**Lab 38**

Write a program to illustrate:

a) Different ways of creating URL objects and URL parsing

b) Reading Directly from URL

c) Reading from a URL Connection.

d) InetAddress Class.

The **URLConnection** class in Java is a high-level class that provides methods to communicate with a remote server over the network. It is a part of the java.net package and is used to implement various network-related functionalities in Java. It provides a simple interface to create and manage network connections, send and receive data, and handle responses.

The **InetAddress** class in Java is used to represent an Internet Protocol (IP) address. It provides methods to get the host name and IP address of a computer on the network. This class is part of the java.net package and is used to implement network-related functionalities in Java.

**Code:**

import java.net.\*;

import java.io.\*;

public class q38 {

public static void main(String[] args) throws MalformedURLException, URISyntaxException, Exception {

URL url1 = new URL("https://www.w3schools.com/java/default.asp");

URI uri = new URI("https://www.geeksforgeeks.org/url-class-java-examples/");

URL url = uri.toURL();

System.out.println("Using URL\n" + url1.toString());

System.out.println("Protocol: " + url1.getProtocol());

System.out.println("Hostname: " + url1.getHost());

System.out.println("Default Port: " + url1.getDefaultPort());

System.out.println("\nUsing URI\n" + url.toString());

System.out.println("Path: " + url.getPath());

System.out.println("Protocol: " + url.getProtocol());

System.out.println("\nReading Directly from URL");

HttpURLConnection conn = (HttpURLConnection) url1.openConnection();

conn.setRequestMethod("GET");

BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));

String inputLine;

StringBuilder content = new StringBuilder();

while ((inputLine = in.readLine()) != null) {

content.append(inputLine);

}

in.close();

System.out.println(content.toString());

System.out.println("\nReading from a URLConnection");

HttpURLConnection conn1 = (HttpURLConnection) url1.openConnection();

conn1.setRequestMethod("GET");

conn1.setConnectTimeout(5000);

conn1.setReadTimeout(5000);

int responseCode = conn1.getResponseCode();

if (responseCode == 200) {

BufferedReader in1 = new BufferedReader(new InputStreamReader(conn1.getInputStream()));

String inputLine1;

StringBuilder content1 = new StringBuilder();

while ((inputLine1 = in1.readLine()) != null) {

content1.append(inputLine1);

}

in1.close();

System.out.println(content1.toString());

} else {

System.out.println("Failed to get webpage: " + responseCode);

}

System.out.println("\nDemostration of InetAddress class");

String host = "www.w3schools.com";

InetAddress addr = InetAddress.getByName(host);

System.out.println("Host name: " + addr.getHostName());

System.out.println("IP address: " + addr.getHostAddress());

}

}

**Lab 39**

Write Java programs using TCP Sockets to demonstrate two way communication between a simple server and a client.

TCP (Transmission Control Protocol) sockets are a type of network socket that allows for reliable, bidirectional communication between two endpoints over a network. In Java, TCP sockets are implemented using the java.net.Socket and java.net.ServerSocket classes.

**Code:**

**TCPClient.java**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class TCPClient {

public static void main(String[] args) throws IOException {

try {

Socket s = new Socket("localhost", 6970);

Scanner ins = new Scanner(s.getInputStream());

Scanner scanner = new Scanner(System.in);

PrintWriter out = new PrintWriter(s.getOutputStream(), true);

System.out.print("Enter an integer: ");

int x = scanner.nextInt();

out.println(x);

int receive = ins.nextInt();

System.out.println("Sent Data: " + x);

System.out.println("Received Data: " + receive);

scanner.close();

ins.close();

out.close();

s.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

**TCPServer.java**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class TCPServer {

public static void main(String[] args) throws IOException {

ServerSocket ss = new ServerSocket(6970);

Socket clientSocket = ss.accept();

Scanner ins = new Scanner(clientSocket.getInputStream());

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

int num = ins.nextInt();

int twi = num \* 2;

out.println(twi);

ins.close();

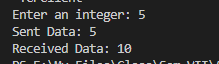
out.close();

ss.close();

}

}

Output:



**Lab 40**

Write Java programs using DatagramSoket and DatagramPacket (UDP Sockets) to demonstrate client and server communication.

UDP (User Datagram Protocol) is a connectionless protocol for communication over IP networks, where data is sent in the form of datagrams. In Java, UDP sockets are implemented using the java.net.DatagramSocket and java.net.DatagramPacket classes

**UDPServer.java**

import java.net.\*;

import java.io.\*;

public class UDPServer {

public static void main(String[] args) throws IOException {

DatagramSocket socket = new DatagramSocket(6969);

System.out.println("Server Listening ");

byte[] bytes = new byte[256];

DatagramPacket packet = new DatagramPacket(bytes, bytes.length);

socket.receive(packet);

InetAddress address = packet.getAddress();

int port = packet.getPort();

String s = "Hello Client From Server";

bytes = s.getBytes();

packet = new DatagramPacket(bytes, bytes.length, address, port);

socket.send(packet);

socket.close();

}

}

**UDPClient.java**

import java.net.\*;

import java.io.\*;

public class UDPClient {

public static void main(String[] args) throws IOException {

DatagramSocket socket = new DatagramSocket();

byte[] bytes = new byte[256];

InetAddress address = InetAddress.getByName("localhost");

DatagramPacket packet = new DatagramPacket(bytes, bytes.length, address, 6969); // Create Empty Packet

socket.send(packet);

socket.receive(packet);

String received = new String(packet.getData());

System.out.println("Server Sent: " + received);

socket.close();

}

}

**Output:**



**Lab 41**

Write a java program to demonstrate a) Sending simple mail b) Sending mail with attachment

c) Receiving mail

**a)Sending simple mail**

package network;

import javax.mail.Authenticator;

import javax.mail.PasswordAuthentication;

import java.util.Properties;

import javax.mail.Message;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

public class MailDemo {

public static void main(String[] args) {

String senderId="sender @gmail.com";

String senderPass="senderpass";

String host="smtp.gmail.com";

String port="587";

Properties prop=new Properties();

prop.put("mail.smtp.ssl.protocols", "TLSv1.2");

prop.put("mail.smtp.starttls.required", "true");

prop.put("mail.smtp.auth", "true");

prop.put("mail.smtp.starttls.enable", "true");

prop.put("mail.smtp.host", host);

prop.put("mail.smtp.port", port);

prop.put("mail.smtp.user", senderId);

prop.put("mail.smtp.password", senderPass);

Session session=Session.getInstance(prop,new Authenticator(){

@Override

protected PasswordAuthentication getPasswordAuthentication(){

return new PasswordAuthentication(senderId,senderPass);

}

});

try{

MimeMessage message=new MimeMessage(session);

message.setFrom(new InternetAddress(senderId));

message.addRecipient(Message.RecipientType.TO,new InternetAddress("email@gmail.com"));

message.setSubject("Test Email for Java Lab");

message.setText("hello ");

Transport.send(message);

System.out.println("Message Sent");

}catch(Exception ex){

ex.printStackTrace();

}

}

}

**b)Sending mail with attachment**

package javalab4;

import javax.mail.Authenticator;

import javax.mail.PasswordAuthentication;

import java.util.Properties;

import javax.mail.Message;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

import javax.activation.\*;

public class Q41\_sendAttachment {

public static void main(String[] args) {

String senderId = "sender@gmail.com";

String senderPass = "senderpass";

String host = "smtp.gmail.com";

String port = "587";

Properties prop = new Properties();

prop.put("mail.smtp.ssl.protocols", "TLSv1.2");

prop.put("mail.smtp.starttls.required", "true");

prop.put("mail.smtp.auth", "true");

prop.put("mail.smtp.starttls.enable", "true");

prop.put("mail.smtp.host", host);

prop.put("mail.smtp.port", port);

prop.put("mail.smtp.user", senderId);

prop.put("mail.smtp.password", senderPass);

Session session = Session.getInstance(prop, new Authenticator() {

@Override

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(senderId, senderPass);

}

});

try {

MimeMessage message = new MimeMessage(session);

message.setFrom(new InternetAddress(senderId));

message.addRecipient(Message.RecipientType.TO, new InternetAddress("receive@gmail.com"));

message.setSubject("Test Email for Java Lab");

message.setText("hello ");

String filename = "Path\\nepal.png";

DataSource source = new FileDataSource(filename);

message.setDataHandler(new DataHandler(source));

message.setFileName(filename);

Transport.send(message);

System.out.println("Message Sent");

} catch (Exception ex) {

ex.printStackTrace();

}

}

}

**c)Receiving mail**

package javalab4;

import java.util.Properties;

import javax.mail.Session;

import javax.mail.Folder;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.NoSuchProviderException;

import javax.mail.Store;

public class Q41\_receivingMail {

public static void main(String[] args) {

String userId = " user@gmail.com ";

String userPass = "userpass";

String host = "pop.gmail.com";

try {

Properties prop = new Properties();

prop.put("mail.pop3s.ssl.protocols", "TLSv1.2");

prop.put("mail.store.protocol", "pop3s");

prop.put("mail.pop3.host", host);

prop.put("mail.pop3.port", "995");

prop.put("mail.pop3.starttls.enable", "true");

Session session = Session.getDefaultInstance(prop);

Store st = session.getStore("pop3s");

st.connect(host, userId, userPass);

Folder emailFolder = st.getFolder("INBOX");

emailFolder.open(Folder.READ\_ONLY);

Message[] messages = emailFolder.getMessages();

int j = ((messages.length) - 1);

Message message = messages[j];

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

System.out.println("Email Number " + (j + 1));

System.out.println("Subject: " + message.getSubject());

System.out.println("From: " + message.getFrom()[0]);

System.out.println("Size: " + message.getSize());

emailFolder.close(false);

st.close();

} catch (NoSuchProviderException e) {

System.out.println(e);

} catch (MessagingException e) {

System.out.println(e);

}

}

**Lab 42**

Write a servlet program to display following

⎯ "Java is one of the powerful programming languages!" 10 times

⎯ Current date and time, Your Name/Roll No/Section

A servlet is a Java program that extends the capabilities of a server. It is a server-side technology used to extend the functionality of a web server by processing requests and generating dynamic content. Servlets are used in conjunction with JavaServer Pages (JSPs) to create Java web applications.

**Code:**

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Date;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class Q1 extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String message = "Java is one of the powerful programming languages!";

response.setContentType("text/html");

PrintWriter out = response.getWriter();

for (int i = 0; i < 10; i++) {

out.println("<h3><center>" + message + "</center></h3>");

}

Date today = new Date();

out.println("<h2><center>Today Date is</center></h2>");

out.println("<b><center>" + today + "</b><br><br>");

out.println("<b><center>Name: Shiash Shresha</center></b>");

out.println("<b><center>Roll No: 17 </center></b>");

out.println("<b><center>Section: A </center></b>");

}

}

**Lab 43**

Create a web application using servlet to take principal, time and rate values from a HTML form and display the simple interest.

**Code:**

**SimpleInterest.html**

<html>

<head>

<title>Simple Interest</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<form action="Q42" method="POST">

Enter Principle:<input type="number" name="principal"><br><br>

Enter Time:<input type="number" name="time"><br><br>

Enter rate:<input type="number" name="rate"><br><br>

<input type="submit" value="calculate">

</form>

</body>

</html>

**Q42.java**

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Date;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class Q42 extends HttpServlet {

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

float p = Float.parseFloat(request.getParameter("principal"));

float t=Float.parseFloat(request.getParameter("time"));

float r=Float.parseFloat(request.getParameter("rate"));

float si=(p\*t\*r)/100;

out.println("<html><body>");

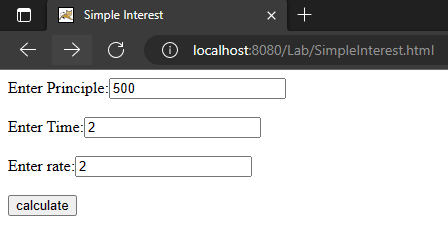
out.println("Simple Interest is "+si);

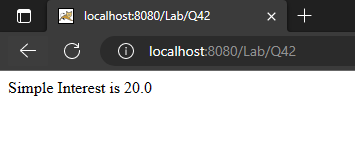
out.println("</body></html>");

}

}

**Output:**





**Lab 44**

Create a web application showing CRUD operation using Servlet and JDBC. Create appropriate UI using HTML and CSS. Use Relational Database in MySQL Server.

**index.html**

<html>

<head>

<title>CRUD operation</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>

body {

background-color:whitesmoke;

}

</style>

</head>

<body><center>

<h1>Add Student's Data</h1>

<form action="saveServlet" method="post">

<table style="border-collapse: separate; border-spacing: 25px; background-color: lightblue;border-radius: 20px;box-shadow: 10px 10px 5px #999;" >

<tr><td>Name:</td><td><input type="text" name="name"/></td></tr>

<tr><td>Address:</td><td><input type="text" name="address"/></td></tr>

<tr><td>Phone No:</td><td><input type="text" name="phone"/></td></tr><br>

<tr><td colspan="2" style="text-align: center;"><input type="submit" value="Save Data" style="padding: 8px 20px;box-shadow: 2px 2px 5px #999;"></td></tr>

</table> </form> <br/>

<form action="viewServlet" method="GET">

<input type="submit" value="View Data" style="padding: 8px 20px;box-shadow: 2px 2px 5px #999;">

</form>

</center> </body>

</html>

**Student.java**

package servletDemo;

public class student {

private int id;

private String name, address, phone;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

}

**studentDatabase.java**

package servletDemo;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

public class studentDatabase {

static String driver = "com.mysql.cj.jdbc.Driver";

static String dbURL = "jdbc:mysql://localhost:3306/ksdata";

static String dbUser = "root";

static String dbPassword = "";

public static Connection getConnection() {

Connection con = null;

try {

Class.forName(driver);

con = DriverManager.getConnection(dbURL, dbUser, dbPassword);

} catch (Exception e) {

System.out.println(e);

}

return con;

}

public static int save(student s) {

int res = 0;

try {

Connection con = studentDatabase.getConnection();

String myQuery = "insert into student(name,address,phone) values(?,?,?)";

PreparedStatement ps = con.prepareStatement(myQuery);

ps.setString(1, s.getName());

ps.setString(2, s.getAddress());

ps.setString(3, s.getPhone());

res = ps.executeUpdate();

System.out.println("Data inserted... " + res + " row(s)affected!");

con.close();

} catch (Exception ex) {

System.out.println(ex);

}

return res;

}

public static int update(student s) {

int res = 0;

try {

Connection con = studentDatabase.getConnection();

String myQuery = "update student set name=?,address=?,phone=? where id=?";

PreparedStatement ps = con.prepareStatement(myQuery);

ps.setString(1, s.getName());

ps.setString(2, s.getAddress());

ps.setString(3, s.getPhone());

ps.setInt(4, s.getId());

res = ps.executeUpdate();

System.out.println("Data Updated... " + res + " row(s)affected!");

con.close();

} catch (Exception ex) {

System.out.println(ex);

}

return res;

}

public static int delete(int id) {

int res = 0;

try {

Connection con = studentDatabase.getConnection();

String myQuery = "delete from student where id=?";

PreparedStatement ps = con.prepareStatement(myQuery);

ps.setInt(1, id);

res = ps.executeUpdate();

System.out.println("Data deleted... " + res + " row(s)affected!");

con.close();

} catch (Exception ex) {

System.out.println(ex);

}

return res;

}

public static student getStudentById(int id) {

student s = new student();

try {

Connection con = studentDatabase.getConnection();

String myQuery = "select \* from student where id=?";

PreparedStatement ps = con.prepareStatement(myQuery);

ps.setInt(1, id);

ResultSet rs = ps.executeQuery();

if (rs.next()) {

s.setId(rs.getInt(1));

s.setName(rs.getString(2));

s.setAddress(rs.getString(3));

s.setPhone(rs.getString(4));

}

con.close();

} catch (Exception ex) {

System.out.println(ex);

}

return s;

}

public static List<student> getAllStudent() {

List<student> list = new ArrayList<student>();

try {

Connection con = studentDatabase.getConnection();

String myQuery = "select \* from student";

PreparedStatement ps = con.prepareStatement(myQuery);

ResultSet rs = ps.executeQuery();

while (rs.next()) {

student s = new student();

s.setId(rs.getInt(1));

s.setName(rs.getString(2));

s.setAddress(rs.getString(3));

s.setPhone(rs.getString(4));

list.add(s);

}

} catch (Exception ex) {

System.out.println(ex);

}

return list;

}

}

**saveServlet.java**

package servletDemo;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class saveServlet extends HttpServlet {

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String name = request.getParameter("name");

String address = request.getParameter("address");

String phone = request.getParameter("phone");

if (name == null || name.trim().isEmpty() || address == null || address.trim().isEmpty() || phone == null || phone.trim().isEmpty()) {

out.println("<p style='color:red;text-align:center;'>Please enter all fields!</p>");

request.getRequestDispatcher("index.html").include(request, response);

out.close();

} else {

student s = new student();

s.setName(name);

s.setAddress(address);

s.setPhone(phone);

int res = studentDatabase.save(s);

if (res > 0) {

out.println("<p style='color:green;text-align:center;'>Record saved successfully!</p>");

request.getRequestDispatcher("index.html").include(request, response);

} else {

out.println("Sorry! unable to save record");

}

out.close();

}

}

}

**viewServlet.java**

package servletDemo;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class viewServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.println("<html><body style='background-color:whitesmoke;'><center>");

out.println("<h1>Student's Data:</h1>");

List<student> list = studentDatabase.getAllStudent();

out.print("<table border='5' width='100%' cellspacing='0' cellpadding='5px'");

out.println("<tr><th>SN</th><th>Name</th><th>Address</th><th>Phone Number</th><th>Edit</th><th>Delete</th></tr>");

int sn = 0;

for (student s : list) {

sn++;

out.print("<tr>");

out.print("<td align='center'>" + sn + "</td>");

out.print("<td align='center'>" + s.getName() + "</td>");

out.print("<td align='center'>" + s.getAddress() + "</td>");

out.print("<td align='center'>" + s.getPhone() + "</td>");

out.print("<td align='center'><a href='editServlet?id=" + s.getId() + "'>edit</a></td>");

out.print("<td align='center'><a href='deleteServlet?id=" + s.getId() + "'>delete</a></td>");

out.print("</tr>");

}

out.println("</table>");

out.println("<br>");

out.println("<form action='index.html' method='get'>");

out.println("<input type='submit' value='Add Data' style='padding: 8px 20px;box-shadow: 2px 2px 5px #999;'>");

out.println("</form>");

out.println("</center></body></html>");

out.close();

}

}

editServlet.java

package servletDemo;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class editServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.println("<html><body style='background-color:whitesmoke;'><center>");

out.println("<h1><center>Update Student's data:</center></h1>");

int id = Integer.parseInt(request.getParameter("id"));

student s = studentDatabase.getStudentById(id);

String error = request.getParameter("error");

if (error != null && !error.isEmpty()) {

out.println("<p style='color: red;'>" + error + "</p>");

}

out.print("<form action='updateServlet' method='post'>");

out.print("<table style=\"border-collapse: separate; border-spacing: 25px; background-color: lightblue;border-radius: 20px;box-shadow: 10px 10px 5px #999;\">");

out.print("<tr><td></td><td><input type='hidden' name='id' value='" + s.getId() + "'/></td></tr>");

out.print("<tr><td>Name:</td><td><input type='text' name='name' value='" + s.getName() + "'/></td></tr>");

out.print("<tr><td>Address:</td><td><input type='text' name='address' value='" + s.getAddress() + "'/></td></tr>");

out.print("<tr><td>Phone No:</td><td><input type='text' name='phone' value='" + s.getPhone() + "'/></td></tr>");

out.print("<tr><td colspan='2' style=\"text-align: center;\"><input type='submit' value=' Update ' style=\"padding: 8px 20px;box-shadow: 2px 2px 5px #999;\"></td></tr>");

out.print("</table>");

out.print("</form>");

out.println("</center></body></html>");

out.close();

}

}

**updateServlet.java**

package servletDemo;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class updateServlet extends HttpServlet {

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

int id = Integer.parseInt(request.getParameter("id"));

String name = request.getParameter("name");

String address = request.getParameter("address");

String phone = request.getParameter("phone");

if (name == null || name.trim().isEmpty() || address == null || address.trim().isEmpty() || phone == null || phone.trim().isEmpty()) {

response.sendRedirect("editServlet?id=" + id + "&error=Please%20enter%20all%20fields!");

out.close();

} else {

student s = new student();

s.setId(id);

s.setName(name);

s.setAddress(address);

s.setPhone(phone);

int res = studentDatabase.update(s);

if (res > 0) {

response.sendRedirect("viewServlet");

} else {

out.println("Sorry! unable to update record");

}

out.close();

}

}

}

**deleteServlet.java**

package servletDemo;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class deleteServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

int id = Integer.parseInt(request.getParameter("id"));

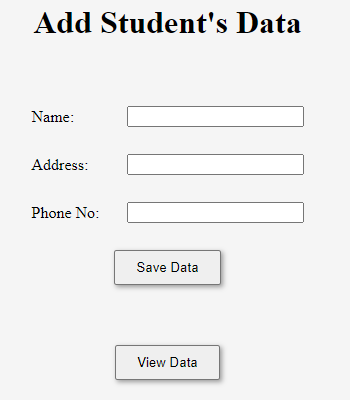
studentDatabase.delete(id);

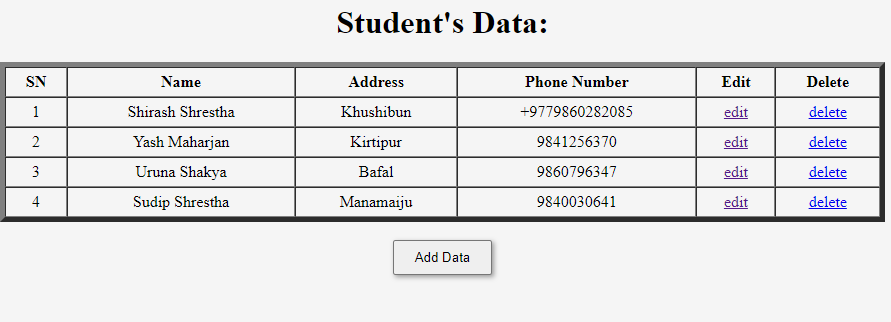
response.sendRedirect("viewServlet");

}

}

**Output:**





**Lab 45**

Write a servlet to demonstrate how

a) session is created, read and destroyed.

b) cookie is handled

In web applications, a **session** is a way to maintain state between a client and server over multiple requests. When a client makes a request to a server, the server can create a session and associate it with the client. The session allows the server to store information about the client, such as login credentials or shopping cart items, and retrieve it on subsequent requests.

**Code:**

**Session:**

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Date;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

public class Session extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

HttpSession session = request.getSession(true);

String sessionId = session.getId();

Date createTime = new Date(session.getCreationTime());

Date lastAccessTime = new Date(session.getLastAccessedTime());

session.setMaxInactiveInterval(300);

session.setAttribute("username", "User");

session.setAttribute("email", "user@email.com");

String username = (String) session.getAttribute("username");

String email = (String) session.getAttribute("email");

session.invalidate();

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.println("<html><body>");

out.println("<h1>Session Example</h1>");

out.println("<p>Session ID: " + sessionId + "</p>");

out.println("<p>Session Creation Time: " + createTime + "</p>");

out.println("<p>Last Accessed Time: " + lastAccessTime + "</p>");

out.println("<p>Username: " + username + "</p>");

out.println("<p>Email: " + email + "</p>");

out.println("</body></html>");

}

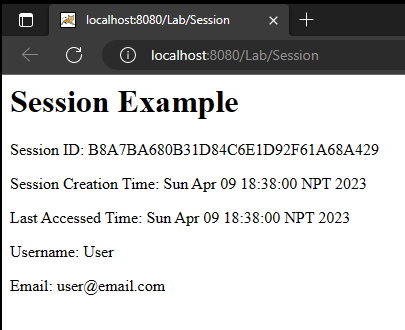
@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

}

}



**Cookie:**

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class CookieDemo extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

Cookie cookie = new Cookie("Username", "User");

response.addCookie(cookie);

out.println("<html><body>");

out.println("<h1>Cookie handling Example</h1>");

out.println("<p>Hello " + cookie.getValue()+"</p>");

out.println("</body></html>");

}

}



**Lab 46**

Create a simple web application to demonstrate login and logout features using servlet.

**Code:**

**Login.html**

<html>

<head>

<title>Login</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<form action="loginServlet" method="POST">

username:<input type="text" name="uname"><br><br>

password:<input type="password" name="password"><br><br>

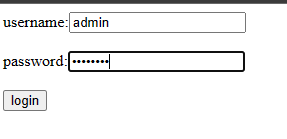
<input type="submit" value="login">

</form>

</body>

</html>

**Output:**



**loginServlet.java**

package servletDemo;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

public class loginServlet extends HttpServlet {

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String username = request.getParameter("uname");

String pass = request.getParameter("password");

if (username.equals("admin") && pass.equals("password")) {

HttpSession session = request.getSession();

session.setAttribute("username", username);

RequestDispatcher dispatcher = request.getRequestDispatcher("homeServlet");

dispatcher.forward(request, response);

} else {

out.println("<script type='text/javascript'>");

out.println("alert('Invalid username or password');");

out.println("location='login.html';");

out.println("</script>");

}

out.close();

}

}

**homeServlet.java**

package servletDemo;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

public class homeServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

HttpSession session = request.getSession();

String username = (String) session.getAttribute("username");

response.setContentType("text/html");

PrintWriter out = response.getWriter();

if (username != null) {

out.println("<html><body>");

out.println("<h1>Hello, " + username + "</h1>");

out.println("<form action=\"logoutServlet\" method=\"post\">");

out.println("<input type=\"submit\" value=\"Logout\">");

out.println("</form>");

out.println("</body></html>");

} else {

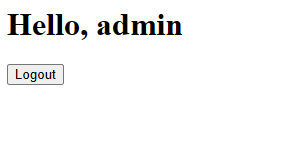
response.sendRedirect("login.html");

}

}

}

**Output:**



**logoutServlet.java**

package servletDemo;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

public class logoutServlet extends HttpServlet {

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

HttpSession session = request.getSession(false);

if (session != null) {

session.invalidate();

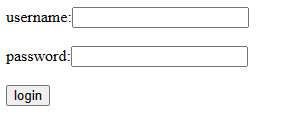
}

response.sendRedirect("login.html");

}

}

Output:



**Lab 47**

Write a JSP program to demonstrate ⎯ JSP Directives ⎯ JSP Declaration ⎯ JPS Expression ⎯ JSP Scriplet ⎯ JSP Comment

**Code:**

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>JSP Example</title>

</head>

<body>

<!-- JSP Comment -->

<%-- This is a JSP comment --%>

<!-- JSP Directive -->

<%@ page import="java.util.\*" %>

<!-- JSP Declaration -->

<%!

String message = "Hello, World!";

%>

<!-- JSP Expression -->

<p><%= message %></p>

<!-- JSP Scriptlet -->

<% for (int i = 1; i <= 5; i++) { %>

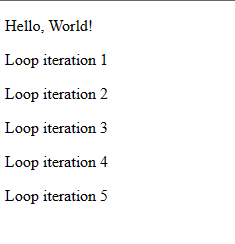
<p>Loop iteration <%= i %></p>

<% } %>

</body>

</html>

**Output:**



**Lab 48**

Write a JSP program to

a) take some values from a form and display them.

b) demonstrate scope of implicit objects

a)

**Code:**

form.html

<html>

<head>

<title>JSP demo</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<form action = "formJSP.jsp" method = "POST">

First Name: <input type = "text" name = "fname"><br>

Last Name: <input type = "text" name = "lname" ><br>

Section: <input type = "text" name = "sec" ><br>

Roll no: <input type = "text" name = "roll" ><br>

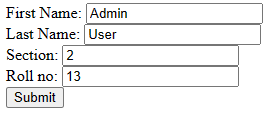
<input type = "submit" value = "Submit" >

</form>

</body>

</html>

**Output:**



**formJSP.jsp**

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

</head>

<body>

<p><b>First Name:</b><%= request.getParameter("fname")%></p>

<p><b>Last Name:</b> <%= request.getParameter("lname")%> </p>

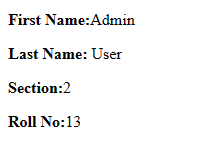
<p><b>Section:</b><%= request.getParameter("sec")%> </p>

<p><b>Roll No:</b><%= request.getParameter("roll")%></p>

</body>

</html>

**Output:**



b)

**form\_b.html**

<html>

<head>

<title>JSP demo</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<form action = "Q48\_b.jsp" method = "POST">

First Name: <input type = "text" name = "fname"><br>

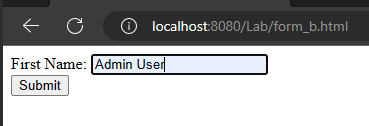
<input type = "submit" value = "Submit" >

</form>

</body>

</html>

Output:



Q48\_b.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Implicit object</title>

</head>

<body>

<%

out.print("For out implicit object:<br>");

out.print("Today Date is " + java.util.Calendar.getInstance().getTime());

%><br><br>

<%

String name = request.getParameter("fname");

out.print("For request implicit object: <br>Welcome " + name);

%><br><br>

<%

session.setAttribute("user", name);

String sessionName = (String) session.getAttribute("user");

out.print("For Session implicit object: <br> Welcome," + sessionName);

%><br><br>

<%

pageContext.setAttribute("user", name, PageContext.SESSION\_SCOPE);

String name1 = (String) pageContext.getAttribute("user", PageContext.SESSION\_SCOPE);

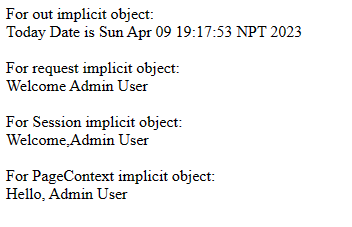
out.print("For PageContext implicit object: <br> Hello, " + name);

%>

</body>

</html>

Output:



**Lab 49**

Create a web application showing CRUD operation using JSP and JDBC. Create appropriate UI using HTML and CSS. Use Relational Database in MySQL Server.

Source code:

addForm.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP CRUD Operation</title>

<style>

body {

background-color:whitesmoke;

}

</style>

</head>

<body><center>

<h1>Add Student's Data</h1>

<form action="save.jsp" method="post">

<table style="border-collapse: separate; border-spacing: 25px; background-color: lightblue;border-radius: 20px;box-shadow: 10px 10px 5px #999;" >

<tr><td>Name:</td><td><input type="text" name="name"/></td></tr>

<tr><td>Address:</td><td><input type="text" name="address"/></td></tr>

<tr><td>Phone No:</td><td><input type="text" name="phone"/></td></tr><br>

<tr><td colspan="2" style="text-align: center;"><input type="submit" value="Save Data" style="padding: 8px 20px;box-shadow: 2px 2px 5px #999;"></td></tr>

</table>

</form>

<br/>

<form action="view.jsp" method="POST">

<input type="submit" value="View Data" style="padding: 8px 20px;box-shadow: 2px 2px 5px #999;">

</form>

</center> </body>

</html>

save.jsp

<%@page import="jspDemo.studentDatabase"%>

<%@page import="jspDemo.student"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

</head>

<body>

<%

String name = request.getParameter("name");

String address = request.getParameter("address");

String phone = request.getParameter("phone");

if (name == null || name.trim().isEmpty() || address == null || address.trim().isEmpty() || phone == null || phone.trim().isEmpty()) {

%>

<p style='color:red;text-align:center;'>Please enter all fields!</p>

<% request.getRequestDispatcher("addForm.jsp").include(request, response);

} else {

student s = new student();

s.setName(name);

s.setAddress(address);

s.setPhone(phone);

int res = studentDatabase.save(s);

if (res > 0) {

%>

<p style='color:green;text-align:center;'>Record saved successfully!</p>

<% request.getRequestDispatcher("addForm.jsp").include(request, response);

} else {

%>

<p style='color:red;text-align:center;'>Sorry! unable to save record</p>

<% request.getRequestDispatcher("addForm.jsp").include(request, response);

}

}

%>

</body>

</html>

view.jsp

<%@page import="jspDemo.studentDatabase"%>

<%@page import="jspDemo.student"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@page import="java.util.List"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

</head>

<body style="background-color:whitesmoke;"><center>

<h1>Student's Data:</h1>

<table border="5" width="100%" cellspacing="0" cellpadding="5px">

<tr>

<th>SN</th>

<th>Name</th>

<th>Address</th>

<th>Phone Number</th>

<th>Edit</th>

<th>Delete</th>

</tr>

<%

List<student> list = studentDatabase.getAllStudent();

int sn = 0;

for (student s : list) {

sn++;

%>

<tr>

<td align="center"><%= sn%></td>

<td align="center"><%= s.getName()%></td>

<td align="center"><%= s.getAddress()%></td>

<td align="center"><%= s.getPhone()%></td>

<td align="center"><a href="edit.jsp?id=<%= s.getId()%>">Edit</a></td>

<td align="center"><a href="delete.jsp?id=<%= s.getId()%>">Delete</a></td>

</tr>

<%

}

%>

</table>

<br>

<form action="addForm.jsp" method="get">

<input type="submit" value="Add Data" style="padding: 8px 20px;box-shadow: 2px 2px 5px #999;">

</form>

</center></body>

</html>

student.java

package jspDemo;

public class student {

private int id;

private String name, address, phone;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

}

studentDatabase.java

package jspDemo;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

public class studentDatabase {

static String driver = "com.mysql.cj.jdbc.Driver";

static String dbURL = "jdbc:mysql://localhost:3306/don ";

static String dbUser = "root";

static String dbPassword = "";

public static Connection getConnection() {

Connection con = null;

try {

Class.forName(driver);

con = DriverManager.getConnection(dbURL, dbUser, dbPassword);

} catch (Exception e) {

System.out.println(e);

}

return con;

}

public static int save(student s) {

int res = 0;

try {

Connection con = studentDatabase.getConnection();

String myQuery = "insert into student(name,address,phone) values(?,?,?)";

PreparedStatement ps = con.prepareStatement(myQuery);

ps.setString(1, s.getName());

ps.setString(2, s.getAddress());

ps.setString(3, s.getPhone());

res = ps.executeUpdate();

System.out.println("Data inserted... " + res + " row(s)affected!");

con.close();

} catch (Exception ex) {

System.out.println(ex);

}

return res;

}

public static int update(student s) {

int res = 0;

try {

Connection con = studentDatabase.getConnection();

String myQuery = "update student set name=?,address=?,phone=? where id=?";

PreparedStatement ps = con.prepareStatement(myQuery);

ps.setString(1, s.getName());

ps.setString(2, s.getAddress());

ps.setString(3, s.getPhone());

ps.setInt(4, s.getId());

res = ps.executeUpdate();

System.out.println("Data Updated... " + res + " row(s)affected!");

con.close();

} catch (Exception ex) {

System.out.println(ex);

}

return res;

}

public static int delete(int id) {

int res = 0;

try {

Connection con = studentDatabase.getConnection();

String myQuery = "delete from student where id=?";

PreparedStatement ps = con.prepareStatement(myQuery);

ps.setInt(1, id);

res = ps.executeUpdate();

System.out.println("Data deleted... " + res + " row(s)affected!");

con.close();

} catch (Exception ex) {

System.out.println(ex);

}

return res;

}

public static student getStudentById(int id) {

student s = new student();

try {

Connection con = studentDatabase.getConnection();

String myQuery = "select \* from student where id=?";

PreparedStatement ps = con.prepareStatement(myQuery);

ps.setInt(1, id);

ResultSet rs = ps.executeQuery();

if (rs.next()) {

s.setId(rs.getInt(1));

s.setName(rs.getString(2));

s.setAddress(rs.getString(3));

s.setPhone(rs.getString(4));

}

con.close();

} catch (Exception ex) {

System.out.println(ex);

}

return s;

}

public static List<student> getAllStudent() {

List<student> list = new ArrayList<student>();

try {

Connection con = studentDatabase.getConnection();

String myQuery = "select \* from student";

PreparedStatement ps = con.prepareStatement(myQuery);

ResultSet rs = ps.executeQuery();

while (rs.next()) {

student s = new student();

s.setId(rs.getInt(1));

s.setName(rs.getString(2));

s.setAddress(rs.getString(3));

s.setPhone(rs.getString(4));

list.add(s);

}

} catch (Exception ex) {

System.out.println(ex);

}

return list;

}

}

edit.jsp

<%@page import="jspDemo.studentDatabase"%>

<%@page import="jspDemo.student"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

</head>

<body style='background-color:whitesmoke;'><center>

<h1><center>Update Student's data:</center></h1>

<%

int id = Integer.parseInt(request.getParameter("id"));

student s = studentDatabase.getStudentById(id);

String error = request.getParameter("error");

if (error != null && !error.isEmpty()) {

out.println("<p style='color: red;'>" + error + "</p>");

}

%>

<form action="update.jsp" method="post">

<table style="border-collapse: separate; border-spacing: 25px; background-color: lightblue;border-radius: 20px;box-shadow: 10px 10px 5px #999;" >

<tr><td></td><td><input type="hidden" name="id" value="<%=s.getId()%>"/></td></tr>

<tr><td>Name:</td><td><input type="text" name="name" value="<%=s.getName()%>"/></td></tr>

<tr><td>Address:</td><td><input type="text" name="address" value="<%=s.getAddress()%>"/></td></tr>

<tr><td>Phone No:</td><td><input type="text" name="phone" value="<%=s.getPhone()%>"/></td></tr><br>

<tr><td colspan="2" style="text-align: center;"> <input type="submit" value=" Update " style="padding: 8px 20px;box-shadow: 2px 2px 5px #999;"></td></tr>

</table>

</form>

</center></body>

</html>

update.jsp

<%@page import="jspDemo.studentDatabase"%>

<%@page import="jspDemo.student"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

</head>

<body>

<h1>Hello World!</h1>

<%

int id = Integer.parseInt(request.getParameter("id"));

String name = request.getParameter("name");

String address = request.getParameter("address");

String phone = request.getParameter("phone");

if (name == null || name.trim().isEmpty() || address == null || address.trim().isEmpty() || phone == null || phone.trim().isEmpty()) {

response.sendRedirect("edit.jsp?id=" + id + "&error=Please%20enter%20all%20fields!");

} else {

student s = new student();

s.setId(id);

s.setName(name);

s.setAddress(address);

s.setPhone(phone);

int res = studentDatabase.update(s);

if (res > 0) {

response.sendRedirect("view.jsp");

} else {

%>

<p style='color:red;text-align:center;'>Sorry! unable to update record</p>

<%

}

}

%>

</body>

</html>

delete.jsp

<%@page import="jspDemo.studentDatabase"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

</head>

<body>

<h1>Hello World!</h1>

<%

int id = Integer.parseInt(request.getParameter("id"));

studentDatabase.delete(id);

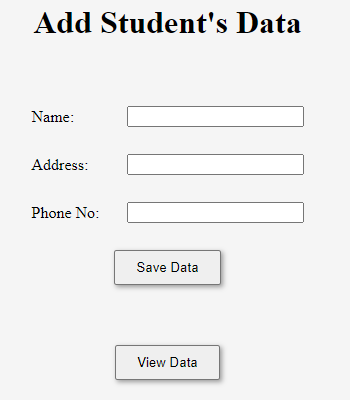
response.sendRedirect("view.jsp");

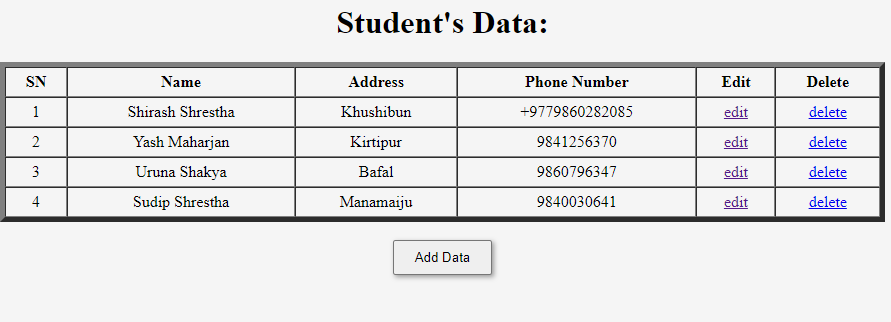
%>

</body>

</html>

Output:





**Lab 50**

Create a simple web application using Spring Framework.

Steps:

1. Create a java web application project.
2. After the creation of the project, edit the web.xml file inside webapp 🡪 WEB-INF🡪web.xml  
   Add the following codes inside the tag <web-app> </web-app>:  
   <servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>WEB-INF/spring-config.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<jsp-config>

<jsp-property-group>

<url-pattern>\*.jsp</url-pattern>

<page-encoding>UTF-8</page-encoding>

</jsp-property-group>

</jsp-config>

1. Create a new xml file named spring-config.xml in the folder WEB-INF and add the following codes:   
   <?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd">

<context:component-scan base-package="com.example.web" />

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/view/" />

<property name="suffix" value=".jsp" />

</bean>

</beans>

1. Edit the index,jsp to create form and add other jsp pages to display the desired pages:  
   index.jsp  
   <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>

<html>

<head>

<title>Welcome</title>

</head>

<body>

<h1>WELCOME TO SPRING WEB</h1>

<a href="/form">Form</a>

</body>

</html>  
  
display.jsp  
<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>Display</title>

</head>

<body>

<p>This is inside display jsp</p>

<h1>Hello, ${name}!</h1>

</body>

</html>  
  
form.jsp  
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>

<!DOCTYPE html>

<html>

<head>

<title>JSP - Hello World</title>

</head>

<body>

<h1><%="FORM"%></h1>

<form action="display" method="post">

Name: <input type="text" name="name"><br><br>

<input type="submit" value="Submit">

</form>

</body>

</html>  
  
helloPage.jsp  
<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>Welcome</title>

</head>

<body>

<h1>WELCOME TO SPRING WEB</h1>

</body>

</html>

1. Create a controller class to control the actions of the jsp pages:  
     
   DemoController.java  
   import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

@Controller

public class DemoController {

@GetMapping("/form")

public String form() {

return "form"; // /WEB-INF/view/form.jsp

}

@PostMapping("/display")

public String sayHello(@RequestParam("name") String name, ModelMap model) {

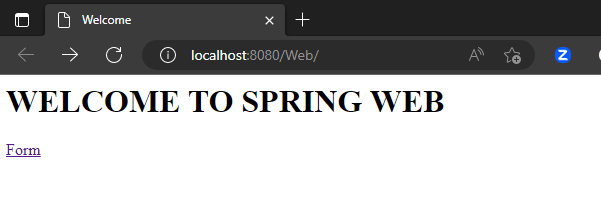
model.addAttribute("name",name);

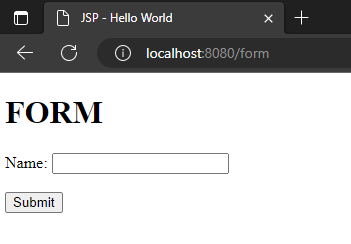
return "display";

}

}

1. Finally run the tomcat server to view the application developed in Spring Web.





**Lab 51**

Write a program using JavaFX to demonstrate a)FlowPane b)BorderPane c)HBox d) VBox e)GridPane

a)Flowpane

FlowPane layout pane organizes the nodes in a flow that are wrapped at the flowpane's boundary. The horizontal flowpane arranges the nodes in a row and wrap them according to the flowpane's width. The vertical flowpane arranges the nodes in a column and wrap them according to the flowpane's height.

**Source code:**

package javafxdemo;

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.layout.FlowPane;

import javafx.scene.text.Text;

import javafx.stage.Stage;

public class Q51 extends Application {

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("FlowPane Example");

FlowPane root = new FlowPane();

Text text = new Text();

root.setVgap(5);

root.setHgap(5);

root.setPrefWrapLength(200);

root.getChildren().add(new Button("Button1"));

root.getChildren().add(new Button("Button2"));

root.getChildren().add(new Button("Button3"));

root.getChildren().add(text);

Scene scene = new Scene(root, 300, 300);

primaryStage.setScene(scene);

primaryStage.show();

}

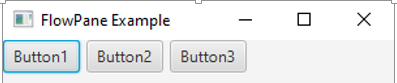
public static void main(String[] args) {

launch(args);

}

}

Output:



b)Boarderpane

BorderPane arranges the nodes at the left, right, centre, top and bottom of the screen. It is represented by javafx.scene.layout.BorderPane class. This class provides various methods like setRight(), setLeft(), setCenter(), setBottom() and setTop() which are used to set the position for the specified nodes.

package javafxdemo;

**Source code:**

import javafx.application.Application;

import static javafx.application.Application.launch;

import javafx.scene.Scene;

import javafx.scene.control.Label;

import javafx.scene.layout.BorderPane;

import javafx.stage.Stage;

public class Q51\_b extends Application {

@Override

public void start(Stage primaryStage) throws Exception {

primaryStage.setTitle("BoarderPane Example");

BorderPane BPane = new BorderPane();

BPane.setTop(new Label("This will be at the top"));

BPane.setLeft(new Label("This will be at the left"));

BPane.setRight(new Label("This will be at the Right"));

BPane.setCenter(new Label("This will be at the Centre"));

BPane.setBottom(new Label("This will be at the bottom"));

Scene scene = new Scene(BPane,400,400);

primaryStage.setScene(scene);

primaryStage.show();

}

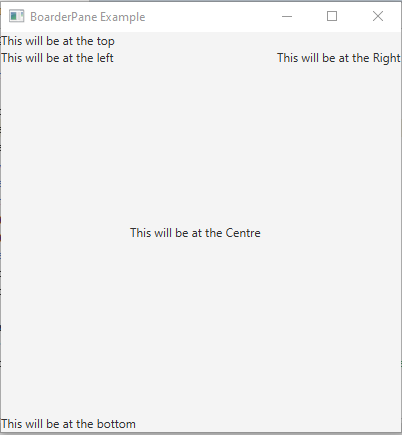
public static void main(String[] args) {

launch(args);

}

}

Output:



c)HBox

HBox layout pane arranges the nodes in a single row. It is represented by javafx.scene.layout.HBox class. We just need to instantiate HBox class in order to create HBox layout.

Source code:

package javafxdemo;

import javafx.application.Application;

import static javafx.application.Application.launch;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.layout.HBox;

import javafx.scene.text.Text;

import javafx.stage.Stage;

public class Q51\_c extends Application {

@Override

public void start(Stage primaryStage) throws Exception {

primaryStage.setTitle("HBox Example");

Text text = new Text();

Button btn1 = new Button("Button 1");

Button btn2 = new Button("Button 2");

HBox root = new HBox();

Scene scene = new Scene(root, 400, 400);

root.getChildren().addAll(btn1, btn2,text);

root.setSpacing(10);

primaryStage.setScene(scene);

primaryStage.show();

}

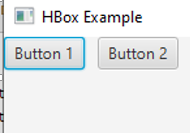
public static void main(String[] args) {

launch(args);

}

}

Output:



d)VBox

Instead of arranging the nodes in horizontal row, Vbox Layout Pane arranges the nodes in a single vertical column. It is represented by javafx.scene.layout.VBox class which provides all the methods to deal with the styling and the distance among the nodes.

Source code:

package javafxdemo;

import javafx.application.Application;

import static javafx.application.Application.launch;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.layout.VBox;

import javafx.scene.text.Text;

import javafx.stage.Stage;

public class Q51\_d extends Application {

@Override

public void start(Stage primaryStage) throws Exception {

primaryStage.setTitle("VBox Example");

Text text = new Text();

Button btn1 = new Button("Button 1");

Button btn2 = new Button("Button 2");

VBox root = new VBox();

Scene scene = new Scene(root, 200, 200);

root.getChildren().addAll(btn1, btn2,text);

root.setSpacing(10);

primaryStage.setScene(scene);

primaryStage.show();

}

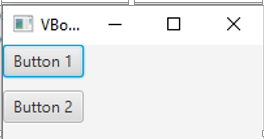
public static void main(String[] args) {

launch(args);

}

}

Output:



e)GridPane

GridPane Layout pane allows us to add the multiple nodes in multiple rows and columns. It is seen as a flexible grid of rows and columns where nodes can be placed in any cell of the grid. It is represented by javafx.scence.layout.GridPane class. We just need to instantiate this class to implement GridPane.

package javafxdemo;

import javafx.application.Application;

import static javafx.application.Application.launch;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import javafx.scene.text.Text;

import javafx.stage.Stage;

public class Q51\_e extends Application {

@Override

public void start(Stage primaryStage) throws Exception {

primaryStage.setTitle("GridPane Example");

Text text = new Text();

Label fname = new Label("First Name");

Label lname = new Label("Last Name");

TextField tf1 = new TextField();

TextField tf2 = new TextField();

Button Submit = new Button("Submit");

GridPane root = new GridPane();

Scene scene = new Scene(root, 400, 400);

root.addRow(0, fname, tf1);

root.addRow(1, lname, tf2);

root.addRow(2, Submit);

root.addRow(3, text);

primaryStage.setScene(scene);

primaryStage.show();

}

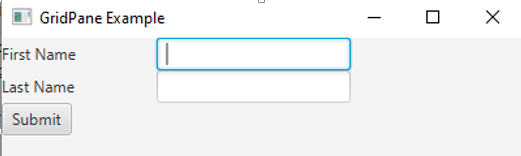
public static void main(String[] args) {

launch(args);

}

}

Output:



**Lab 52**

Create a JavaFX UI having following controls:a)Label b)TextField Button c) RadioButton d)CheckBox, e)Hyperlink, Menu, Tooltip, f)FileChooser.

Theory:

**label** is the component that is used to place any text information on the screen. It is mainly used to describe the purpose of the other components to the user.

**Text Field** is basically used to get the input from the user in the form of text. javafx.scene.control.TextField represents TextField.

**Radio Button** is used to provide various options to the user. The user can only choose one option among all. A radio button is either selected or deselected.

**Check Box** is used to provide more than one choices to the user. It can be used in a scenario where the user is prompted to select more than one option or the user wants to select multiple options.

In JavaFx, we can use **hyper-links** to refer the web pages. It is similar to anchor links in HTML.

JavaFX provides a **Menu class** to implement menus. Menu is the main component of a any application. In JavaFX, javafx.scene.control.Menu class provides all the methods to deal with menus.

**JavaFX Tool tip** is used to provide hint to the user about any component. It is mainly used to provide hints about the text fields or password fields being used in the application.

**JavaFX File chooser** enables users to browse the files from the file system. javafx.stage.FileChooser class represents FileChooser.

**Source Code:**

import javafx.application.Application;

import javafx.scene.\*;

import javafx.scene.control.\*;

import javafx.scene.layout.\*;

import javafx.scene.text.Text;

import javafx.stage.\*;

public class App extends Application {

@Override

public void start(Stage primaryStage) throws Exception {

primaryStage.setTitle("JavaFX UI Example");

Text text = new Text();

Text text1 = new Text("Select Faculty:");

Text text2 = new Text("Select Food:");

Text text3 = new Text("Link Tree: ");

Label fname = new Label("First Name: ");

Label lname = new Label("Last Name: ");

TextField tf1 = new TextField();

TextField tf2 = new TextField();

GridPane gpane = new GridPane();

BorderPane bp = new BorderPane();

ToggleGroup group = new ToggleGroup();

RadioButton r1 = new RadioButton("CIST");

RadioButton r2 = new RadioButton("BBA");

RadioButton r3 = new RadioButton("BBS");

r1.setToggleGroup(group);

r2.setToggleGroup(group);

r3.setToggleGroup(group);

CheckBox c1 = new CheckBox("MoMo");

CheckBox c2 = new CheckBox("Pizza");

CheckBox c3 = new CheckBox("Salad");

CheckBox c4 = new CheckBox("Sandwich");

Hyperlink hp = new Hyperlink("https://link.tree");

MenuBar mbar = new MenuBar();

Menu m1 = new Menu("File");

Menu m2 = new Menu("Edit");

MenuItem mi1 = new MenuItem("New Project");

MenuItem mi2 = new MenuItem("Open Project");

MenuItem mi3 = new MenuItem("save");

MenuItem mi4 = new MenuItem("cut");

MenuItem mi5 = new MenuItem("copy");

MenuItem mi6 = new MenuItem("paste");

m1.getItems().addAll(mi1, mi2, mi3);

m2.getItems().addAll(mi4, mi5, mi6);

mbar.getMenus().addAll(m1, m2);

bp.setTop(mbar);

bp.setLeft(gpane);

Tooltip tool = new Tooltip("Enter first\_name");

tf1.setTooltip(tool);

Tooltip tool1 = new Tooltip("Enter last\_name");

tf2.setTooltip(tool1);

Button btn = new Button("Choose File");

Button btn1 = new Button("Save");

btn.setOnAction(e -> {

FileChooser file = new FileChooser();

file.setTitle("Select File");

file.showOpenDialog(primaryStage);

});

gpane.setVgap(10);

gpane.addRow(1, lname, tf1);

gpane.addRow(2, fname, tf2);

gpane.addRow(3, text1);

gpane.addRow(4, r1, r2, r3);

gpane.addRow(5, text2);

gpane.addRow(6, c1, c2);

gpane.addRow(7, c3, c4);

gpane.addRow(8, text3, hp);

gpane.addRow(9, btn);

gpane.addRow(10, btn1);

gpane.addRow(11, text);

Scene scene = new Scene(bp, 400, 450);

primaryStage.setScene(scene);

primaryStage.show();

}

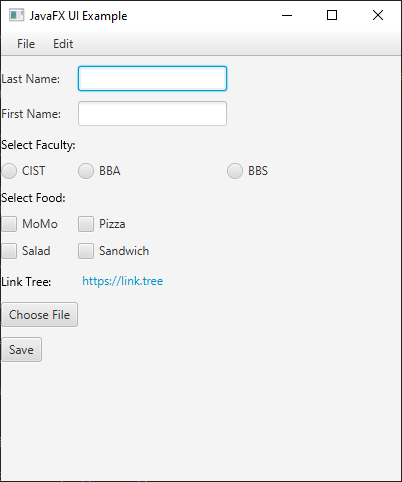
public static void main(String[] args) {

launch(args);

}

}

Output:



**Lab 53**

Write Java programs to demonstrate

a) RMI

b) CORBA

a) RMI

The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.The RMI provides remote communication between the applications using two objects stub and skeleton.

Steps:

Create the interface “Remoteinterface” then write necessary code in it

Create the class “RemoteServer” then write necessary code in it

Create the class “RemoteClient” then write necessary code in it

Run the RemoteServer.java file first then run RemoteClient.java file

Source code:

RemoteInterface.java

package rmidemo;

import java.rmi.\*;

public interface RemoteInterface extends Remote {

public String sayHello() throws RemoteException;

public int add(int a,int b) throws RemoteException;

}

RemoteServer.java

package rmidemo;

import java.rmi.\*;

import java.rmi.server.\*;

import java.rmi.registry.Registry;

import java.rmi.registry.LocateRegistry;

public class RemoteServer extends UnicastRemoteObject implements RemoteInterface {

public RemoteServer() throws RemoteException {

super();

}

public String sayHello() throws RemoteException {

return "This is a demonstration of RMI";

}

public int add(int a,int b) throws RemoteException {

return a+b;

}

public static void main(String[] args) throws Exception {

try{

Registry reg=LocateRegistry.createRegistry(999);

reg.rebind("hi server",new RemoteServer());

System.out.println("server is ready");

}catch(RemoteException e){

System.out.println("exception "+e);

}

}

}

RemoteClient.java

package rmidemo;

import java.rmi.\*;

import java.rmi.registry.Registry;

import java.rmi.registry.LocateRegistry;

import java.util.Scanner;

public class RemoteClient {

public static void main(String[] args) throws Exception {

try {

Registry reg = LocateRegistry.getRegistry("localhost", 999);

RemoteInterface rem = (RemoteInterface) reg.lookup("hi server");

String message = rem.sayHello();

System.out.println("Server says: " + message);

Scanner sc=new Scanner(System.in);

System.out.println("Enter two number:");

int a=sc.nextInt();

int b=sc.nextInt();

System.out.println("Addition is "+rem.add(a, b));

System.out.println("\nLab No:53 \nName: \nRoll No/Section: ");

} catch (RemoteException e) {

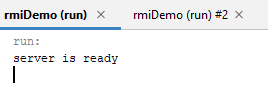
System.out.println("exception " + e);

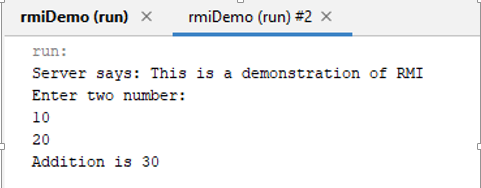
}

}

}

Output:





b) CORBA

CORBA (Common Object Request Broker Architecture) is a distributed object-oriented middleware technology that allows objects written in different programming languages and running on different platforms to communicate and interoperate with each other. It is a specification that defines a standard interface and protocol for objects to communicate with each other, regardless of the language or platform used to implement them.

Steps:

Create two new projects

Create a new project named "corbaServer" for the server and another named "corbaClient" for the client.

Create the IDL file

In the "corbaServer" project, create an empty file named "hello.idl".

Generate the required files using IDL Compiler

Open the terminal and navigate to the directory where "hello.idl" is located.

Run the IDL compiler command:

idlj -fall hello.idl (if environmental path for Java is set).

"C:\Program Files\Java\jdk1.8.0\_202\bin\idlj" -fall hello.idl (if environmental path for Java is not set).

Six files (Java files) will be generated.

Create new helloImp class

Create a new class named "helloImp" in the "corbaServer" project.

Write the necessary code for server and client

In the "corbaServer" project, implement the necessary code in corbaServer.java .

In the "corbaClient" project, implement the necessary code in corbaClient.java .

Set the program arguments

In the "corbaServer" project, go to Properties > Run and set the argument to "-ORBInitialPort 600".

In the "corbaClient" project, go to Properties > Run and set the argument to "-ORBInitialPort 600".

Start the Object Request Broker Daemon (ORBD)

Open the terminal and run the command:

start orbd -ORBInitialPort 600 (if environmental variable for Java is set).

"C:\Program Files\Java\jdk1.8.0\_202\bin\orbd” -ORBInitialPort 600 (if environmental variable for Java is not set).

This starts the ORB daemon on port 600.

Run the server and client

Run the server application by executing the "corbaServer" class in the "corbaServer" project.

Run the client application by executing the "corbaClient" class in the "corbaClient" project.

**Source code:**

**Hello.idl**

module CorbaExample {

interface HelloWorld {

string sayHello();

double add(in double a,in double b);

};

};

**helloImp.java**

package corbaserver;

import CorbaExample.HelloWorldPOA;

import org.omg.CORBA.ORB;

public class helloImp extends HelloWorldPOA {

private ORB orb;

public void setOrb(ORB orb1){

orb=orb1;

}

@Override

public String sayHello() {

return "This is an example of CORBA";

}

@Override

public double add(double a, double b) {

return a+b;

}

}

**corbaServer.java**

package corbaserver;

import CorbaExample.HelloWorld;

import CorbaExample.HelloWorldHelper;

import org.omg.CORBA.ORB;

import org.omg.CosNaming.NameComponent;

import org.omg.CosNaming.NamingContextExt;

import org.omg.CosNaming.NamingContextExtHelper;

import org.omg.PortableServer.POA;

import org.omg.PortableServer.POAHelper;

public class CorbaServer {

public static void main(String[] args) {

try {

// Create and initialize the ORB

ORB orb = ORB.init(args, null);

POA rootpoa = POAHelper.narrow(orb.resolve\_initial\_references("RootPOA"));

rootpoa.the\_POAManager().activate();

// Create the remote object

helloImp imp = new helloImp();

imp.setOrb(orb);

org.omg.CORBA.Object ref = rootpoa.servant\_to\_reference(imp);

HelloWorld href = HelloWorldHelper.narrow(ref);

// Get a reference to the root naming context

org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");

NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

// Bind the remote object to the naming service

NameComponent[] nc = ncRef.to\_name("HelloWorld");

ncRef.rebind(nc, href);

System.out.println("Server ready and waiting...");

// Wait for requests

orb.run();

} catch (Exception e) {

System.err.println("Error: " + e);

e.printStackTrace(System.out);

}

}

}

**corbaClient.java**

package corbaclient;

import CorbaExample.HelloWorld;

import CorbaExample.HelloWorldHelper;

import java.util.Scanner;

import org.omg.CORBA.ORB;

import org.omg.CosNaming.NamingContextExt;

import org.omg.CosNaming.NamingContextExtHelper;

public class CorbaClient {

public static void main(String[] args) {

try {

// Create and initialize the ORB

ORB orb = ORB.init(args, null);

// Get a reference to the root naming context

org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");

NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

// Get a reference to the remote object

HelloWorld helloWorld = HelloWorldHelper.narrow(ncRef.resolve\_str("HelloWorld"));

// Invoke the remote method

String message = helloWorld.sayHello();

System.out.println("Message from server: " + message);

Scanner sc = new Scanner(System.in);

System.out.println("Enter 1st number: ");

double a = sc.nextDouble();

System.out.println("Enter 2nd number: ");

double b = sc.nextDouble();

double sum = helloWorld.add(a, b);

System.out.println("Addition is " + sum);

System.out.println("\nLab No:53\_b \nName: \nRoll No/Section: ");

} catch (Exception e) {

System.err.println("Error: " + e);

e.printStackTrace(System.out);

}

}

}

Output:

