Social Aggregator

**Web Services**

Axis2, WSDL, MySQL, MTOM, Ant, Hibernate, JUnit and Java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [1. ABSTRACT 333](#_Toc203212192)  [2. INTRODUCTION 3](#_Toc203212193)  [3. OBJECTIVE 4](#_Toc203212194)  [4. PROPOSED SOLUTION 4](#_Toc203212195)  [4.1. Technologies and Approach 4](#_Toc203212196)  [4.2. Architectural Concept 5](#_Toc203212197)  [4.3. Design Patterns Used 7](#_Toc203212198)  [4.3.1. Black board approach 7](#_Toc203212199)  [4.4. Screenshots 10](#_Toc203212207)  [4.5. Testing 12](#_Toc203212208)  [5. Assumptions & Future Enhancements 15](#_Toc203212211)  [6. Hardware & Software Used 15](#_Toc203212212)  [7. Steps in running the project 15](#_Toc203212213)  [8. References 16](#_Toc203212214) | | 3  4  5  6  8  12  13  15  18  19  23 |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
|  | |  |  | |
| layout_center | 1. Abstract | | | |
|  | |  |  | |
| It is a collaborative project for which our team is providing web services for social site that uses aggregators to collect and locate information .It is a social aggregator using web services .Web service components use both a direct web service implementation and have web service component framework .Each aggregator is of different type like image, site, document and video. It does not share its process space .Traveling or traversal of the services is not centralized .The traversal uses a ring network as well as a blackboard approach .All the web services are queried using a metadata which each files in web service have like author , time ,location ,type and tags. The aggregation services will collect data from multiple sources and other services. The aim of the project is to design and develop a social aggregator using web services.  This document presents an overview of our design approach, assumptions, considerations, limitations and future enhancements. | | |  |  | |
|  | |  |  | |

|  |  |
| --- | --- |
| *layout_center* | 2. Introduction |

A web service is platform unbiased, language unbiased also can be retrieved over

World Wide Web.

In software industry, service oriented architecture have dissimilar sub-standard and openly-discussed description. Basically, it is an effort to provide collection of principles or main ideas utilized during stages of system development and integration. Such architecture is supposed to package functionality as a interoperable services within different business domains. Several departments within a company or different organizations may integrate or use such services, software modules provided as a service even if their respective client systems are substantially different. It is an attempt to develop yet another means for software module integration. SOA defines the interface in terms of protocols and functionality. An endpoint is the entry point to such an SOA implementation.

A web service has many different ports. Each one of them is a binding installed at a particular network address called endpoint. A binding is a port type using a specific layout of the message and specific transport protocol. A port type has many or single functions which have an input message as well as output message consisting of different parts. Each part is either a certain element defined in the schema of the web service, or any element belonging to a certain element type in that schema. All this is clearly described in Web Service Description Language (WSDL).

|  |  |
| --- | --- |
| layout_center | 3. Objective |

The project has the following core functionalities:

1. Separate services for video, images, sites, and documents.

2. Aggregators must be able to query a local data source and other aggregators.

3. Each service must be completely separate – do not share process space, database.

4. Implement all flavors of services they are deployed as separate services.

5. Traveling or traversal of the services is not centralized. Traversal must provide two types of traversal:

1. Asynchronous using a ring network.
2. Asynchronous using a blackboard approach.

# layout_center 4. PROPOSED SOLUTION

## Technologies and Approach

This section presents the technologies, used for implementing the system.

**Axis 2**

Apache Axis  is an open source XML based web service base framework. It consists of Java and a C++ implementation of the SOAP server, and various utilities and APIs for generating and deploying Web service applications. Using Apache Axis, developers can create interoperable, distributed computing applications. Axis is developed under the support of Apache software Foundation.

Apache axis 2 is our web service container and apache Tomcat is our web server. So in this project we are using Axis 2 as a container for creation of our web service as well as its deploy the web service in Apache tomcat.

**Soap**

SOAP is Simple Object access protocol, is protocol specification for exchanging structured information in the implementation on Web Services in Computer Networks. It relies on  XML as its message format, and usually relies on other application layer  protocols ( E.g. .Remote Procedure Calls(RPC) and HTTP) for message transfer and SOAP can form the foundation layer of web service protocol stack providing a basic messaging framework upon which web services are be built.

**Hibernate**

We have used hibernate in our project as an object-relational mapping to the database. Hibernate is an object relational mapping (ORM) library for the Java language, providing framework for mapping an object oriented to a traditional relational database. Hibernate solves object relational mismatch problems by replacing direct persistence related database accesses with high-level object handling functions.

Hibernate is free as open source software  that is distributed under the GNU lesser general public license Hibernate’s primary feature is mapping from Java classes to database tables . Hibernate also provides data query and retrieval facilities. Hibernate generates the SQL calls and relieves the developer from manual result set handling and object conversion, keeping the application portable to all supported SQL databases, with database portability delivered at very little performance overhead. The query language used by hibernate is HQL.

* 1. To retrieve the data from the database of each web service using Hibernate query language since we created separate database for each web service .
  2. To hibernate the data in each web service database

**Parallel processing ( Hadoop , Map reduce)**

Parallel processing is the ability of an entity to carry out multiple operations or tasks simultaneously. The simultaneous use of more than one CPU or processor core to execute a program or multiple computational threads. Ideally, parallel processing makes programs run faster because there are more engines (CPUs or cores) running it. In practice, it is often difficult to divide a program in such a way that separate CPUs or cores can execute different portions without interfering with each other. Most computers have just one CPU, but some models have several, and multi-core processor chips are becoming the norm. There are even computers with thousands of CPUs.

With single-CPU, single-core computers, it is possible to perform parallel processing by connecting the computers in a network. However, this type of parallel processing requires very sophisticated software called distributed processing software.

**MTOM**

MTOM is the Message Transmission Optimization Mechanism according to the standards of W3C, a way of proficiently transfer of binary data to the Web Services and vice versa. It uses XOP (XML-binary Optimized Packaging) to send binary data and is intended to replace both MIME an DIME attachments. Binary content often has to be reencoded to be sent as text data with SOAP messages. MTOM allows more efficient sending of binary data in a SOAP request or response.

In our case we use MTOM for transfer of data from one web service aggregator to another in the ring network.

.

**JUnit Test Tool**

JUnit is a simple open source Java testing framework. It is used to write and run repeatable automated tests. Eclipse supports creating test cases and running test suites, so it is easy to use for the Java applications. Using the JUnit test tool, functionality can be tested at very unit level. While the development is in progress before all pieces work together we want to make sure each unit is working properly. So, it is very helpful for debugging at unit level. It is because of the above reasons we have selected JUnit test tool and performed the testing.

|  |  |
| --- | --- |
| *layout_center* | 4.2. Architecture |

**Working of the ring network:**

Document Thread

Image Thread

message

message

message

message

hql

hql

hql

hql

hql

hql

Internet

Video Thread

Site thread

WWW

**Client**

**Yahoo Search**

**Third party video**

**Image Service**

**Site Service**

**Video service**

**Document service**

**Look up service**

The important components of the ring network are the **Client, the lookup service, Image service aggregator, Document service aggregator, Video service aggregator, Site aggregator** and the **Third party web service.**

The client queries the image server which is the first in the ring. The client queries the database based on the metadata. Each web service has its own separate database and the data is mapped using hibernate. When the client sends a query to the image service the image server spawns a thread which retrieves the data from the local image database collects the data and contacts the look up service to check for the next node. Once the lookup service replies the image thread contacts and gives collected data in the form of message to the document server. The thread keeps a note of the originating node, message and the query.

Now the document server spawns a new thread retrieves the data from the local document database using hibernate query language appends the previous data collected and contacts the look up service for the next node to send the data. After getting a response sends the data to the video service .Now the video server spawns a thread to retrieve the data from its local video database based on the query appends the previously collected data and contacts the lookup service for the next node.

The look up service now will send the next node and the thread sends message data and contacts the site server . Here based on the hql the site server spawns a thread and retrieves the data from site local database append the previous data and then contact the look up service for the next node, which on reply the thread goes back again to the image server identify it based on the originator node in the thread .Once the thread knows it sends all the data back to the client. Thus the client will receive all the images , documents ,videos, sites and third party videos based on the query.

**Parallel processing architecture**

HQLHQLHQL

HQL

Image aggregator

Document

aggregator

Video

aggregator

Site

aggregator

messagemessagemessage

Site Thread

Video Thread

Image thread

Document

Thread

Parallel Processing

Module

message

**C:\Program Files (x86)\Microsoft Office\MEDIA\CAGCAT10\j0292982.wmf**

**Client**

**Working of the Parallel processing architecture:**

The main components in the parallel processing are the parallel processing unit ,the image service ,document service , video service and the site service including yahoo search.

The client queries the parallel processing unit based on some metadata .Now the parallel processing unit spawns four independent threads for four different web service like image thread for image server ,document thread for document server, video thread for video server and site thread for site server including yahoo search. Here image thread will contact the image server retrieve the data based on the hql query from the local image database and the image thread will send back the collected data to the parallel processing unit. The document thread will contact the document server , retrieve the document data from the local document database and send the result to the parallel processing unit, The video thread spawned will contact the video server, retrieve data from local as well as third party and give it to the parallel processing unit. Now the last one is the site thread spawned will contact the site server retrieve data based on the query from local site database and send the result to the parallel processing unit. The parallel processing unit after receiving all the data from all threads will append the data together and send back to the client. Hence here blackboard approach has been implemented. Thus the process of spawning all the threads to the all the four web service is called mapping process while the sending all the data collected from different web service back to the parallel processing unit and to the client is call reducing. Hence we are mapping and reducing the data.

|  |  |
| --- | --- |
| layout_center | **4.3 Design Pattern** |

**Black Board design pattern:**

The blackboard pattern generalizes the observer pattern to permit multiple data sources as well as multiple viewers. It also has the effect of completely decoupling producers and consumers of information.

A blackboard is a repository of messages which is readable and writable by all processes. Whenever an event occurs that might be of interest to another party, the process responsible for or knowledgeable about the event adds to the blackboard an announcement of the event. Other processes can read the blackboard. In the typical case, they will ignore most of its contents, which do not concern them, but they may take action on other events. A process which posts an announcement to the blackboard has no idea whether zero, one, or many other processes are paying attention to its announcements.

Blackboards generally do not enforce a particular structure on their announcements, but a well-understood message format is required so that processes can interoperate. Some blackboards provide filtering services so that clients do not see all announcements.

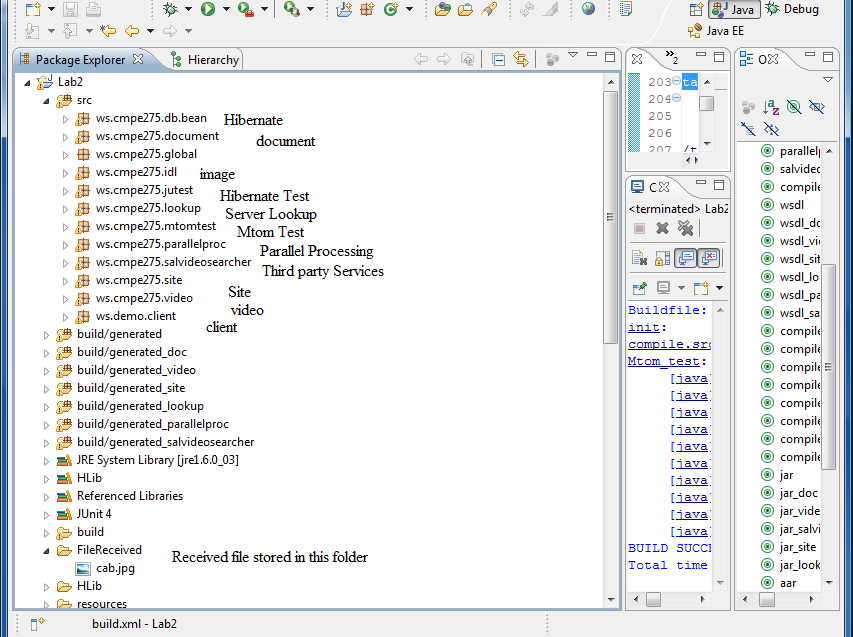
An ordinary bulletin board (either the physical or the electronic kind) is an example of a blackboard system.

The idea behind the Blackboard architecture is a collection of independent programs that work cooperatively on a common data structure. Each program is specialized for solving a particular part of the overall task, and all programs work together on the solution. These specialized programs are independent of each other. They do not call each other, nor is there a predetermined sequence for their activation. Instead, the direction taken by the system is mainly determined by the current state of progress. A central control component evaluates the current state of processing and coordinates the specialized programs. This data-directed control regime makes experimentation with different algorithms possible, and allows experimentally-derived heuristics to control processing.

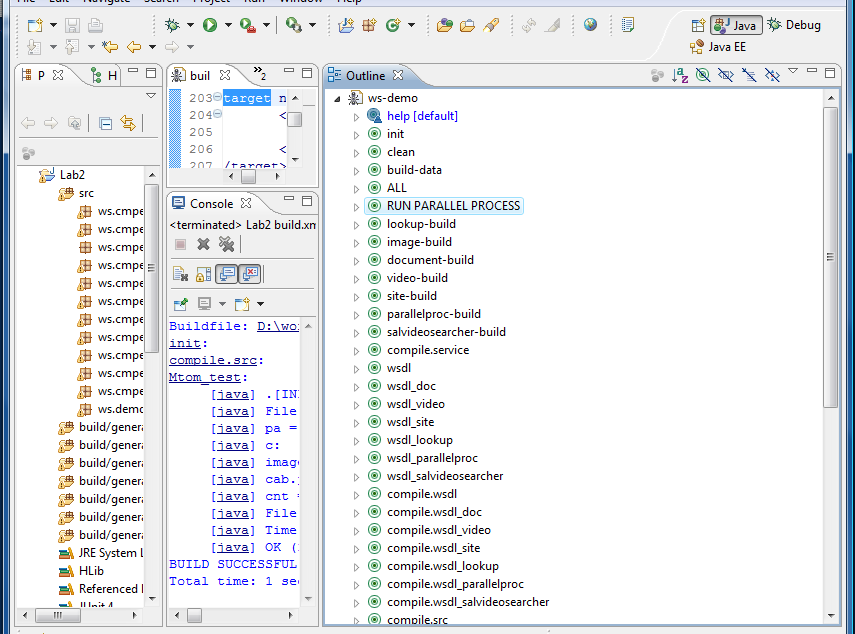
In the project the client calls the parallel processing module and the module invokes four different threads in java to four web service aggregators. The threads are programs for solving particular part of the overall task. They give back the result at the same time to module and then to the client. So the result is pretty quick as compared to other approaches.

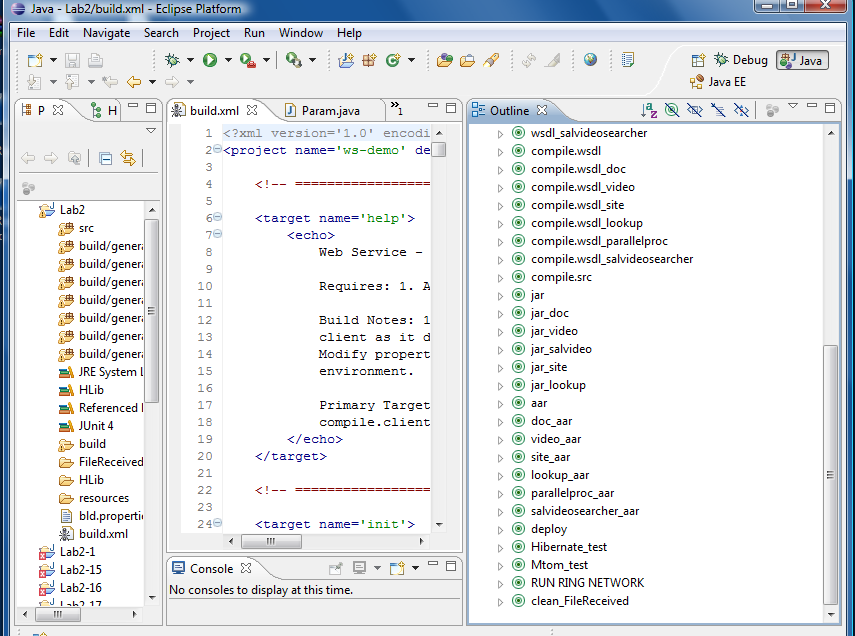
|  |  |
| --- | --- |
| layout_center | **4.4 Screenshots** |

Lab2 project class descriptions.



Ant script descriptions.





|  |  |
| --- | --- |
| layout_center | **5. Assumptions & Future Enhancement** |

**Assumptions:**

In the ring network we have assumed that web services register to the lookup

service.

We assumed that our parallel processing unit performs the job of mapping and reducing like the hadoop, GridGain.

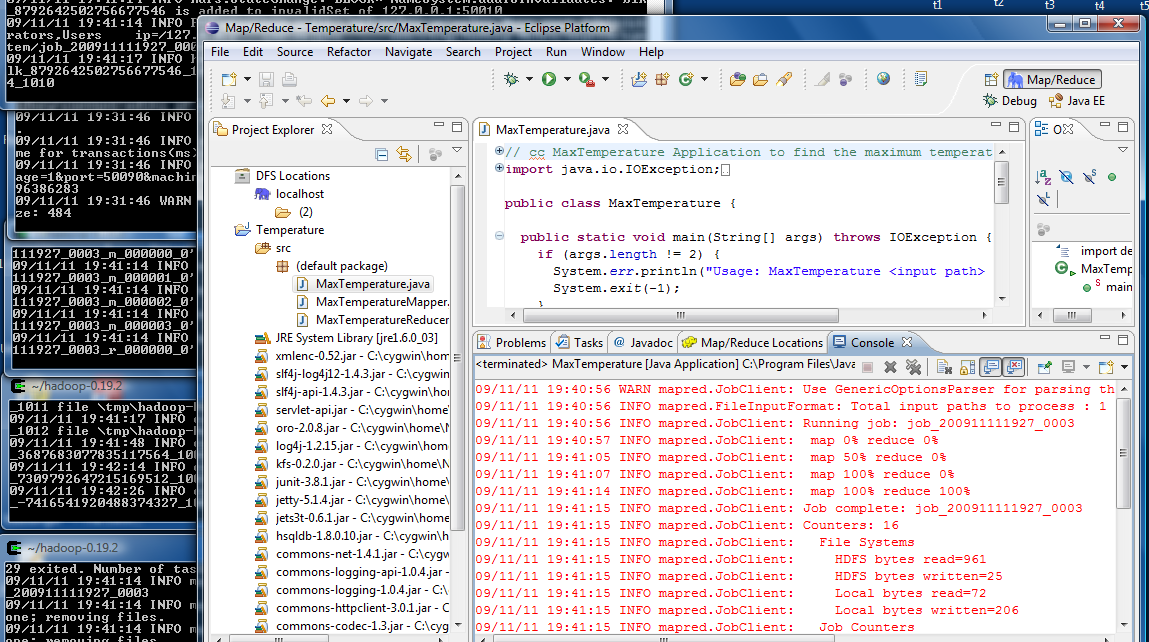
**Future Enhancements**:

Since the ring network is asynchronous if more number of nodes having different web services join and register with the lookup service then also the ring will work.

The parallel processing architecture contains a parallel processing unit which maps and reduces the data like hadoop or Gridgain hence this architecture can be used for any number of web services for different nodes

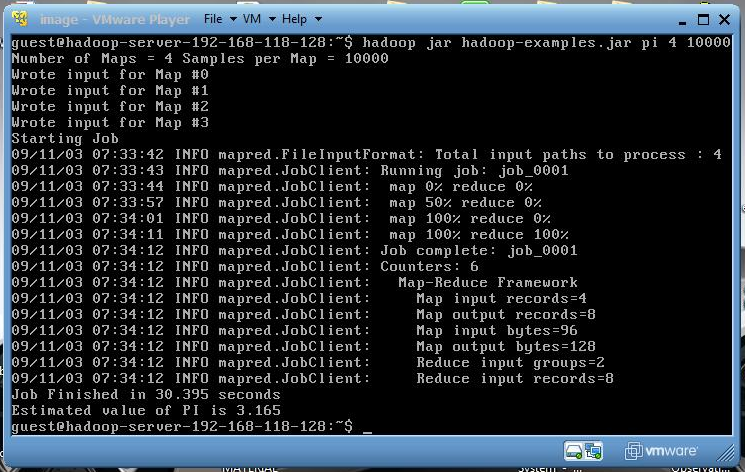
|  |  |
| --- | --- |
| layout_center | **6. Research on Hadoop** |

We ran setup hadoop cluster and ran sample program.



VMWare Player.

We installed VMWare player and ran sample program.



After researching all these technologies for parallel processing, we decided to develop parallel processing unit. And we implemented parallel processing of search using mult-threading.

.

|  |  |
| --- | --- |
| layout_center | **6. Hardware & Software Used** |

**Hardware:**

Processor: Intel Core 2 duo 2.0 GHz

Ram: 4 GB

HD: 180 GB

**Software**:

Hibernate 3.0

Eclipse GA

MTOM

JUnit

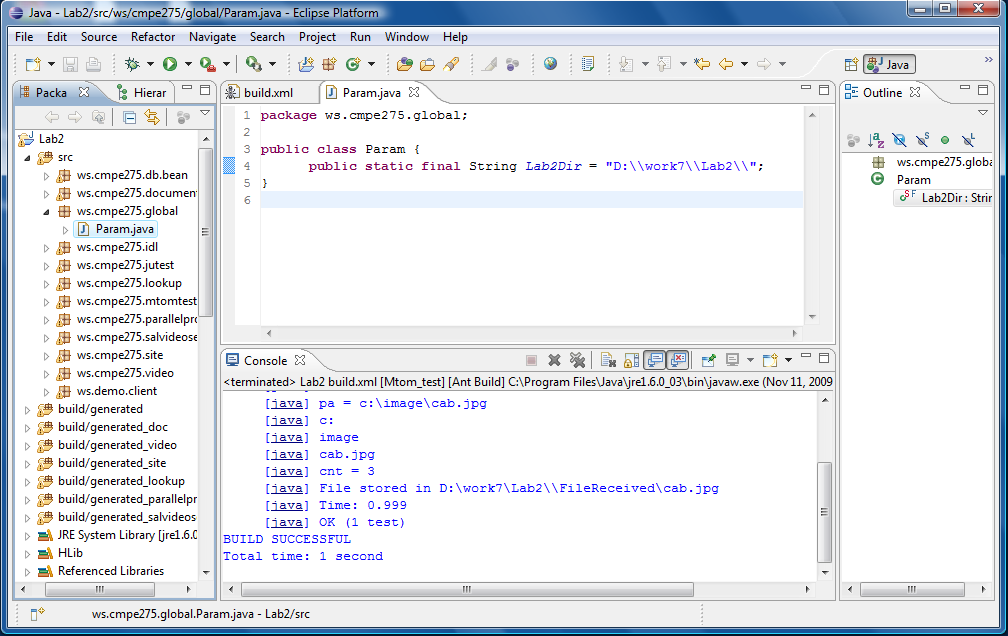
Apache ANT

Apache axis 2.0

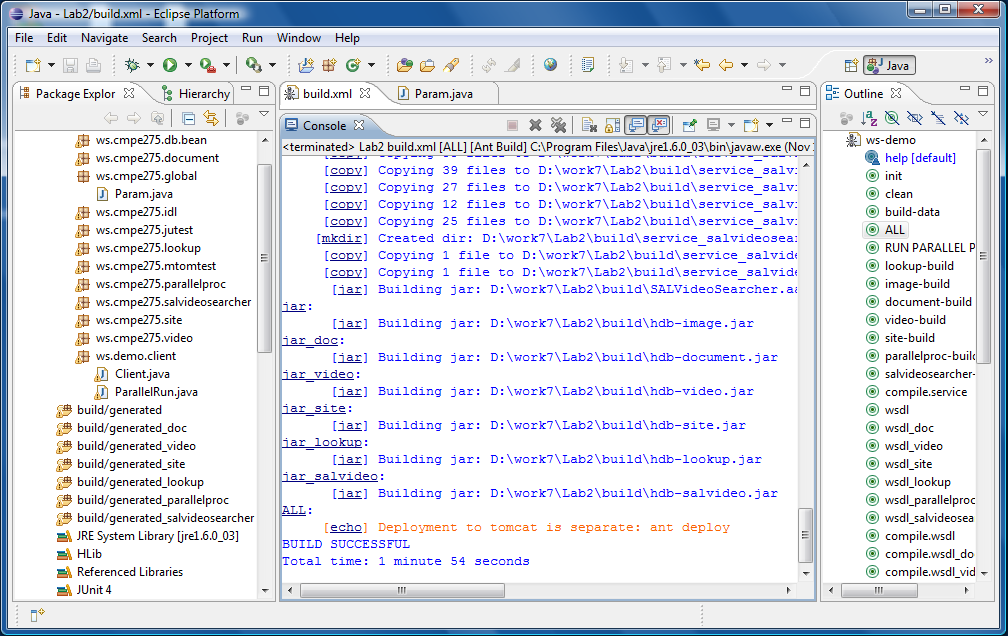
|  |  |
| --- | --- |
| layout_center | **7. Steps for running the project** |

1. Change Lab2Dir in Param.java in package ws.cmpe275.global to working directory for the project. Our working directory for this project is

*Lab2Dir* = "D:\\work7\\Lab2\\";



1. Build the project using ALL in the ant script



1. Setting up Tomcat server

We created 7 tomcat server running on different ports they are

C:\t1 – port 8080 – ImageServer

C:\t2 – port 8081 – DocumentServer

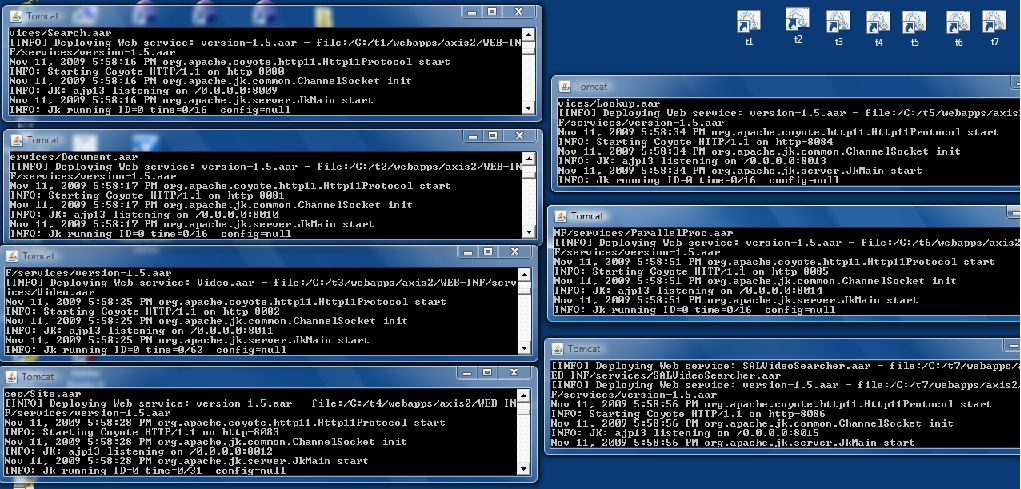
C:\t3 – port 8082 – VideoServer

C:\t4 – port 8083 – SiteServer

C:\t5 – port 8084 – LookupServer

C:\t6 – port 8085 – ParalledProcessingUnit

C:\t7 – port 8086 – SALVideoServer – Third Party



1. By running “RUN RING NETWORK” from the ant script we can see the searched data from the each of the services from Image, document, video, site, third party and yahoo search.

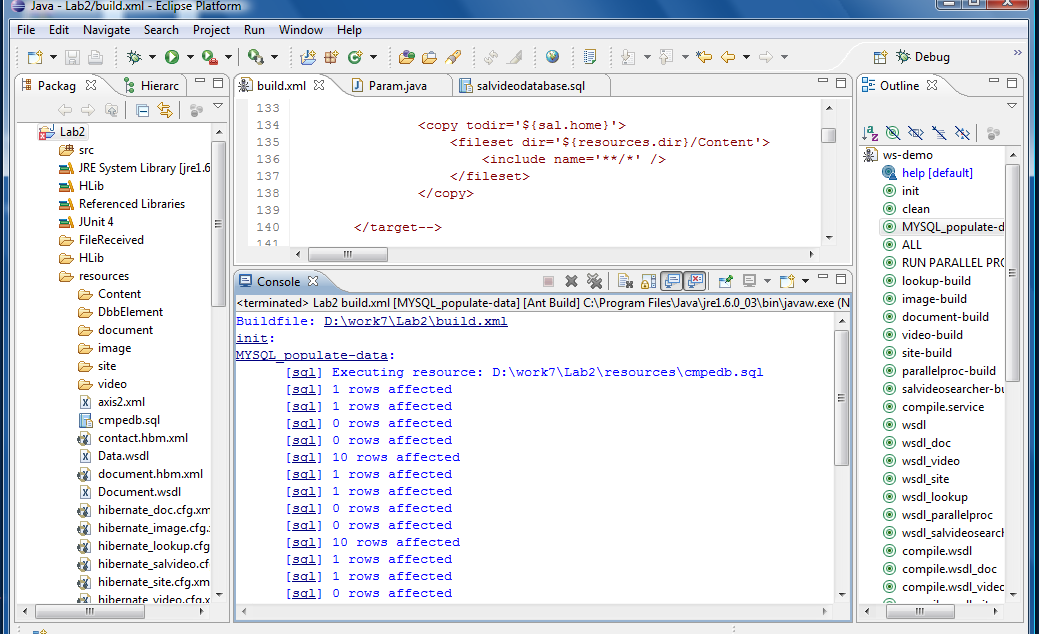
We used yahoo search API from axis2 samples. These codes helps us to search yahoo site from the site services.

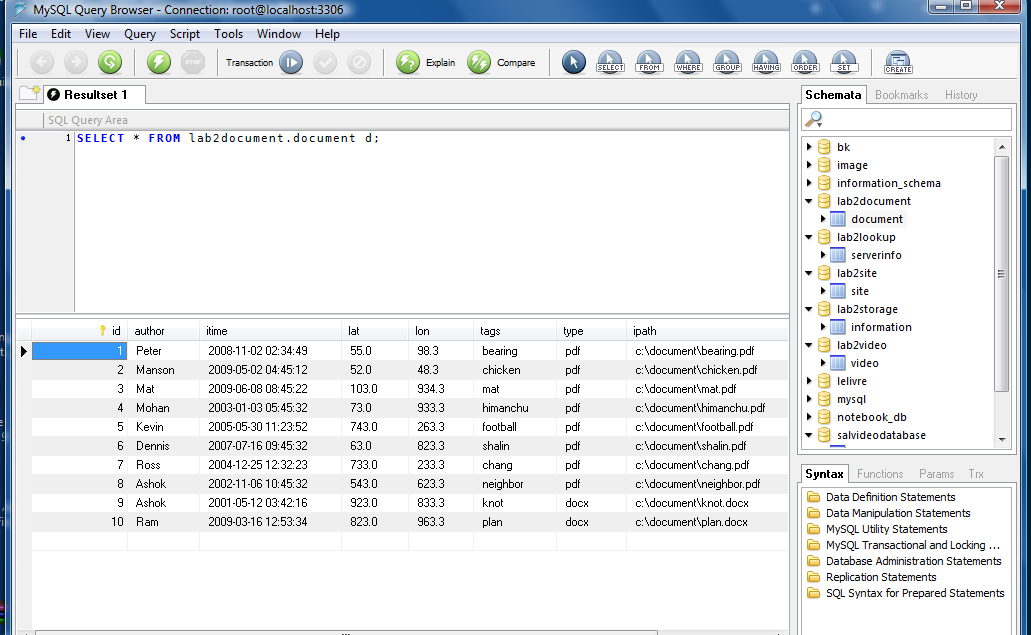
C:\axis2\samples\yahoojsonsearch

The received file from the services will be saved to FileReceived folder in the Lab2 project directory.

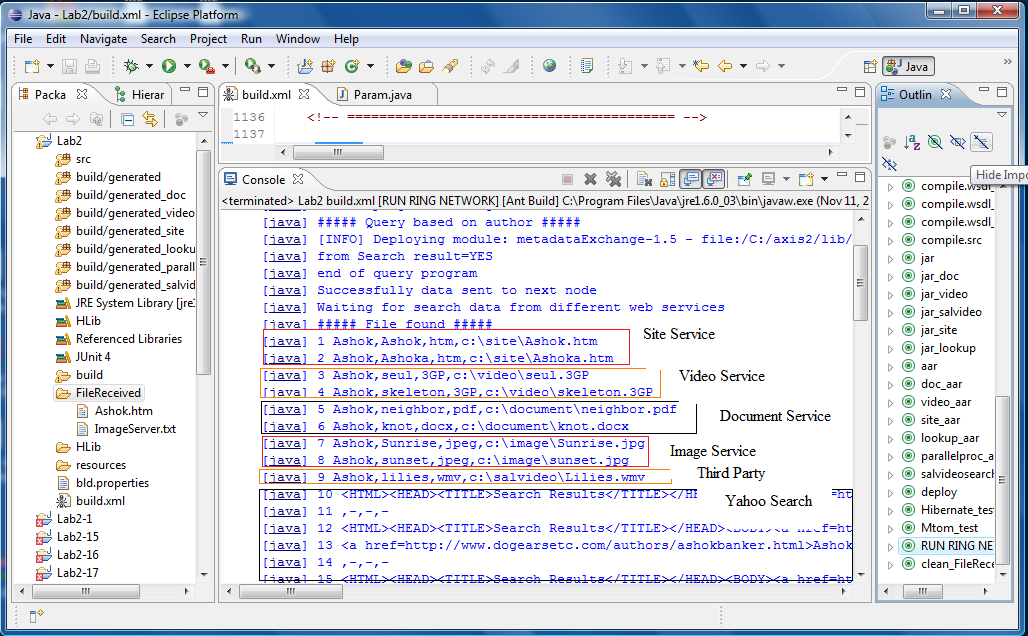
MySQL Setup

MySQL\_populate\_data in ant script will populate mysql databse for all the web services.

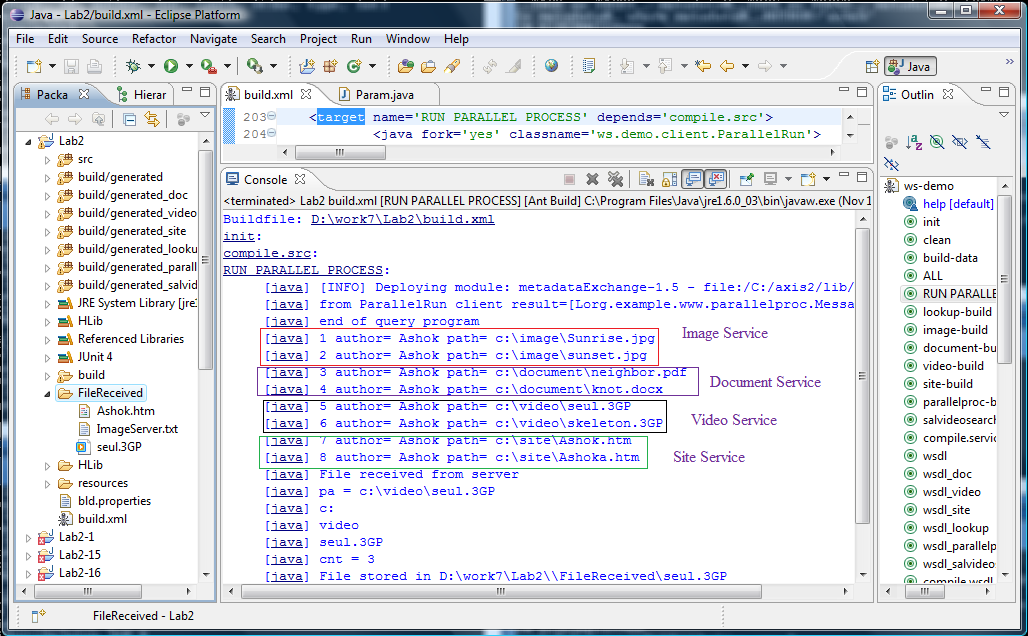




By running “RUN RING NETWORK” from the ant script will produce the output as shown below.



1. By running “RUN PARALLED PROCESS” we can see the search data from the four services.

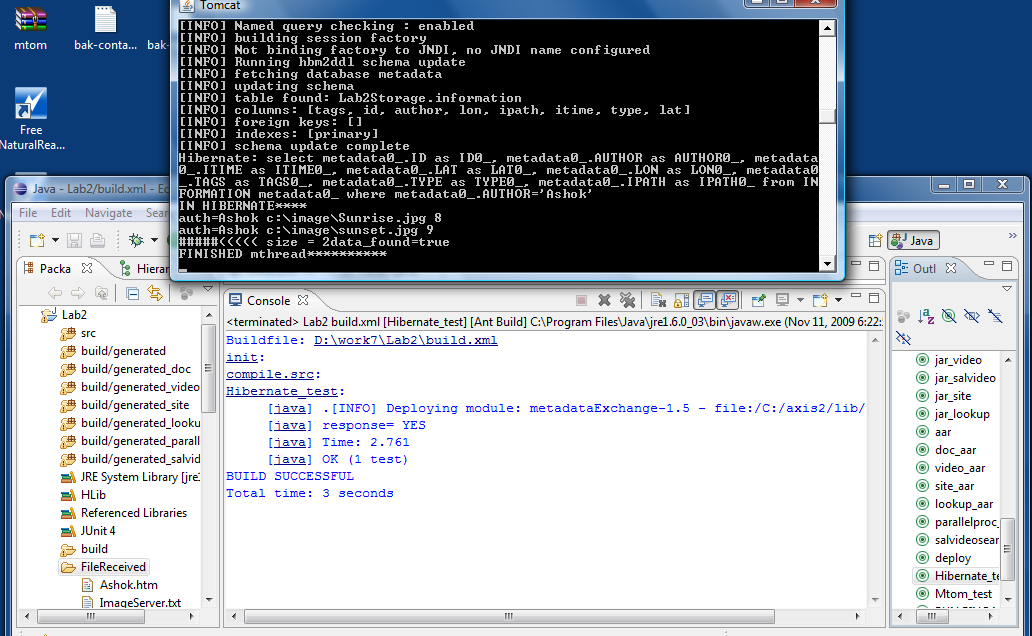


|  |  |
| --- | --- |
| layout_center | **8.0 Testing** |

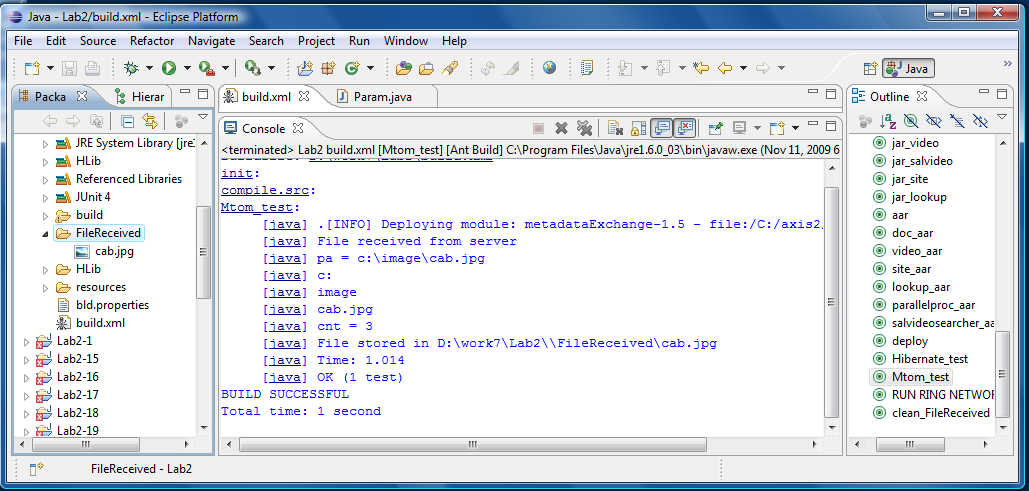
**Hibernate**

Junit test module for hibernate validation.

We know that author named ashok is available in the database



**MTOM Testing**



|  |  |
| --- | --- |
| layout_center | **8. References** |

<http://www.wikipedia.org/>

<http://www.vico.org/pages/PatronsDisseny/Pattern%20Blackboard/>