ISSN: 2230-9926

Available online at http://www.journalijdr.com

IJDR

International Journal of Development Research
Vol. 07, Issue, 09, pp.15140-15143, September, 2017

ORIGINAL RESEARCH ARTICLE OPEN ACCESS

IMPLEMENTATION OF JAVA MODULE FOR PRINT SERVER APPLICATION

[1]Shwetha V., [2]Dr. Luke Melita and [3]Dr. Amruth Ramesh Thelkar

[1]Department of computer science Engineering, Maharaja Institute Technology, Visvesvaraya technological University, Karnataka, India
[2]Assistant Professor, School of Computing,Jimma Institute of Technology, Jimma University, Ethiopia
[3]Assistant Professor, School of ECE, Jimma Institute of Technology, Jimma University, Ethiopia

ARTICLE INFO

Article History:
Received 04th June, 2017
Received in revised form
07th July, 2017
Accepted 20th August, 2017
Published online 29th September, 2017

Key words:

Client JAVA Module,
Server JAVA Module,
Print job, Print Queue,
Printer,
Snippet.

ABSTRACT

In modern era a print server is a device that connects printers to client computers over a network. Print server supports a variety of industry standard or proprietary printing protocols including Internet printing protocol, line printer daemon protocol, Netware, NetBIOS/NetBEUI or Jet Direct. When the clients are in network and the printer does not support the network connection then we use the Print Server software where all the clients are connected to a single server and the server is connected to the printer. Print server is to build upon the windows application. The most executable JAVA viz Client screen JAVA and Server JAVA has been implemented to accomplish the printing task from various computers without any interruption.

Corresponding author

Citation: Shwetha V., Dr. Luke Melita and Dr. Amruth Ramesh Thelkar. 2017. "Implementation of java module for print server application.".", *International Journal of Development Research*, 7, (09), 15140-15143.

INTRODUCTION

A print server plays a very important role in modern era. As all office work is more computers dependent, our resources have to be well utilized. That's where a print server plays an important role. As there is only one printer in most of the offices, the entire printing job has to be done on a single printer. Often it is the case that, a printer connected to a single computer sits idle for a long time; and so there arose a need to find a facility that would allow many users in a network to utilize the printer. A printer shared in a network causes scheduling and resource problems. Printer sharing requires an intelligent management scheme to store and line up the print jobs separately, allowing computers to continue other processing while the printer is doing its tasks.

The remedy was a print server that receives its printing instructions from several computers, stored them in a queue and fed them to the shared printer [Herbert Schildt, 2002].

Advantages

Reduction in the number of printers required:

If a conventional local printing technique is used, each user who needs to print must have his or her own personal printer. This scenario can be very expensive and hard to maintain.

Check on number of pages printed:

In an organization, keeping a check on the number of pages printed by each employee is possible. A restriction on the number of pages can be set.

Use of a single high-performance printer

Instead of a company making a compromise on price and performance of printers due to the requirement of many printers, a single high performance, and expensive printer can be invested in (Andres et al., 2017).

METHODOLOGY

The proposed system is a replacement for the existing system where only one user can use the printer at a time. The users can be of organization, institutes or any offices where there is only a single printer which does not support network and multiple users in network. The product features are like Use for a single high-performance printer, accepts the printing job only for authorized users**, r**eduction of the number of printers required, a check on number of pages printed and authorized Printouts (Huiling, 2013). The Print Server is developed by Using Java. Here we are to be aware of the network basics and also how the files are transferred between the systems. Also how the networking is done in Java.

The Print Server has two main parts, they are:

Server side - This acts as the base for the interconnection and network handling.

Client side - Which acts as an interface to select and send the print file that is to be printed.

The server acts as the main base for the data transfer from client to the printer. The main job of the server is to get the clients request and print the jobs in the First In First Out basis. Figure1 shows the Activity Diagram of Server Side.

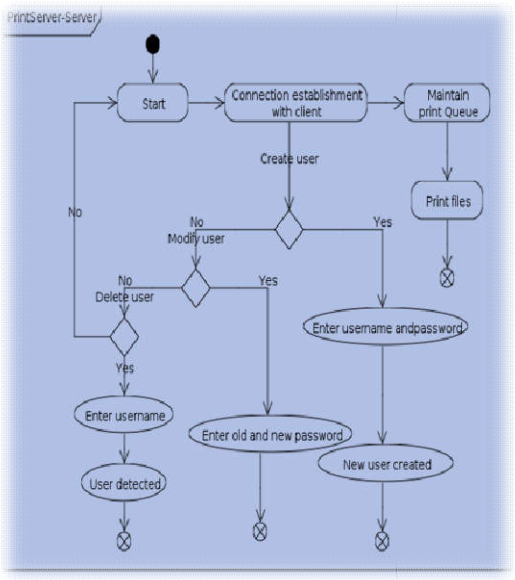

Figure 1: Activity Diagram of Server Side

It has to maintain three options: *Create User* - to create a new user, *Modify User* – to modify the password of the already existing user, *Delete User* – to delete an already existing user.

The client acts as the user side data transfer who will help the user to select the required file to be printed and to send that file to the server for printing.

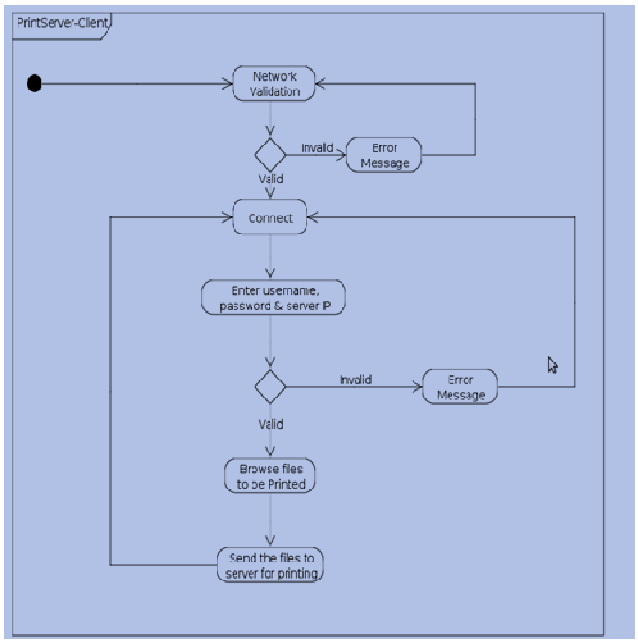

Figure 2: Activity Diagram of Client Side

The connection should be established where only the valid users can connect to the server by knowing the IP address of the server and then the data transfer occurs. There is a queue which says its contents. The print queue contains the number of files that are to be printed by the server from different clients. The connection established by the client is disconnected once the printing job is done. Figure1 shows the Activity Diagram of Client Side.

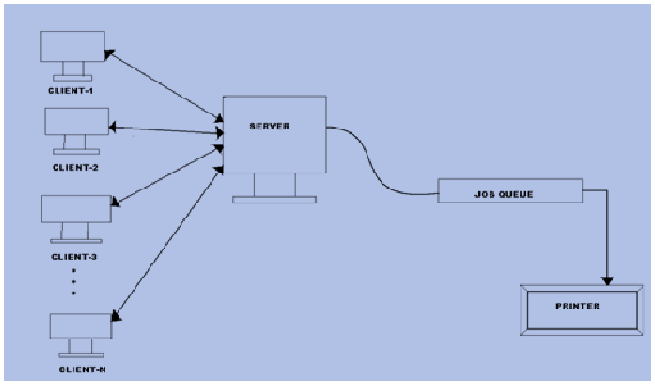

Figure 3: Overall concept of Print Server

Figure 3 shows the overall concept of Print Server. In this model there are five phases namely, requirements, design, implementation, testing, deployment and maintenance.

- In first phase, we collected the requirements and analyze in detail to understand the problem. In design phase, we tried to answer for many questions by preparing models and entities interaction model and we tested few core algorithms.
- In implementation phase, we implement the design in java programming language and have run the application. In testing phase, we test the application based on many criteria.
- In deployment and maintenance phase, we install the developed software at the customer end and maintain them. In deployment stage, under real environment, we need all clients' systems to be in network. And treat a

machine as server by installing the server software and parallely install the client's software in client's machine and need to connect printer for the server which doesn't support network, server is connected to all clients.

Need to run the server software which opens a connection for clients and then run the client side software to carry out the prescribed job.In maintenance, we check whether the given software is working as per the requirement andany changes to be required to make the software much more enhanced. It has been entirely developed on the WINDOWS OS, which provides a robust and failsafe platform for software development. It was developed keeping in mind the corporate scenario where Windows based Operating Systems are used extensively for providing network services.

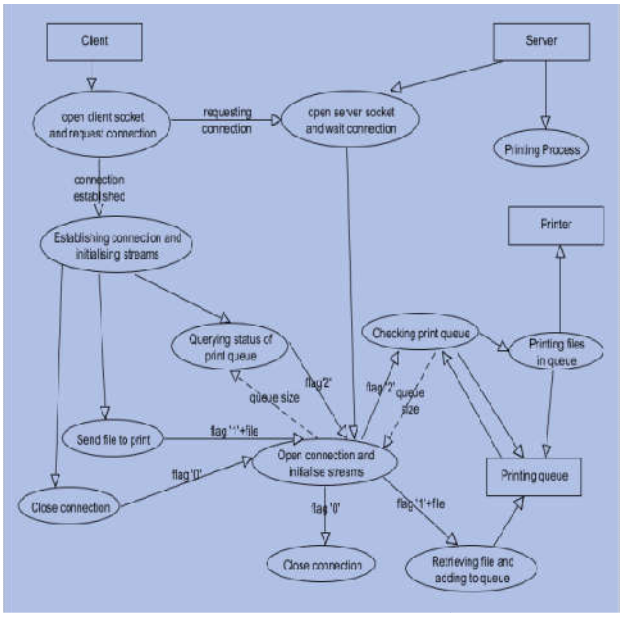

Figure 4: Data Flow Diagram for Print Server

The two main executable JAVA files implemented were:

- ClientScreen.java
- Sever.java

In Client java module, the client opens a socket at *port number 1234,* for connection. It opens one object of input stream class and places a particular file in that stream. The BufferedReader class creates one object, which reads the file byte by byte and places in the stream. OutputStream class creates an object, which transfers the file to the server. Now the communication between the client and server is established. All the opened in-built stream classes have to be closed.

Our server is multithreaded module in which each request is handled by a separate thread. The server module begins with the creations of a *server at port 1234*.

Server sckServer = new ServerSocket (1234);

Now the server is listening for a request from client. It accepts a request using the following command:

Socket sckClient = sckServer.accept ();

A separate thread handles each client request. Then the required IO operation for the printer is performed. If a document is queued on to the printer, a separate thread which handles the printer events informs the client about the document.

Client Module

The client is started using the command:

java ClientScreen

A Socket is opened.A Connection with server is made by giving the server IP address.An input and output stream to the socket is opened.File can be chosen using File Open Dialog box. The content of the file is read from the disk.The size, extension and contents of the file are sent to the server.If the file transfer is successful then an acknowledgement is sent to the client by the server.All the streams and connections are closed.

Snippet in which server and client establishes a connection and send file for print

```
String srvUrl;//Server IP address
Socket sck;
SendFile sf;
sck = new Socket(srvUrl,5167);
sf = new SendFile();
```

Snippet to send file to server

```
SendFile(Socket sck)
{
        newClt = sck;
try {
//outStream to write to serveroutStream =
newPrintStream(newClt.getOutputStream() );
                //instream to read from serverinStream =
newBufferedInputStream(newClt.getInputStream() );
        }
catch(Exception e)
 {
                System.out.println("Unable To
Initialize Streams..");
        }
}
```

Server Module

Sequences of operations are to be performed by the server module. A server socket is created to listen to a specific port. The print queue vector is initiated to store print request. The server starts concurrent thread to monitor the contents of the print queue vector and another thread to handle events from printer. The server socketruns in an infinite loop waiting for the incoming connections from a client.Accepting the connections: the 'accept()' method of the server waits until a client starts up and request for a connection on the hosts. Creating a new socket: when a connection is requested by a client, the server successfully establishes a connection, the accept() method returns a new socket object which is bound to a new port. The socket input, Output stream are obtained and a reader and writer is opened on them. The server checks the operation code sent by the client to determine the action to be performed. If the operation code is '0', the client sends a connection closing request to server and closes the active

connection. If the operation code is '1', the client wishes to send a fresh print job. The file content and related information are received in the form of packets from the client and is stored in the print queue vector. If the operation code is '2', the client wishes to monitor status of the print queue. The server then queries the vector and returns the current status of the print queue. The Thread monitoring the vector checks the contents of the vector at regular intervals of time. If the job vector is not found empty, it picks up the print job from the head of the queue and sends it to the printer for printing. It deletes the job from the queue after printing is complete.

Server Side Snippets

```
ServerSocket sckServer = new ServerSocket(5167);//server socket to listen

Socket clt = sckServer.accept();  //accepting the client request
InetAddress a = clt.getInetAddress();  //getting the address of client address
Connection con = new Connection(clt,flVect);  //creating new connection for the client
```

Using vector to store files that is to be sent for printing

```
Vector flVect; = new Vector();  //vector to handle files to print as queue
new Server(flVect);
new PrintFile(flVect);
```

To get file from the vector and create file object for printing

```
void getFile() {
        data = (DataObj)flVect.elementAt(0);    //getting the first element
        flName = data.getName();
        flSize = data.getSize();
        uName = data.getUser();
System.out.println("Reading File ["+uName+"_"+ flName +"]");
prntFile = newFile("temp/"+uName+"_"+flName);
}
```

For printer to accept files for printing

```
if (!flVect.isEmpty()) {//if vector is not empty
getFile ();  //get the first file to print
//System.out.println("File Got" + prntFile.getName());
if(pJob.print(prntFile)) {    //printing the file
flVect.removeElementAt(0);//if success removing from //vector
data = new DataObj();    //setting dataobj to null, dataobj is //required for creating file structure.
}
```

Conclusion

- The print server automates the entire process of printing in a network, eliminating many manual and time consuming steps.
- The system considerably reduces the cost of printing, by allowing one printer to be shared by multiple clients.
- Once client submits his printing request he can continue with his other works. System has been developed very user friendly for doing reliable printing.
- Backing up uncompleted jobs is required when the server is closed down without completing all the requested jobs.
- It has to back up and restore the uncompleted jobs when server is restarted and notify the user about his requested job such as job is completed, job is cancelled etc.

REFERENCES

Andres, G., Sanchez, G. 2017. A Robert Vaca, Satiago Manzano, "Print server with web user interface for real time image control and monitoring of a 3D printer using open source tools." IEEE conference publications,DOI:10.23919/CISTI.2017.7975957,1-4.

Herbert Schildt, 2002. 7th Edition JAVA™ : The Complete Reference, Tata McGraw-Hill Publishing Company Ltd.

Huiling MA, Yongbin Zhang, 2013. "The design and implementation of a process based printing order management system", IEEE conference publications, DOI:10.1109/ISCC-C.2013.137,48-53.

****