

School of Engineering & Technology
Department of Computer Science & Technology



Advanced Java Programming Lab
(CSP014)

Lab File
(2024-2025)

for

B. Tech. (CSE)
4th Semester

Submitted To:

Dr. Krishan Kumar
Associate Professor,
Department of Computer Science & Engineering
School of Engineering & Technology
Sharda University

Submitted By:

Aditya Pandey
B. Tech. CSE [4th Semester]
2301010055
Group 1



SCHOOL OF ENGINEERING & TECHNOLOGY DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

COURSE CODE	COURSE TITLE	L	T	P
CSP014	ADVANCE JAVA PROGRAMMING LAB.	0	0	2

Course Outcomes

1. Design Collection Framework with Data structures.
2. Develop software applications using object-oriented design methodology and database.
3. Create a program using JDBC connectivity.
4. Develop web-based program using servlet and JSP.
5. Model the concept of MVC architecture.
6. Make use of web development approach to develop real life applications.

INDEX

S.NO	EXPERIMENT NAME	COURSE OUTCOME	PAGE NO.	REMARK
1.	1. Create a program to generate ArrayList to store, retrieve, and manipulate data. 2. Create a program to generate HashMap to store key-value pairs and perform operations like adding, retrieving, and checking for keys.	CO1		
2.	Create a program to illustrates the use of common implementations of the Set interface, such as HashSet, LinkedHashSet, and TreeSet.	CO1		
3.	Establish a CRUD operation using MySQL	CO1		
4.	Write a program to display all records of student table.	CO2		
5.	Implement controller in JPA with session	CO2		
6.	1. Write a java Program to create a simple servlet and run it using tomcat server. 2. Write Servlet application to print current date & time	CO3		
7.	Write a java Program to create a servlet to read information from client Registration page	CO4		
8.	1. Write a program in hibernate with CFG and HBM file to perform a CRUD operation. 2. Create a program to define a JPA entity and a corresponding repository to interact with a database.	CO4		
9.	Develop a simple JSP program for user login form and authenticate user from data-base entries	CO5, CO6		
10.	Create REST APIs with Spring boot	CO5		
11.	Write programs to fetch details of students using spring framework.	CO5		
12.	Create a service-based code in spring boot	CO6		

VALUE ADDED EXPERIMENTS

S.NO .	EXPERIMENT NAME	COURSE OUTCOM E
A	Create a employee CRUD code in java using hibernate.	C01
B	Create a web application using JSP for employee management	C03
C	Display given employee details from employee database	CO2
D	Create a console-based student system in spring boot.	CO5

Experiment-01

Title: 1- Basic operations of ArrayList and HashMaps

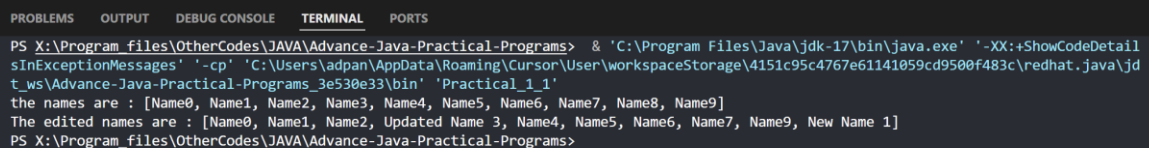
Problem – 1.1: Create a program to generate ArrayList to store, retrieve, and manipulate data.

Source Code:

```
import java.util.*;

public class Practical_1_1 {
    public static void main(String[] args) {
        ArrayList<String> names = new ArrayList<>();
        for(int i = 0 ; i < 10 ; i++){
            names.add("Name"+i);
        }
        System.out.println("the names are : "+names);
        names.set(3, "Updated Name 3");
        names.remove(8);
        names.add("New Name 1");
        System.err.println("The edited names are : "+names);
    }
}
```

Sample Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS X:\Program_files\OtherCodes\JAVA\Advance-Java-Practical-Programs> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetail
sInExceptionMessages' '-cp' 'C:\Users\adpan\AppData\Roaming\Cursor\User\workspaceStorage\4151c95c4767e61141059cd9500f483c\redhat.java\jd
t_ws\Advance-Java-Practical-Programs_3e530e33\bin' 'Practical_1_1'
the names are : [Name0, Name1, Name2, Name3, Name4, Name5, Name6, Name7, Name8, Name9]
The edited names are : [Name0, Name1, Name2, Updated Name 3, Name4, Name5, Name6, Name7, Name9, New Name 1]
PS X:\Program_files\OtherCodes\JAVA\Advance-Java-Practical-Programs>
```

Problem – 1.2: Create a program to generate HashMap to store key-value pairs and perform operations like adding, retrieving, and checking for keys.

Source Code:

```
import java.util.*;
public class Practical_1_2 {
    public static void main(String[] args) {
        HashMap<String,Integer> nameAgeMap = new HashMap<>();
        nameAgeMap.put("John",25);
        nameAgeMap.put("Jane",30);
        nameAgeMap.put("Jim",35);
        nameAgeMap.put("Jill",40);
        System.out.println("Name and Age Map : "+nameAgeMap);

        int ageOfJohn = nameAgeMap.get("John");
        System.out.println("Age of John : "+ageOfJohn);

        boolean isJanePresent =
nameAgeMap.containsKey("Jane");
        if(isJanePresent){
            System.out.println("Jane is present in the map");
        }
        else{
            System.out.println("Jane is not present in the
map");
        }

        for(Map.Entry<String,Integer> eachName :
nameAgeMap.entrySet()){
            System.err.println("Name : "+eachName.getKey()+"
Age : "+eachName.getValue());
        }
    }
}
```

Sample Output:

```
ograms'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\adpan\AppData\Roaming\C
ursor\User\workspaceStorage\4151c95c4767e61141059cd9500f483c\redhat.java\jdt_ws\Advance-Java-Practical-Programs_3e530e33\bin' 'Practical
_1_2'
Name and Age Map : {John=25, Jill=40, Jane=30, Jim=35}
Age of John : 25
Jane is present in the map
Name : John Age : 25
Name : Jill Age : 40
Name : Jane Age : 30
Name : Jim Age : 35
PS X:\Program_files\OtherCodes\JAVA\Advance-Java-Practical-Programs> |
```

Experiment-02

Problem : Create a program to illustrates the use of common implementations of the Set interface, such as HashSet, LinkedHashSet, and TreeSet.

Source Code:

```
import java.util.HashSet;
import java.util.LinkedHashSet;
import java.util.Set;
import java.util.TreeSet;

public class SetExample {
    public static void main(String[] args) {
        // HashSet Example
        Set<String> hashSet = new HashSet<>();
        hashSet.add("Cherry");
        hashSet.add("Banana");
        hashSet.add("Apple");
        hashSet.add("Banana"); // Duplicate element
        System.out.println("HashSet (no specific order, no
duplicates): " + hashSet);

        // LinkedHashSet Example
        Set<String> linkedHashSet = new LinkedHashSet<>();
        linkedHashSet.add("Cherry");
        linkedHashSet.add("Banana");
        linkedHashSet.add("Banana"); // Duplicate element
        linkedHashSet.add("Apple");
        System.out.println("LinkedHashSet (insertion order, no
duplicates): " + linkedHashSet);

        // TreeSet Example
        Set<String> treeSet = new TreeSet<>();
        treeSet.add("Banana");
        treeSet.add("Apple");
        treeSet.add("Cherry");
        treeSet.add("Banana"); // Duplicate element
        System.out.println("TreeSet (sorted order, no
duplicates): " + treeSet);

        System.out.println("\nBehavior Comparison:");
        System.out.println("HashSet retains no order: " +
hashSet);
        System.out.println("LinkedHashSet retains insertion
order: " + linkedHashSet);
        System.out.println("TreeSet retains sorted order: " +
treeSet);
    }
}
```

Sample Output:

```
● PS X:\Program_files\OtherCodes\JAVA\Advance-Java-Practical-Programs> cd "x:\Progr
($?) { java SetExample }
HashSet (no specific order, no duplicates): [Apple, Cherry, Banana]
LinkedHashSet (insertion order, no duplicates): [Cherry, Banana, Apple]
TreeSet (sorted order, no duplicates): [Apple, Banana, Cherry]

Behavior Comparison:
HashSet retains no order: [Apple, Cherry, Banana]
LinkedHashSet retains insertion order: [Cherry, Banana, Apple]
TreeSet retains sorted order: [Apple, Banana, Cherry]
○ PS X:\Program_files\OtherCodes\JAVA\Advance-Java-Practical-Programs> █
```


Experiment-03

Title : Establish a CRUD operation using MySQL

Source Code:

```
package JDBC;

import java.sql.*;
import java.io.*;

public class CRUDOperations {
    private static final String URL =
"jdbc:mysql://127.0.0.1:3306/javasql";
    private static final String USER = "root";
    private static final String PASSWORD = "root";
    private static final BufferedReader reader = new
BufferedReader(new InputStreamReader(System.in));

    public static void main(String[] args) {
        while (true) {
            try {
                System.out.println("Select operation: \n1.
Create \n2. Read \n3. Update \n4. Delete \n5. Exit");
                int choice =
Integer.parseInt(reader.readLine());
                switch (choice) {
                    case 1: createRecord(); break;
                    case 2: readRecords(); break;
                    case 3: updateRecord(); break;
                    case 4: deleteRecord(); break;
                    case 5:
                        System.out.println("Exiting...");
                        reader.close();
                        return;
                    default: System.out.println("Invalid
choice, please try again.");
                }
            } catch (IOException e) {
                System.out.println("Error reading input: " +
e.getMessage());
                break;
            }
        }

        public static void createRecord() {
            try (Connection con = DriverManager.getConnection(URL,
```

```

USER, PASSWORD)) {
    System.out.println("Connection established.");
} catch (Exception e) {
    System.out.println("Error: " + e.getMessage());
}
}

public static void readRecords() {
    try (Connection con = DriverManager.getConnection(URL,
USER, PASSWORD)) {
        String query = "SELECT * FROM table1";
        PreparedStatement stmt =
con.prepareStatement(query);
        ResultSet rs = stmt.executeQuery();
        System.out.println("Person details:");
        while (rs.next()) {
            System.out.println(rs.getInt("id") + " " +
rs.getString("city"));
        }
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
    }
}

public static void updateRecord() {
    try (Connection con = DriverManager.getConnection(URL,
USER, PASSWORD)) {
        System.out.println("Enter old city name:");
        String oldName = reader.readLine();
        System.out.println("Enter new city name:");
        String newName = reader.readLine();
        String query = "UPDATE table1 SET city=? WHERE
city=?";
        PreparedStatement stmt =
con.prepareStatement(query);
        stmt.setString(1, newName);
        stmt.setString(2, oldName);
        int rowsAffected = stmt.executeUpdate();
        System.out.println(rowsAffected > 0 ? "Record
updated successfully" : "No records updated.");
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
    }
}

public static void deleteRecord() {
    try (Connection con = DriverManager.getConnection(URL,

```

```

USER, PASSWORD)) {
    System.out.println("Enter ID to delete:");
    int id = Integer.parseInt(reader.readLine());
    String query = "DELETE FROM table1 WHERE id=?";
    PreparedStatement stmt =
con.prepareStatement(query);
    stmt.setInt(1, id);
    int rowsAffected = stmt.executeUpdate();
    System.out.println(rowsAffected > 0 ? "Record
deleted successfully" : "Record not found.");
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
    }
}
}
}

```

Sample Output:

```

"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C
Select operation:
1. Create
2. Read
3. Update
4. Delete
5. Exit
1
Connection established.
Select operation:
1. Create
2. Read
3. Update
4. Delete
5. Exit
2
Person details:
1 Delhi
3 Los Angeles
4 Noida
5 Meerut
6 San Francisco
Select operation:
1. Create
2. Read
3. Update
4. Delete
5. Exit
3
Enter old city name:
Delhi
Enter new city name:
New Delhi
Record updated successfully

```

```

Record updated successfully
Select operation:
1. Create
2. Read
3. Update
4. Delete
5. Exit
4
Enter ID to delete:
6
Record deleted successfully
Select operation:
1. Create
2. Read
3. Update
4. Delete
5. Exit
2
Person details:
1 New Delhi
3 Los Angeles
4 Noida
5 Meerut
Select operation:
1. Create
2. Read
3. Update
4. Delete
5. Exit
5
Exiting...

```

Experiment-04

Title : Write a program to display all records of student table.

Source Code:

```
import java.sql.*;
import java.io.*;
import java.util.*;
public class SelectQuery {
    public static void main(String[] args){
        String url = "jdbc:mysql://127.0.0.1:3306/javasql";
        String user = "root";
        String password = "kcm@123";
        Scanner s = new Scanner(System.in);
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Establish the connection
            Connection con = DriverManager.getConnection(url,
user, password);
            if (con.isClosed()) {
                System.out.println("Connection is closed.");
            } else {
                System.out.println("Connection formed.");
            }
            String q = " select * from table1";
            PreparedStatement pt = con.prepareStatement(q);
            ResultSet set = pt.executeQuery();
            System.out.println("Person detail are as
follow:");
            while(set.next()){
                int Id = set.getInt("id");
                String city = set.getString("city");
                System.out.println(Id+" "+city);
            }
            con.close();
        }
        catch (Exception e) {
            System.out.println("error:" + e.getMessage());
        }
        s.close();
    }
}
```

Sample Output:

```
[mysql-explain] classPool: $5, jdk.internal.loader.ClassLoaders$AppClassLoader@76ed5528, [class path: jdk.internal.loader.ClassLoaders$AppClassLoader@76ed5528;]
[mysql-explain] visit method: com.mysql.cj.jdbc.ClientPreparedStatement.executeInternal(int,com.mysql.cj.protocol.Message,boolean,boolean,com.mysql.cj.protocol.ColumnDefinition,boolean)
Connection formed.
Person detail are as follow:
1 Noida
2 Greater Noida
3 Jammu
4 Jammu
5 Jammu
6 Jammu
7 Jammu
8 Noida
9 Kerala
10 YYY
```

Experiment-05

Title : Write a program to display "Hello, World!" using a Servlet

Description: This experiment demonstrates the creation of a basic servlet that outputs a "Hello, World!" message when accessed via a web browser.

HelloServlet.java

```
package com.JAVA;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h1>Hello, World!</h1>");
    }
}
```

HTML FILE (index.html)

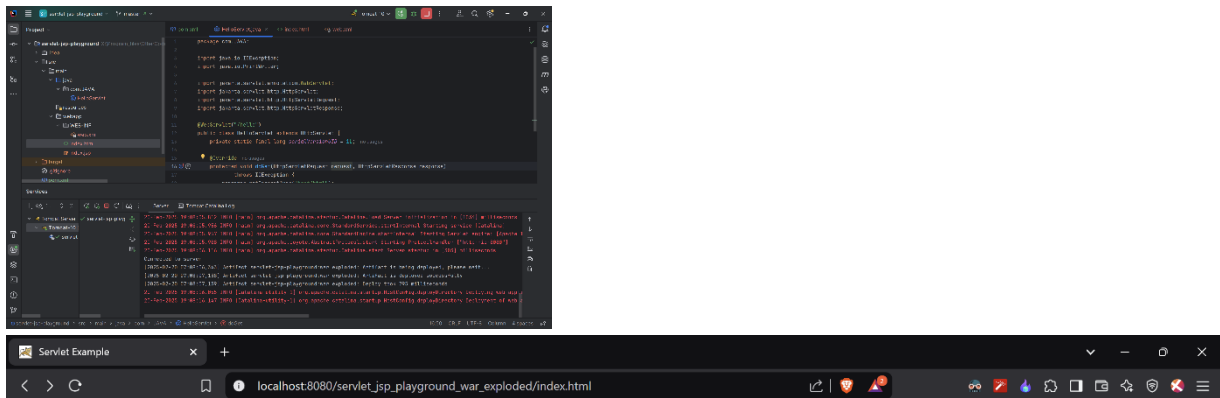
```
<!DOCTYPE html>
<html>
<head>
    <title>Servlet Example</title>
</head>
<body>
<h2>Click to see the servlet response</h2>
<a href="hello">Run Servlet</a>
</body>
</html>
```

Web.xml Configuration

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

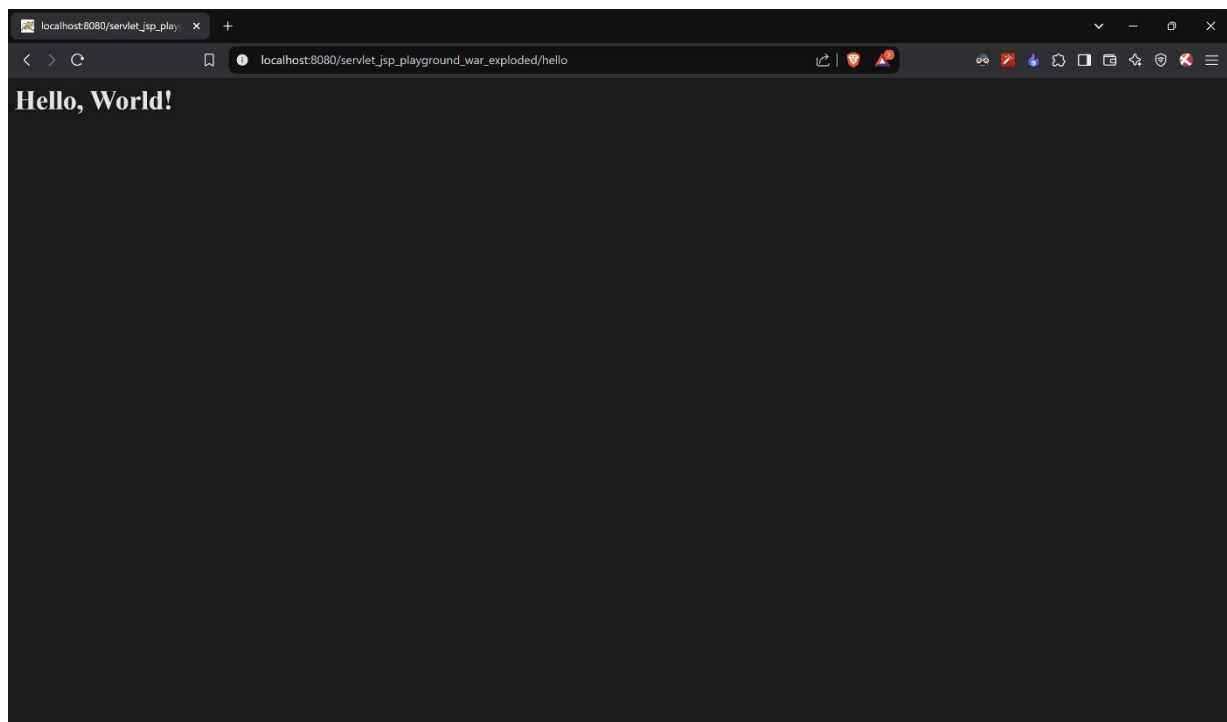
          xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                               http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
          version="4.0">
  <!-- No servlet mapping required if using annotations -->
</web-app>
```

Output :



Click to see the servlet response

Run Servlet



Experiment-06

Program 1 : Write a program to greet the user by handling request parameters in a Servlet

Description: In this experiment, a servlet retrieves a request parameter (the user's name) from the URL query string and displays a personalized greeting.

ParameterServlet.java

```
package com.JAVA;

import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

@WebServlet("/greet")
public class ParameterServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String name = request.getParameter("name");
        if (name == null || name.isEmpty()) {
            name = "Guest";
        }
        out.println("<h1>Hello, " + name + "!</h1>");
    }
}
```

HTML FILE (form.html)

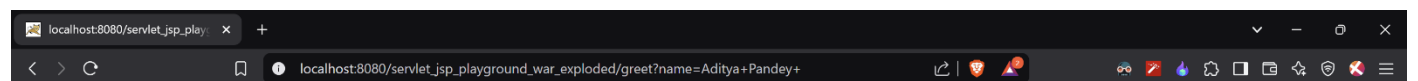
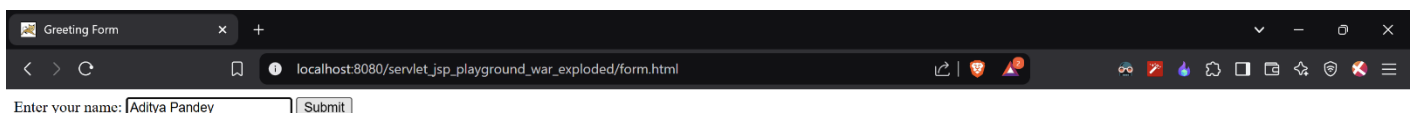
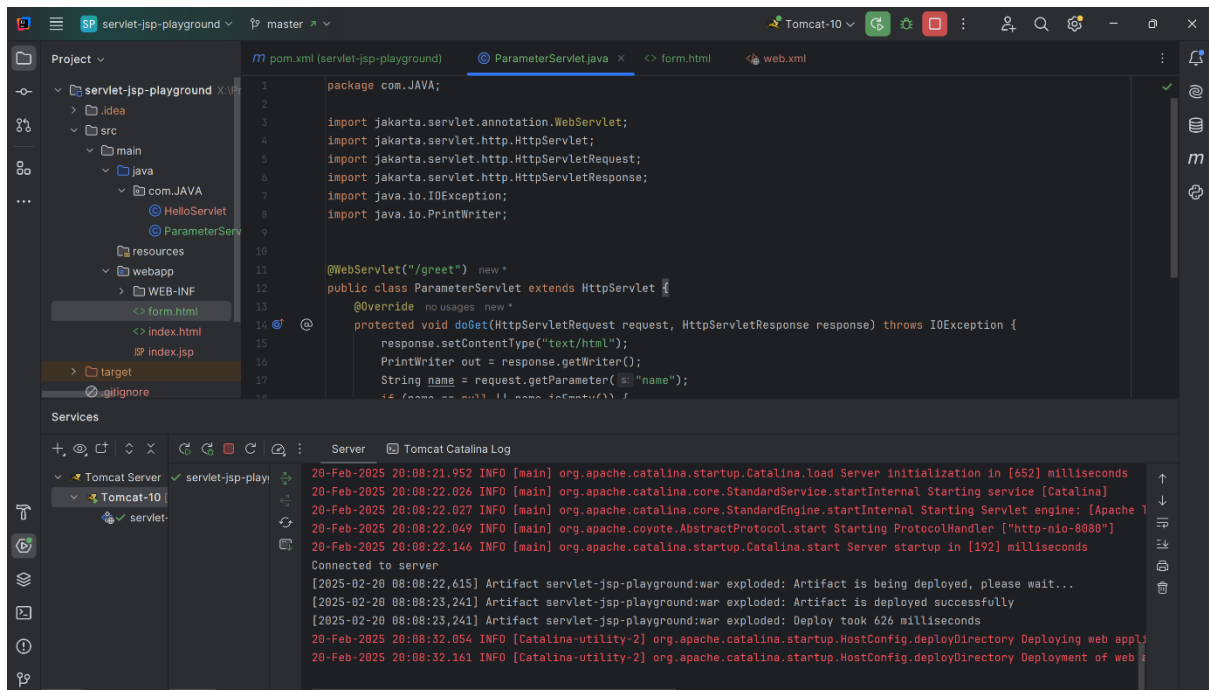
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Greeting Form</title>
</head>
<body>
<form action="greet">
    Enter your name: <input type="text" name="name">
    <input type="submit" value="Submit">
</form>
</body>
</html>
```

Web.xml Configuration

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">
    <!-- No servlet mapping required if using annotations -->
</web-app>
```

Output :



Hello, Aditya Pandey !

Program 2 : Write a program to validate user login using a Servlet handling POST requests

Description: This experiment focuses on processing form data sent via a POST request. The servlet validates the username and password and returns a success or failure message.

PostServlet.java

```
package com.JAVA;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@WebServlet("/login")
public class PostServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        if ("admin".equals(username) &&
            "password123".equals(password)) {
            out.println("<h1>Login Successful</h1>");
        } else {
            out.println("<h1>Invalid Credentials</h1>");
        }
    }
}
```

HTML FILE (login.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Login Page</title>
</head>
<body>
<form method="post" action="login">
    Username: <input type="text" name="username"><br>
    Password: <input type="password" name="password"><br>
    <input type="submit" value="Login">
</form>
</body>
</html>
```

Web.xml Configuration

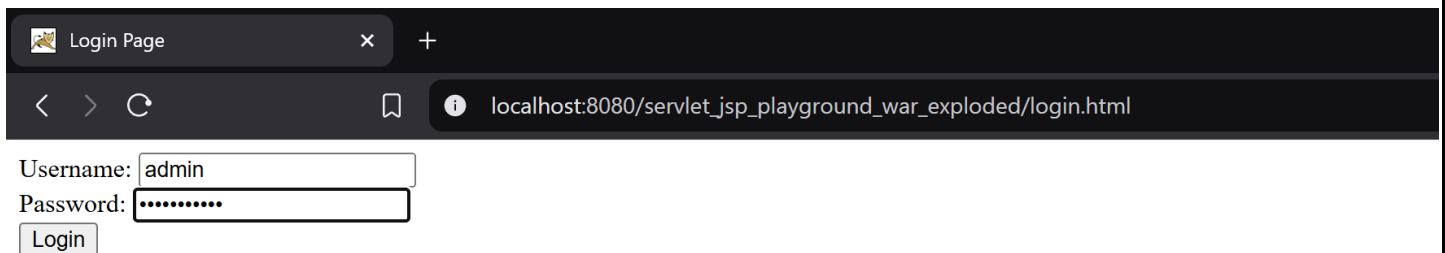
```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
```

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

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

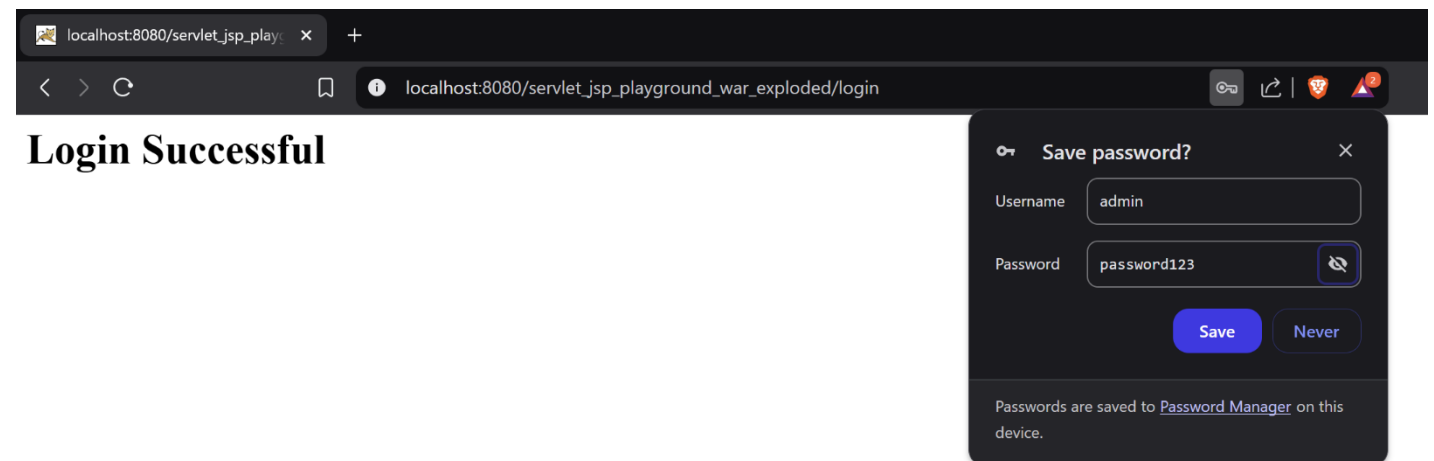
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
    version="4.0">
    <!-- No servlet mapping required if using annotations -->
</web-app>
```

Output :



Username:

Password:



localhost:8080/servlet_jsp_playground_war_exploded/login

Login Successful

Save password?

Username

Password

Passwords are saved to [Password Manager](#) on this device.

Experiment-07

Program 1 : Write a program to demonstrate session management using a Servlet

Description: This experiment illustrates how to use HTTP sessions to manage user state across multiple requests within a servlet-based web application.

SessionServlet.java

```
package com.JAVA;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;

@WebServlet("/session")
public class SessionServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        // Set the response content type
        response.setContentType("text/html");

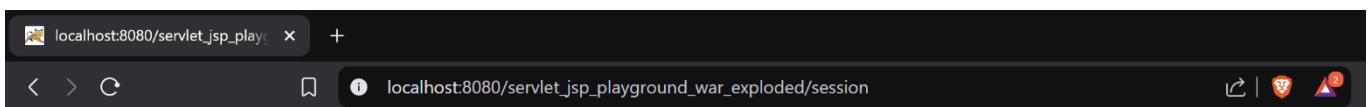
        // Get or create the session for this user
        HttpSession session = request.getSession();

        // Retrieve the "username" attribute from the session
        String username = (String) session.getAttribute("username");

        // If not present, initialize it with a default value
        if (username == null) {
            username = "New User";
            session.setAttribute("username", username);
        }

        // Write the response including the session ID and username
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>Welcome, " + username + "!</h1>");
        out.println("<p>Your session ID is: " + session.getId() +
            "</p>");
        out.println("</body></html>");
    }
}
```

Output :



Welcome, New User!

Your session ID is: 92D8A4FFF91353B530BDD624F56ED564

Experiment-07

Program 2 : Write a program to display all records from the "users" table using a Servlet with Database Connectivity

Description: In this experiment, the servlet connects to a MySQL database, retrieves all records from the "users" table, and displays them on a web page.

SessionServlet.java

```
package com.JAVA;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@WebServlet("/users")
public class DBServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/adityapandey",
                "root", "root"); // Use correct DB name here

            PreparedStatement ps = con.prepareStatement("SELECT *
FROM ADITYA_PANDEY");
            ResultSet rs = ps.executeQuery();

            out.println("<h2>Records from ADITYA_PANDEY
Table:</h2>");
            while (rs.next()) {
                out.println("<p>Record: " +
                    rs.getInt("ID") + " | " +
                    rs.getInt("ROLL NUMBER") + " | " +
                    rs.getString("NAME") + " | " +
                    rs.getInt("AGE") + " | " +
                    rs.getLong("CONTACT_NUMBER") +
                    "</p>");
            }
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            out.println("<p>Error: " + e.getMessage() + "</p>");
        }
    }
}
```

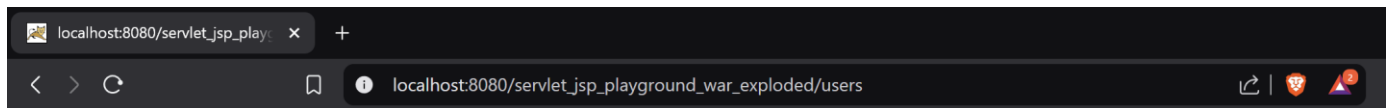
Output :

The screenshot shows an IDE with the following components:

- Project View:** A tree structure showing the project 'servlet-jsp-playground' with subfolders 'src', 'main', 'java', 'resources', and 'webapp'. The 'java' folder contains a package 'com.JAVA' with several servlet classes: DBServlet, HelloServlet, ParameterServlet, PostServlet, and SessionServlet.
- Code Editor:** The 'DBServlet.java' file is open, showing the following code:

```
1 package com.JAVA;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 import java.sql.Connection;
6 import java.sql.DriverManager;
7 import java.sql.PreparedStatement;
8 import java.sql.ResultSet;
9 import jakarta.servlet.ServletException;
10 import jakarta.servlet.annotation.WebServlet;
11 import jakarta.servlet.http.HttpServlet;
12 import jakarta.servlet.http.HttpServletRequest;
13 import jakarta.servlet.http.HttpServletResponse;
14
15 @WebServlet("/users") new *
16 public class DBServlet extends HttpServlet {
17     private static final long serialVersionUID = 1L; no usages
18
19     @Override no usages new *
20     protected void doGet(HttpServletRequest request, HttpServletResponse response)
```
- Services View:** The 'Tomcat-10' server is selected, showing the 'Tomcat Catalina Log'.
- Log Output:** The log shows the following messages:

```
28-Feb-2025 21:07:11.861 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [179] milliseconds
Connected to server
[2025-02-20 09:07:12,030] Artifact servlet-jsp-playground:war exploded: Artifact is being deployed, please wait...
28-Feb-2025 21:07:13.168 INFO [RMI TCP Connection(2)-127.0.0.1] org.apache.jasper.servlet.TldScanner.scanJars At least one J
[2025-02-20 09:07:13,337] Artifact servlet-jsp-playground:war exploded: Artifact is deployed successfully
[2025-02-20 09:07:13,337] Artifact servlet-jsp-playground:war exploded: Deploy took 1,307 milliseconds
28-Feb-2025 21:07:21.788 INFO [Catalina-utility-2] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web appli
28-Feb-2025 21:07:21.893 INFO [Catalina-utility-2] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web e
```



Records from ADITYA_PANDEY Table:

Record: 1 | 1 | Aditya Pandey | 22 | 9876543210

Record: 2 | 2 | Ravi Verma | 23 | 9876543211

Record: 3 | 3 | Sanya Sharma | 21 | 9876543212

Record: 4 | 4 | Rajesh Singh | 24 | 9876543213

Record: 5 | 5 | Neha Gupta | 20 | 9876543214

Experiment-10

Title : Create REST APIs with Spring boot.

Entity Class

```
package com.example.demo.entity;

public class JournalEntity {
    private int id;
    private String title;
    private String content;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
}
```

Controller Class:

```
package com.example.demo.controller;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.demo.entity.JournalEntity;

@RestController
@RequestMapping("/journal")
```

```

public class JournalController {

    Map<Integer,JournalEntity> map = new HashMap<>();

    @GetMapping
    public List<JournalEntity> getAll(){
        if(map.containsKey(0)){
            return new ArrayList<>();
        }
        return new ArrayList<>(map.values());
    }

    @PostMapping
    public JournalEntity createNewEntity(@RequestBody
    JournalEntity newEntity){
        map.put(newEntity.getId(),newEntity);
        return map.get(newEntity.getId());
    }

    @PutMapping("id/{id}")
    public String updateEntries(@PathVariable int id,
    @RequestBody JournalEntity updateData){
        if(!map.containsKey(id)){
            return "Entry does n't exists";
        }
        map.put(id,updateData);
        return "Data update Sucessfully";
    }

    @DeleteMapping("id/{id}")
    public String deleteEntries(@PathVariable int id){
        if(!map.containsKey(id)){
            return "Data can't be deleted";
        }
        map.remove(id);
        return "Data delete Sucessfully";
    }
}

```

Main Java Class:

```

package com.example.demo;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

}

```

Output:

```
[
  {
    "id": 1,
    "title": "happy",
    "content": "I am happy"
  },
  {
    "id": 2,
    "title": "ok",
    "content": "I am ok"
  },
  {
    "id": 3,
    "title": "alright",
    "content": "I am alright"
  },
  {
    "id": 4,
    "title": "good",
    "content": "I am good"
  }
]
```


Experiment-11

Title : Write programs to fetch details of students using spring framework.

Entity Class

```
package com.example.studentapp.model;

import jakarta.persistence.*;

@Entity
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private int age;
    private String course;

    public Student() {}

    public Student(String name, int age, String course) {
        this.name = name;
        this.age = age;
        this.course = course;
    }

    // Getters & Setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
    public String getCourse() { return course; }
    public void setCourse(String course) { this.course = course; }
}
```

StudentRepository

```
package com.example.studentapp.repository;

import com.example.studentapp.model.Student;
import org.springframework.data.jpa.repository.JpaRepository;

public interface StudentRepository extends JpaRepository<Student,
Long> {}
```

StudentService

```
package com.example.studentapp.service;

import com.example.studentapp.model.Student;
import com.example.studentapp.repository.StudentRepository;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class StudentService {
    private final StudentRepository studentRepository;

    public StudentService(StudentRepository studentRepository) {
        this.studentRepository = studentRepository;
    }

    public List<Student> getAllStudents() {
        return studentRepository.findAll();
    }

    public Optional<Student> getStudentById(Long id) {
        return studentRepository.findById(id);
    }

    public Student saveStudent(Student student) {
        return studentRepository.save(student);
    }

    public void deleteStudent(Long id) {
        studentRepository.deleteById(id);
    }
}
```

StudentController

```
package com.example.studentapp.controller;

import com.example.studentapp.model.Student;
import com.example.studentapp.service.StudentService;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/students")
public class StudentController {
    private final StudentService studentService;

    public StudentController(StudentService studentService) {
        this.studentService = studentService;
    }
}
```

```

    @GetMapping
    public List<Student> getAllStudents() {
        return studentService.getAllStudents();
    }

    @GetMapping("/{id}")
    public Optional<Student> getStudentById(@PathVariable Long id) {
        return studentService.getStudentById(id);
    }

    @PostMapping
    public Student addStudent(@RequestBody Student student) {
        return studentService.saveStudent(student);
    }

    @DeleteMapping("/{id}")
    public void deleteStudent(@PathVariable Long id) {
        studentService.deleteStudent(id);
    }
}

```

StudentAppApplication

```

package com.example.studentapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class StudentAppApplication {
    public static void main(String[] args) {
        SpringApplication.run(StudentAppApplication.class, args);
    }
}

```

application.properties

```

spring.datasource.url=jdbc:h2:mem:studentdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
spring.jpa.hibernate.ddl-auto=update

```

Output

```
{  
  "name": "John Doe",  
  "age": 20,  
  "course": "Computer Science"  
}
```

Experiment-12

Title : Create a service-based code in spring boot

UserEntity

```
package com.example.demo.model;

import jakarta.persistence.*;

@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;

    // Constructors
    public User() {}
    public User(String name, String email) {
        this.name = name;
        this.email = email;
    }

    // Getters & Setters
    public Long getId() { return id; }
    public String getName() { return name; }
    public String getEmail() { return email; }
    public void setId(Long id) { this.id = id; }
    public void setName(String name) { this.name = name; }
    public void setEmail(String email) { this.email = email; }
}
```

UserRepository

```
package com.example.demo.repository;

import com.example.demo.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {}
```

UserService

```
package com.example.demo.service;

import com.example.demo.model.User;
import com.example.demo.repository.UserRepository;
import org.springframework.stereotype.Service;
```

```

import java.util.List;
import java.util.Optional;

@Service
public class UserService {
    private final UserRepository userRepository;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public List<User> getAllUsers() {
        return userRepository.findAll();
    }

    public Optional<User> getUserById(Long id) {
        return userRepository.findById(id);
    }

    public User createUser(User user) {
        return userRepository.save(user);
    }

    public void deleteUser(Long id) {
        userRepository.deleteById(id);
    }
}

```

UserController

```

package com.example.demo.controller;

import com.example.demo.model.User;
import com.example.demo.service.UserService;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/users")
public class UserController {
    private final UserService userService;

    public UserController(UserService userService) {
        this.userService = userService;
    }

    @GetMapping
    public List<User> getAllUsers() {
        return userService.getAllUsers();
    }

    @GetMapping("/{id}")
    public Optional<User> getUserById(@PathVariable Long id) {

```

```

        return userService.getUserById(id);
    }

    @PostMapping
    public User createUser(@RequestBody User user) {
        return userService.createUser(user);
    }

    @DeleteMapping("/{id}")
    public void deleteUser(@PathVariable Long id) {
        userService.deleteUser(id);
    }
}

```

application.properties

```

spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true

```

MainClass

```

package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}

```

Output

Request:

json

```
{  
  "name": "Alice",  
  "email": "alice@example.com"  
}
```

Response:

json

```
{  
  "id": 1,  
  "name": "Alice",  
  "email": "alice@example.com"  
}
```