications are used to run current business processes, so starting from scratch to build new infrastructure isn't an option. Enterprises should quickly respond to business changes with quickness; leverage existing investments in applications and application infrastructure to address newer business requirements; support new channels of interactions with customers, partners, and suppliers; and feature an architecture that supports organic business [3]. SOA with its loosely coupled nature allows enterprises to plug in new services or upgrade existing services in a granular fashion to address the new business requirements, provides the option to make the services consumable across different channels, and exposes the existing enterprise and legacy applications as services, thereby safeguarding existing IT infrastructure investments.

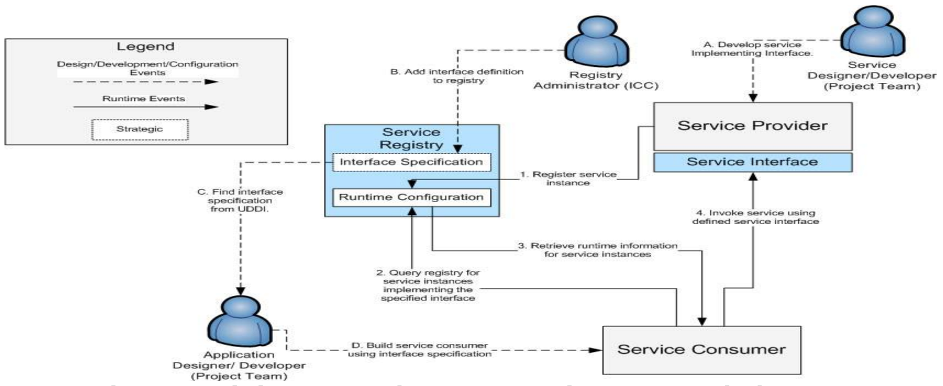

Legend
Design/Development/Configuration Events
Runtime Events
Strategic
B. Add interface definition to registry
Registry Administrator (ICC)
A. Develop service Implementing Interface.
Service Designer/Developer (Project Team)
Service Registry
Interface Specification
Runtime Configuration
Service Provider
Service Interface
1. Register service instance
C. Find interface specification from UDDI.
4. Invoke service using defined service interface
3. Retrieve runtime information for service instances
2. Query registry for service instances implementing the specified interface
Application Designer/ Developer (Project Team)
D. Build service consumer using interface specification
Service Consumer

Figure 3.1 shows Detailed Service Architecture to implement SOA which enterprises need.

In Figure 3.2, several service consumers can invoke services by sending messages. These messages are typically transformed and routed by a service bus to an appropriate service implementation. This service architecture can provide a business rules engine that allows business rules to be incorporated in a service or across services. The service architecture also provides a service management infrastructure that manages services and activities like auditing, billing, and logging [6]. In addition, the architecture offers enterprises the flexibility of having agile business processes, better addresses the regulatory, and changes individual services without affecting other services.

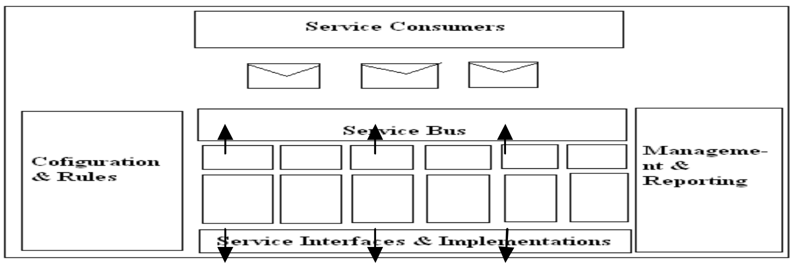


Figure 3.2 Service Architecture.

To run and manage SOA applications, enterprises need an SOA infrastructure that is part of the SOA platform. An SOA infrastructure must support all the relevant standards and required runtime containers. A typical SOA infrastructure looks like Figure 3.3. There is a general confusion about the relationship between SOA and Web services. Fundamentally, SOA is an architectural pattern, while Web services are services implemented using a set of standards; Web services is one of the ways you can implement SOA. The benefit of implementing SOA with Web services is that you achieve a platform-neutral approach to accessing services and better interoperability as more and more vendors support more and more Web services specifications [1].

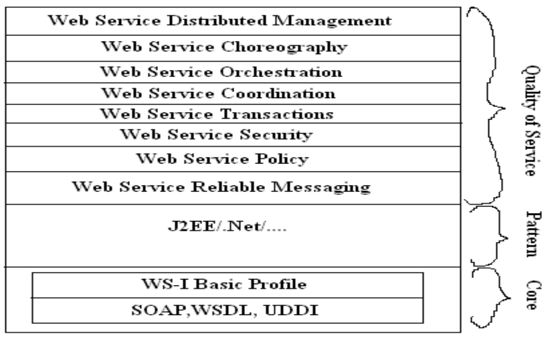


Figure 3.3. SOA Infrastructure.

A Service is a unit of work done by a service provider to achieve desired end results for a service consumer. It represents a publicized package of functionality. Both service provider and service consumer are the roles played by software agents on behalf of their owner. A service is Compose able and Discoverable
The actual use of service is often based upon an agreed-upon contract with the provider, including in detail what is provided and what is the quality(availability, cost etc) of the service. SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents.SOA is a specific architectural style that is concerned with loose coupling and dynamic binding between services. On the other hand, Web Services is an approach to realizing a SOA. A software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process able format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with XML serialization in conjunction with other Web-related standards [4]. Web Services is the base for implementing SOA and the Service bus is the centerpiece of this implementation. So Figure 3.4 shows the architecture of the Service Bus.

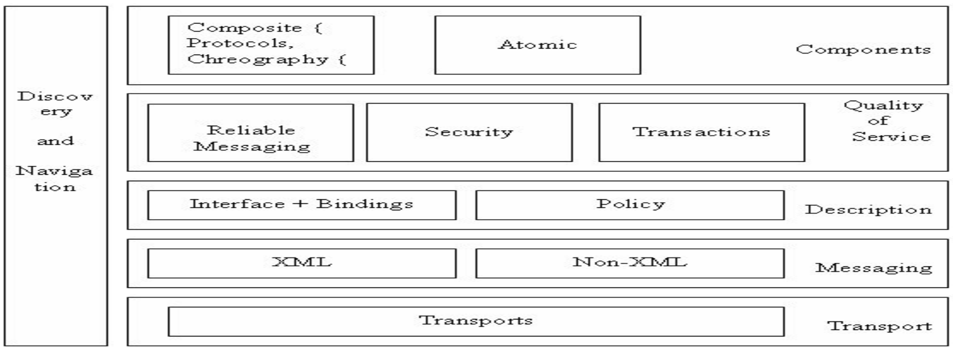


Figure 3.4 Architecture of Service Bus.

IV. WEB SERVICES

The Figure 4.1 shows Web Services Architecture. The bottom (transport) layer presents its capabilities to cope with various transport protocols to communicate between a service and a requester. In case of web services you can transport messages by using the ubiquitous Web protocols such as Hypertext Transport Protocol (HTTP) or Secure HTTP (HTTPS) to give the widest possible coverage in terms of support for the protocols, you can also transport them over any communications protocol.

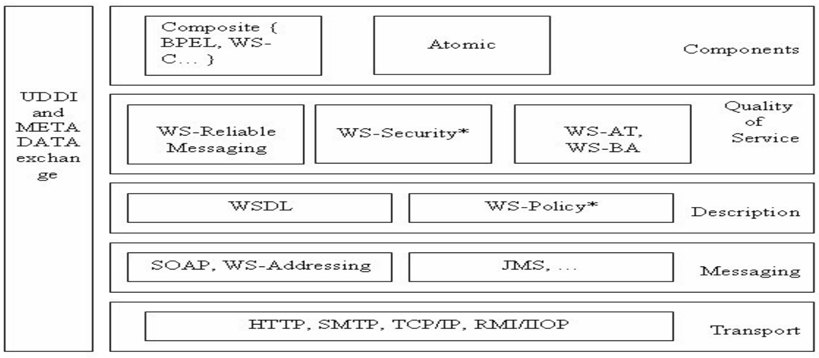


Figure 4.1 Web Services Architecture

The messaging layer on top of the transport layer enables the bus to deal with messages, both XML and non-XML. The latter is important, for example, if a requester and the service needed reside in the same J2EE application server and the bus can simply transport the data in its Java rendering, avoiding unnecessary transformations. The messaging services

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

component of the framework contains the most fundamental Web services specifications and technologies, including Extensible Markup Language (XML), SOAP, and WS-Addressing. Collectively, these specifications form the basis of interoperable messaging between Web services. XML provides the interoperable format to describe message content between Web services and is the basic language in which the Web services specifications are defined.
The next (description) layer of the bus facilitates and deals with the description of services in terms of functions supported, quality of services of these functions, and supported binding mechanisms. This metadata is important, and it is fundamental to achieving the loose coupling that is associated with an SOA viz. WSDL, WS-Policy
The actual quality of services that the bus enforces based on appropriate parameterization via polices resides in the layer that follows. The specific issues involving this layer include security, reliability of message delivery, and support for transactions. Likewise web Services provides WS-Reliable Messaging, WS-Security.
The top layer represents the various kinds of virtual components that Web services represent. This layer has atomic services that are not composed as far as a requester's experience with the service is concerned. Composite services that the service bus inherently supports are choreographies and societies of services that cooperate following an agreement protocol to decide on the success of the cooperation at the end of the cooperation.
Finally, another layer provides features for discovery of services and their descriptions and to agree on a mode of interaction between a requester and a service. UDDI and META-DATA Exchange are the ways it is done using web services.

V. IMPLEMENTING REMOTE LAB

The structure of the proposed remote lab access is shown in Figure 5.1. The system simply represents the remote lab with single structure of a computer to interact with the student through the internet implemented with Service Oriented Architecture. It consists of two sections, the student domain and the administrator domain. The student domain enables the client in his place outside the lab to book date and time. After booking, the student is allowed to login on that particular date to the remote lab to perform experiments. The second section, the administrator, adds and deletes students, Reduces overlapping of booking dates, for setting up and defining date and time slots. There are three Modules in this application. Student Front End, Back End Using oracle, Connecting to Remote Computer using SOA and AXIS.

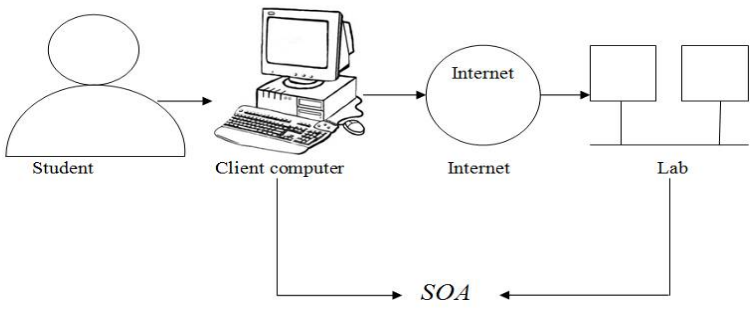


Figure 5.1 Remote Lab Structure

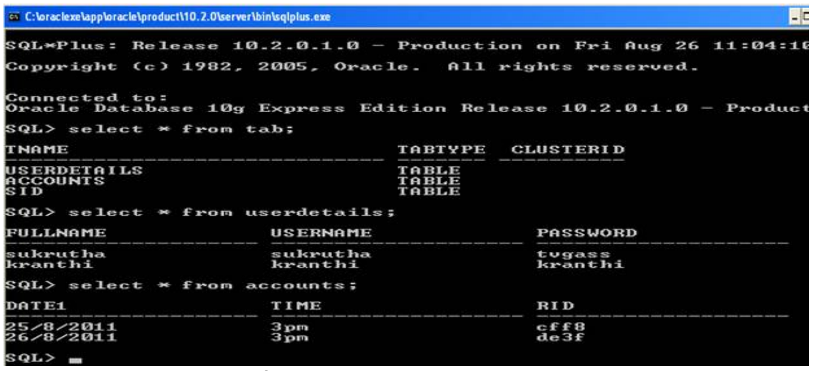
V.I STUDENT FRONT END

Front-end development is a relatively difficult to understand. Historically, this role has been known under several names, html, web designer, coder, front ender and so on, but its core function remains the same. It is a center part that requires both visual sensitivity and programmatic severity.A "front-end" application is one that users interact with directly. Here in this remote lab application the front end user is the Student. The technologies that have been used to develop front end are:

HTML (Hypertext Markup Language): HTML, the structure of the page is the foundation of remote lab application, very important to place the document with the right hooks for the classes and the ids that will provide the style and the interaction that the reader will ultimately use.HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts in languages such as JavaScript which affect the behavior of HTML WebPages.
Cascading Style Sheets: Css, cascading style sheets, a core functionality of front-end development, the styles that lay out the page and give it both its unique visual style and a clear, user-friendly view to allow students, who don't have technical knowledge, some help to read the contents quickly. Design means both how something looks and how something is structured, and in a good design, both come together. An important aspect of styling is checking across several browsers and to write short code that is specific yet generic at the same time and displays well in as many renderers as possible.
Cross-Browser : The browser on your computer is to remain the most advanced and feature-rich client application to access the web for a long time Since the browser wars between Netscape and Internet Explorer on PCs, much has happened. Nowadays, browsers compete with each other for page-rendering speed, plug-ins and add-ons to achieve both a lean and comprehensive browser experience.
Programming: The programming for front end in remote lab uses Servlets and JavaScript. A servlet is a Java programming language class used to extend the capabilities of servers that host applications access via a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by Web servers. JavaScript has fully grown up from inline commands embedded in html to full-blown asynchronous applications executed on the fly on the browser as unobtrusive rich functionality. The widespread usage of java script libraries has produced a excess, of visual effects that turn web pages into a more three-dimensional immersive experience.

V.II BACK END USING ORACLE

A back-end application or program serves indirectly in support of the front-end services, usually by being closer to the required resource or having the capability to communicate with the required resource. The back-end application interacts directly with the front-end. This remote lab application uses Oracle as its Back end. The bridge between the front end and back end database is done using drivers. This is referred to as JDBC-ODBC connection. Here, Backend database connection uses Type IV driver (Oracle thin driver).
Install oracle. Run SQL command line in Oracle. Connect with Username and Password. In this, Database has a single table Tab which in turn consists of three other tables **User details table**, whenever a student registers with full name, username and password values they are automatically inserted in to database. **Accounts table**, Whenever a student books date and time they are inserted in to database and a random id (rid) is given after booking. **Sid table** , only these students are allowed to login in to remote lab. Other than these students id's the system gives error. Figures 5.1, 5.2 shows snapshot of Database Tables.

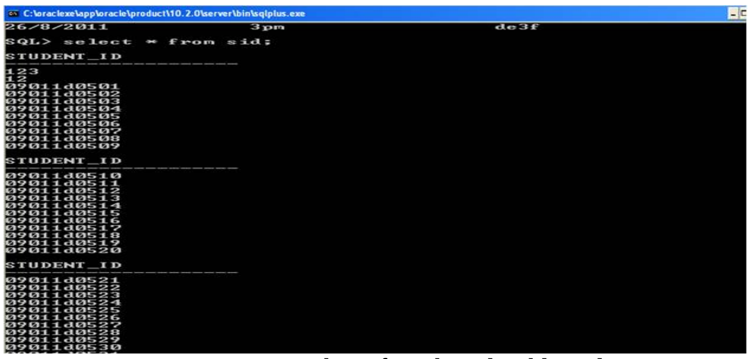

```
C:\oraclexe\app\oracle\product\10.2.0\server\bin\sqlplus.exe

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Aug 26 11:04:1
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to:
Oracle Database 10g Express Edition Release 10.2.0.1.0 - Product
SQL> select * from tab;
TNAME                              TABTYPE  CLUSTERID
------------------------------ ------- ----------
USERDETAILS                        TABLE
ACCOUNTS                           TABLE
SID                                TABLE
SQL> select * from userdetails;
FULLNAME                 USERNAME                 PASSWORD
--------------------     --------------------     --------------------
sukrutha                 sukrutha                 tvgass
kranthi                  kranthi                  kranthi
SQL> select * from accounts;
DATE1                    TIME                     RID
--------------------     --------------------     --------------------
25/8/2011                3pm                      cff8
26/8/2011                3pm                      de3f
SQL> _
```
Figure 5.1 Snapshot of Back end Tables tab, UserDetails and Accounts

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016



```
C:\oraclexe\app\oracle\product\10.2.0\server\bin\sqlplus.exe
26/8/2011                    3pm                   de3f
SQL> select * from sid;
STUDENT_ID
--------------------
123
12
09011d0501
09011d0502
09011d0503
09011d0504
09011d0505
09011d0506
09011d0507
09011d0508
09011d0509
STUDENT_ID
--------------------
09011d0510
09011d0511
09011d0512
09011d0513
09011d0514
09011d0515
09011d0516
09011d0517
09011d0518
09011d0519
09011d0520
STUDENT_ID
--------------------
09011d0521
09011d0522
09011d0523
09011d0524
09011d0525
09011d0526
09011d0527
09011d0528
09011d0529
09011d0530
```

Figure 5.2 Snapshot of Back end Table Sid

V.III CONNECTING TO REMOTE COMPUTER USING SOA AND AXIS

The figure 5.3 shows Building Blocks for delivering Service Oriented Architecture implemented with Web Services [4].

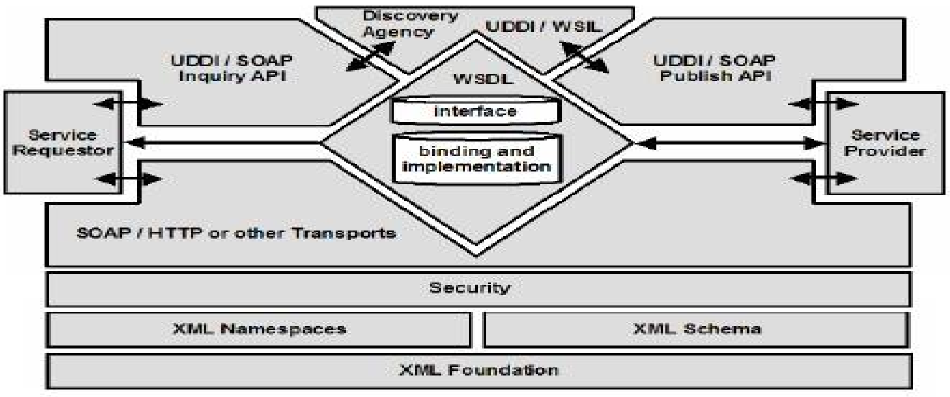


Figure 5.3**.** Building Blocks for delivering Service Oriented Architecture implemented with Web.

XML: It is a Markup language composed of tags and data, Elements and Attributes. Read by an XML processor. Requires Grammar Definition, Valid and well formed.XML Name Spaces gives Global Naming Mechanism for XML, Qualified names: Prefix and local parts, Multiple Name spaces in same document . XML Schema Provides grammar for XML instance docs, Built in types and Simple and Complex custom data types.
SOAP: Simple Object Access Protocol (SOAP) is a light weight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application defined data-types, and a convention for representing remote procedure calls and responses.
WSDL: Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages, containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.
UDDI: Discovery and Integration (UDDI) is a directory service where providers can register and clients search for Web services. Universal Description, Discovery, and Integration (UDDI) provides the definition of a set of services supporting the description and discovery of businesses, organizations, and other Web Services providers, the Web Services they make available, and the technical interfaces which may be used to access those services. The idea is to "discover" organizations and the services that organizations offer, much like using a phone book or dialing information.

JAVA AND WEB SERVICES: Java Web Services uses the find-bind-invoke paradigm as shown in Figure 5.4. In this paradigm, service providers register their service in a public registry. This registry is used by consumers to find services that match certain criteria. If the registry has such a service, it provides the consumer with a contract and an endpoint address for that service.

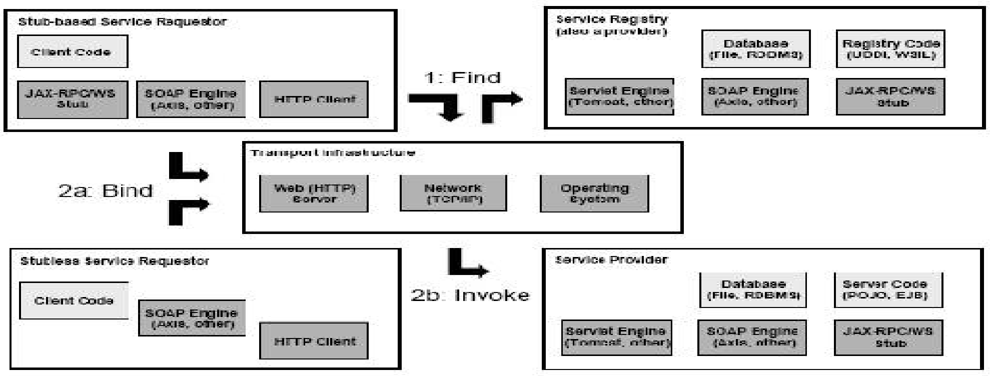
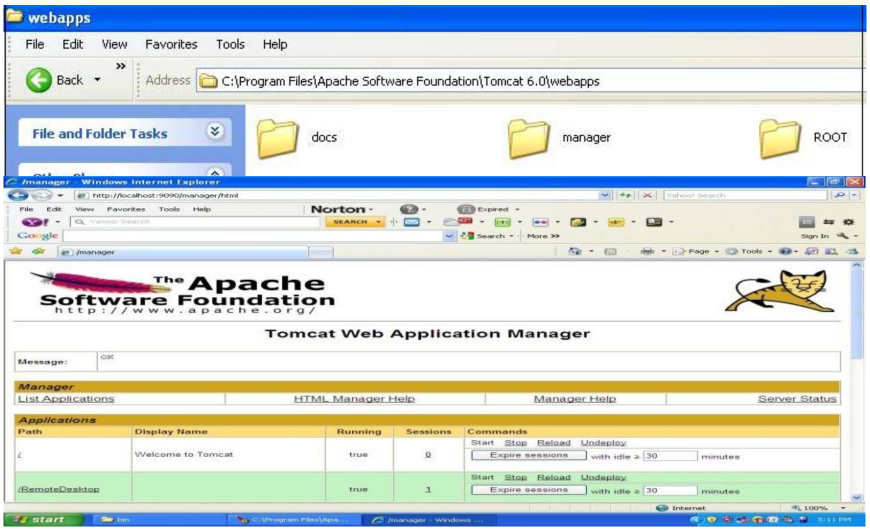

Stub-based Service Requestor
Client Code
JAX-RPC/WS Stub
SOAP Engine (Axis, other)
HTTP Client
1: Find
Service Registry (also a provider)
Database (File, RDBMS)
Registry Code (UDDI, WSIL)
Servlet Engine (Tomcat, other)
SOAP Engine (Axis, other)
JAX-RPC/WS stub
2a: Bind
Transport Infrastructure
Web (HTTP) Server
Network (TCP/IP)
Operating System
Stubless Service Requestor
Client Code
SOAP Engine (Axis, other)
HTTP Client
2b: Invoke
Service Provider
Database (File, RDBMS)
Server Code (POJO, EJB)
Servlet Engine (Tomcat, other)
SOAP Engine (Axis, other)
JAX-RPC/WS stub

Figure 5.4 Java Web services architecture – logical and process view

TOMCAT: Apache Tomcat is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the Java Server Pages (JSP) specifications from Oracle Corporation, and provides a "pure Java" HTTP web server environment for Java code to run.

Remote Lab uses Apache Tomcat as its web server. The Client end application is run and tested in tomcat. The port number used here is 9090. Figures 5.6 show that the Application **Remote Desktop** is copied and pasted in Webapps folder of Apache Tomcat and viewed in Tomcat Web Server.


webapps

Address C:\Program Files\Apache Software Foundation\Tomcat 6.0\webapps

docs manager ROOT

http://localhost:9090/manager/html

The Apache Software Foundation
http://www.apache.org/

Tomcat Web Application Manager

Message: OK

Manager

List Applications | HTML Manager Help | Manager Help | Server Status

Applications

Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	0	Start Stop Reload Undeploy; Expire sessions with idle ≥ 30 minutes
/RemoteDesktop		true	1	Start Stop Reload Undeploy; Expire sessions with idle ≥ 30 minutes

Figure 5.6 Snapshot of RemoteDesktop folder in Apache Tomcat Web Server

AXIS: AXIS stands for Apache Extensible Interaction System. It is an open-source project and implements standard JAX-RPC API of Java. AXIS is SOAP engine which works on server and client. It allows Java classes to be deployed as Web services so that they can be accessed from anywhere.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

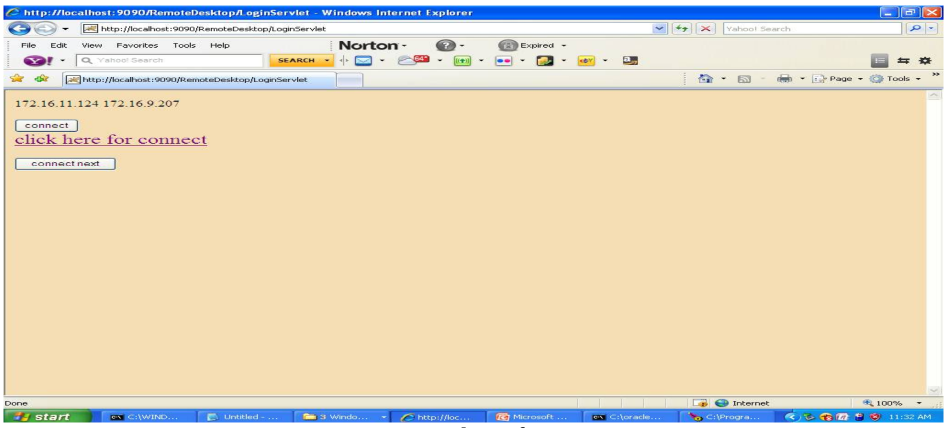
(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

It acts as a simple stand-alone server, A server that plugs into Servlet engines such as Tomcat and gives an extensive support for the Web Services Description Language (WSDL).It is also used as an Emitter tooling that generates Java classes from WSDL and a tool for monitoring TCP/IP packets. The features of axis include:

- **Speed**: Axis uses SAX (event-based) parsing to achieve significantly greater speed than earlier versions of Apache SOAP.
- **Flexibility:** The Axis architecture gives the developer complete freedom to insert extensions into the engine for custom header processing, system management, or anything else you can imagine.
- **Stability**: Axis defines a set of published interfaces which change relatively slowly compared to the rest of Axis.
- **Component-oriented deployment**: You can easily define reusable networks of Handlers to implement common patterns of processing for your applications, or to distribute to partners.
- **Transport framework**: A clean and simple abstraction for designing transports (i.e., senders and listeners for SOAP over various protocols such as SMTP, FTP, message-oriented middleware, etc), and the core of the engine is completely transport-independent.
- **WSDL support**: Axis supports the Web Service Description Language, version 1.1, which allows you to easily build stubs to access remote services, and also to automatically export machine-readable descriptions of your deployed services from Axis.

IMPLEMENTING AND DEPLOYING WEB SERVICES IN AXIS: Install apache axis in any folder. Build a Remote lab Web Service with a normal Java class that provides the Web Service implementation methods and a Web Services Deployment Descriptor (WSDD). Figure 5.8 shows how to start Axis Server. A java client program is written and compiled using ClientInitiator.java in order to connect to the axis server. Writing a Deploy.wsdd (web service deployment descriptor) file . The deployment descriptor provides three key pieces of information. The name of the Web Service, Class that implements the Web Services and the Methods in the class that we want Axis to expose. The class file (ClientInitiator.class) generated after compiling ClientInitiator.java is cut and pasted in to another folder remoteclient. In order to integrate wsdd file in to axis, setting class path and Compiling client program. Start java **org. apache.axis.transport.http**. SimpleAxisServer– p 65535 java org.apache.axis.client.AdminClient-p 65535 deploy.wsd. The web service has been deployed in to axis server and successfully running. The student can login in his/her booked slot by giving the remote ip address and clicking **connect** as shown in figure 5.7.


http://localhost:9090/RemoteDesktop/LoginServlet - Windows Internet Explorer

172.16.11.124 172.16.9.207

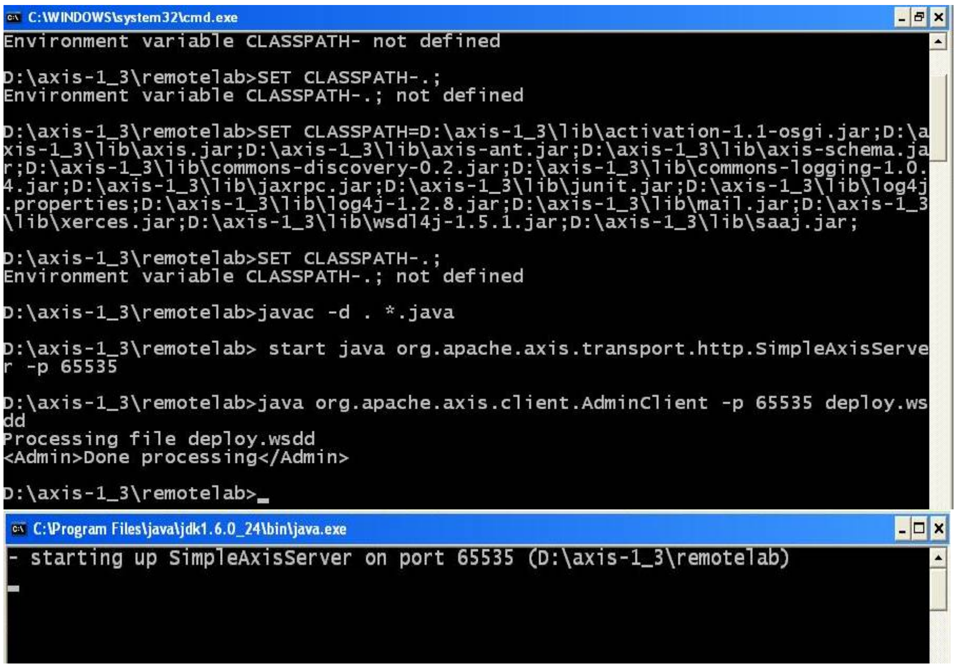
Figure 5.7 Snapshot of Connection

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016



C:\WINDOWS\system32\cmd.exe

```
Environment variable CLASSPATH- not defined

D:\axis-1_3\remotelab>SET CLASSPATH-.;
Environment variable CLASSPATH-.; not defined

D:\axis-1_3\remotelab>SET CLASSPATH=D:\axis-1_3\lib\activation-1.1-osgi.jar;D:\a
xis-1_3\lib\axis.jar;D:\axis-1_3\lib\axis-ant.jar;D:\axis-1_3\lib\axis-schema.ja
r;D:\axis-1_3\lib\commons-discovery-0.2.jar;D:\axis-1_3\lib\commons-logging-1.0.
4.jar;D:\axis-1_3\lib\jaxrpc.jar;D:\axis-1_3\lib\junit.jar;D:\axis-1_3\lib\log4j
.properties;D:\axis-1_3\lib\log4j-1.2.8.jar;D:\axis-1_3\lib\mail.jar;D:\axis-1_3
\lib\xerces.jar;D:\axis-1_3\lib\wsdl4j-1.5.1.jar;D:\axis-1_3\lib\saaj.jar;

D:\axis-1_3\remotelab>SET CLASSPATH-.;
Environment variable CLASSPATH-.; not defined

D:\axis-1_3\remotelab>javac -d . *.java

D:\axis-1_3\remotelab> start java org.apache.axis.transport.http.SimpleAxisServe
r -p 65535

D:\axis-1_3\remotelab>java org.apache.axis.client.AdminClient -p 65535 deploy.ws
dd
Processing file deploy.wsdd
<Admin>Done processing</Admin>

D:\axis-1_3\remotelab>_
```

C:\Program Files\java\jdk1.6.0_24\bin\java.exe

```
- starting up SimpleAxisServer on port 65535 (D:\axis-1_3\remotelab)
```

Figure 5.8 Snapshot of Starting Web Service in Axis Server

VI. CONCLUSION AND FUTURE WORK

In this project work, a new methodology is described for implementing remote lab. This proposal allows the student to do programming from any place with Internet access. The validation of the use of remote lab is compared with the traditional method of students physically attending to the lab. Results show a great dissimilarity between both which concludes that remote lab is more useful to students than the traditional methods and gives support to the students who actually face different problems and also useful for a student in saving time and energy. The Present work also explains how Remote lab is developed using Service Oriented Architecture with the help of Web Services and the reasons for why this method has been used. Developers can implement this application more easily because different technologies and execution platforms can easily be communicated through SOA method. There is a lot of scope for Service Oriented Architecture applications, as there is an increasing demand for cloud computing in the near future. This remote lab application is limited to students practicing programming languages and software side applications. It can be further developed to a next level where students can practice on physical equipments similarly as above.

REFERENCES

1. Binildas A. Christudas, Malhar Barai &amp; Vincenzo Caselli; “Service Oriented Architecture with Java: Using SOA and web services to build powerful Java applications”, Packt Publishing ltd, Birmingham, 2008.
2. Chen.S.H, Chen.R, Ramakrishnan.V &amp; others; “Chen.S.H,Chen.R, Ramakrishnan.V &amp; others; “Development of Remote Laborator Experimentation through Internet”, blog, 2008; http://discoverlab.com/References/vlabhksrc99.doc.
3. Debu Panda; “An Introduction to Service-Oriented Architecture from a Java Developer Perspective”, blog, 2005; http://onjava.com/pub/a/onjava/2005/01/26/soa-intro.html?page=1.
4. EdOrt ; “Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools”, blog, 2005; http:// java.sun.com/ developer/ technicalArticles/ WebServices/ soa2/.
5. Elio Sancristobal, Manuel Castro, Sergio Martin and others; “Remote Labs as Learning Services in the Educational Arena”; http://www. psut.edu.jo / sites/ EDUCON/ program/ contribution1481_b.pdf, blog, 2010.
6. Manoj Mansukhani ; “Service Oriented Architecture White Paper”, ftp://www.compaq.fr/pub/services/spotlight/info/soa_wp_062005.pdf, blog, 2005.
7. S. Uran, D. Hercog and K. Jezernik; “Remote Lab Experiment RC Oscillator for Learning of Control”, www.slideshare.net/avinashpathak/telelab-2 - United States, blog, 2010.

8. Grace A. Lewis, Dennis B. Smith, Kostas Kontogiannis; "A Research Agenda for Service-Oriented Architecture (SOA): Maintenance and Evolution of Service-Oriented Systems, Research, Technology, and System Solutions Program, March 2010.
9. Qusay H. Mahmoud, "Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI)" , Oracle Technology Network, April 2005.
10. Liang-jie , Zhang, "Web Services research for Emerging Applications: Discoveries and trends", feb 28,2010.
11. Eric J. Bruno, " SOA, web services, and restful systems" , June 8 2007 .

BIOGRAPHY

T.V.Lakshmi Sukrutha is an Assistant Professor in Computer Science Engineering Department, Methodist College Of Engineering and Technology, Hyderabad, Telangana, India. She has received Masters Degree in Computer Science Engineering in 2012 from Jawaharlal Nehru Technological University, Hyderabad, Telangana, India. Her research interests are Web Programming Services, Service Oriented Architecture, Principles of Programming Languages, Cloud Computing, and Computer Organization.