Assignment – 4

**1. Write an RMI based program to read a file from another machine.**

**# GetFile.java :-**

package Assignment4.Question1.Server;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface GetFile extends Remote

{

public String readFile(String file) throws RemoteException;

}

**# GetFileClass.java :-**

package Assignment4.Question1.Server;

import java.io.File;

import java.io.FileReader;

import java.io.BufferedReader;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class GetFileClass extends UnicastRemoteObject implements GetFile

{

protected GetFileClass() throws RemoteException

{

super();

}

@Override

public String readFile(String file\_path) throws RemoteException

{

StringBuilder fileContent = new StringBuilder();

try

{

File file = new File(file\_path);

if (!file.exists() || !file.isFile())

{

return "File Not Found";

}

try (BufferedReader fileReader = new BufferedReader(new FileReader(file)))

{

String line;

while ((line = fileReader.readLine()) != null)

{

fileContent.append(line).append("\n");

}

}

}

catch (Exception e)

{

e.printStackTrace();

return "Error reading file: " + e.getMessage();

}

return fileContent.toString();

}

}

**# Server.java :-**

package Assignment4.Question1.Server;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Server

{

public static void main(String[] args)

{

try

{

GetFile getFileClass = new GetFileClass();

Registry reg = LocateRegistry.createRegistry(1020);

reg.bind("gfc", getFileClass);

// Naming.rebind("gfc", getFileClass);

System.out.println("Server is running...");

}

catch (Exception e)

{

System.out.println("Error : " + e.getMessage());

e.printStackTrace();

}

}

}

**# Output :-**

Server is running...

**# Client.java :-**

package Assignment4.Question1.Client;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.rmi.Naming;

import Assignment4.Question1.Server.GetFile;

public class Client

{

public static void main(String[] args)

{

try

{

GetFile getFileClass = (GetFile) Naming.lookup("rmi://localhost:1020/gfc");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.print("Enter the file Name : ");

String file = br.readLine();

System.out.println("\nFile Content...");

String fileContent = getFileClass.readFile(file);

System.out.println(fileContent);

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

**# Output :-**

Enter the file Name : LICENSE

File Content...

MIT License

Copyright (c) 2024 Khushi 💜

Permission is hereby granted, free of charge, to any person obtaining a copy

of this software and associated documentation files (the "Software"), to deal

in the Software without restriction, including without limitation the rights

to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

copies of the Software, and to permit persons to whom the Software is

furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE

SOFTWARE.

**2. Write an RMI based program to send the request to a remote function, to fetch a record from the Book database for a particular book and the send the result to the requesting client.**

**# BookStore.java :-**

package Assignment4.Question2.Server;

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.List;

import java.util.Map;

public interface BookStore extends Remote

{

public List<Map<String, Object>> getBookInfo(int id)throws RemoteException;

}

**# BookStoreClass.java :-**

package Assignment4.Question2.Server;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.sql.\*;

import java.util.\*;

public class BookStoreClass extends UnicastRemoteObject implements BookStore {

protected BookStoreClass() throws RemoteException {

super();

}

@Override

public List<Map<String, Object>> getBookInfo(int id) throws RemoteException {

List<Map<String, Object>> bookList = new ArrayList<>();

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/books?characterEncoding=latin1", "root", "khushi");

PreparedStatement pst = con.prepareStatement("SELECT \* FROM book WHERE bookId = ?");

pst.setInt(1, id);

ResultSet rs = pst.executeQuery();

ResultSetMetaData metaData = rs.getMetaData();

int columnCount = metaData.getColumnCount();

while (rs.next()) {

Map<String, Object> book = new HashMap<>();

for (int i = 1; i <= columnCount; i++) {

book.put(metaData.getColumnName(i), rs.getObject(i));

}

bookList.add(book);

}

rs.close();

pst.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return bookList;

}

}

**#Server.java :-**

package Assignment4.Question2.Server;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Server

{

public static void main(String[] args)

{

try

{

BookStore bookStore = new BookStoreClass();

Registry reg = LocateRegistry.createRegistry(1304);

reg.bind("b", bookStore);

System.out.println("Server is running...");

}

catch (Exception e)

{

System.out.println("Error : " + e.getMessage());

e.printStackTrace();

}

}

}

**# Output :-**

Server is running...

**#Client.java :-**

package Assignment4.Question2.Client;

import Assignment4.Question2.Server.BookStore;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.List;

import java.util.Map;

public class Client

{

public static void main(String[] args)

{

try

{

Registry registry = LocateRegistry.getRegistry("localhost", 1304);

BookStore bookStore = (BookStore) registry.lookup("b");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.print("Enter the Book ID : ");

int bookId = Integer.parseInt(br.readLine());

List<Map<String, Object>> books = bookStore.getBookInfo(bookId);

if (books.isEmpty())

{

System.out.println("Book Not Found...!");

}

else

{

for (Map<String, Object> book : books)

{

for (Map.Entry<String, Object> entry : book.entrySet())

{

String key = entry.getKey();

Object value = entry.getValue();

System.out.println(key + ": " + value);

}

System.out.println("-----------------------------");

}

}

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

**# Output :-**

Enter the Book ID : 2

priceOfBook: 60.78

authorNames: Sahil Bhanderi

publication: Sahil Publication

totalQuantityToOrder: 200

bookName: Life of CR

totalCost: 10000.00

bookId: 2

dateOfPublication: 2030-03-01

-----------------------------

**3. Write an RMI based program to get the information about the Operating System version, total size & available size of hard disk, total size of memory & memory in use of a remote machine.**

**# SystemInfo :-**

package Assignment4.Question3.Server;

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.Map;

public interface SystemInfo extends Remote

{

Map<String, Object> getSystemInfo() throws RemoteException;

}

**# SystemInfoClass :-**

package Assignment4.Question3.Server;

import java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;

import java.io.File;

import java.util.HashMap;

import java.util.Map;

public class SystemInfoClass extends UnicastRemoteObject implements SystemInfo

{

protected SystemInfoClass() throws RemoteException

{

super();

}

@Override

public Map<String, Object> getSystemInfo() throws RemoteException

{

Map<String, Object> systemInfo = new HashMap<>();

systemInfo.put("OS Name", System.getProperty("os.name"));

systemInfo.put("OS Version", System.getProperty("os.version"));

File disk = new File("/");

long totalSpace = disk.getTotalSpace() / (1024 \* 1024 \* 1024);

long freeSpace = disk.getFreeSpace() / (1024 \* 1024 \* 1024);

systemInfo.put("Total Disk Size (GB)", totalSpace);

systemInfo.put("Available Disk Size (GB)", freeSpace);

Runtime runtime = Runtime.getRuntime();

long totalMemory = runtime.totalMemory() / (1024 \* 1024);

long freeMemory = runtime.freeMemory() / (1024 \* 1024);

long usedMemory = totalMemory - freeMemory;

systemInfo.put("Total Memory (MB)", totalMemory);

systemInfo.put("Used Memory (MB)", usedMemory);

systemInfo.put("Free Memory (MB)", freeMemory);

return systemInfo;

}

}

**# Server.java :-**

package Assignment4.Question3.Server;

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class Server

{

public static void main(String[] args)

{

try

{

SystemInfo systemInfo = new SystemInfoClass();

LocateRegistry.createRegistry(1099);

Naming.bind("sys", systemInfo);

System.out.println("System Info RMI Server is running...");

}

catch (Exception e)

{

System.out.println("Server Error: " + e.getMessage());

e.printStackTrace();

}

}

}

**# Output :-**

System Info RMI Server is running...

**# Client.java :-**

package Assignment4.Question3.Client;

import java.rmi.Naming;

import java.util.Map;

import Assignment4.Question3.Server.SystemInfo;

public class Client

{

public static void main(String[] args)

{

try

{

SystemInfo systemInfo = (SystemInfo) Naming.lookup("rmi://localhost:1099/sys");

Map<String, Object> info = systemInfo.getSystemInfo();

System.out.println("System Information:");

info.forEach((key, value) -> System.out.println(key + ": " + value));

}

catch (Exception e)

{

System.out.println("Client Error: " + e.getMessage());

e.printStackTrace();

}

}

}

**# Output :-**

System Information:

OS Name: Windows 8.1

OS Version: 6.3

Available Disk Size (GB): 183

Total Memory (MB): 15

Free Memory (MB): 12

Total Disk Size (GB): 424

Used Memory (MB): 3