Experiment: Banking Management System Web Page using Servlet, JavaScript Validation & Database Connectivity

Course / Subject: Web Technologies / Advanced Java (Servlets & JDBC)
Experiment No.: 04
Date: _____

Aim

To develop a Banking Management System web page using HTML for the user interface, JavaScript for validating form inputs, and Java Servlet for backend processing and database connectivity.

Apparatus / Software Required

- Java JDK (8 or above)
- Apache Tomcat Server (8/9/10)
- MySQL / PostgreSQL Database
- JDBC Driver
- Text Editor (VS Code / Eclipse / IntelliJ)
- Web Browser (Chrome / Firefox)

Theory / Background

A Banking Management System (BMS) manages customer details, account data, deposits, withdrawals, and transactions. This experiment focuses on creating a **customer account creation form** validated using JavaScript and processed using Java Servlets. The servlet will store customer details into the database.

Components Used

- **HTML** → UI for user input.
- **JavaScript** → Validates form entries on client side.
- **Servlet** → Processes request and interacts with backend.
- **JDBC (Java Database Connectivity)** → Executes SQL queries.
- **MySQL** → Stores banking records.

Procedure

1. Install and configure Tomcat server in your IDE.
2. Create a dynamic web project named `BankingSystem`.
3. Create a SQL database named `bankdb` with a table `customers`.
4. Create an HTML form for bank account registration.
5. Add JavaScript code to validate the form.
6. Write a Servlet (`BankServlet.java`) that connects to the database and stores form data.
7. Configure servlet mapping in `web.xml`.
8. Deploy the project and test the system.

Database Structure

Table: **customers**

Field	Type	Description
id	INT AUTO_INCREMENT	Primary Key
name	VARCHAR(100)	Customer Name
email	VARCHAR(100)	Email Address
mobile	VARCHAR(15)	Contact Number
accountType	VARCHAR(30)	Savings / Current
balance	DOUBLE	Opening Balance

Code Implementation

1. HTML Page — `bank.html`

```
<!DOCTYPE html>
<html>
<head>
  <title>Bank Account Registration</title>
  <script>
    function validateForm() {
      let name = document.getElementById("name").value;
      let email = document.getElementById("email").value;
      let mobile = document.getElementById("mobile").value;
      let balance = document.getElementById("balance").value;

      if (name.trim() === "") {
        alert("Name is required");
        return false;
      }

      const emailPattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;
```

```
      if (!email.match(emailPattern)) {
        alert("Enter a valid email");
        return false;
      }

      if (mobile.length !== 10 || isNaN(mobile)) {
        alert("Mobile number must be 10 digits");
        return false;
      }

      if (isNaN(balance) || balance <= 0) {
        alert("Enter a valid opening balance");
        return false;
      }
      return true;
    }
  </script>
</head>

<body>
  <h2>Banking Management System</h2>

  <form action="BankServlet" method="post" onsubmit="return validateForm()">
    <label>Customer Name:</label><br>
    <input type="text" id="name" name="name"><br><br>

    <label>Email:</label><br>
    <input type="text" id="email" name="email"><br><br>

    <label>Mobile No:</label><br>
    <input type="text" id="mobile" name="mobile"><br><br>

    <label>Account Type:</label><br>
    <select name="accountType">
      <option value="Savings">Savings</option>
      <option value="Current">Current</option>
    </select><br><br>

    <label>Opening Balance:</label><br>
    <input type="text" id="balance" name="balance"><br><br>

    <input type="submit" value="Create Account">
  </form>

</body>
</html>
```

---

2. Servlet — `BankServlet.java`

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class BankServlet extends HttpServlet {
  protected void doPost(HttpServletRequest request, HttpServletResponse
response)
      throws ServletException, IOException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    String name = request.getParameter("name");
    String email = request.getParameter("email");
    String mobile = request.getParameter("mobile");
    String accountType = request.getParameter("accountType");
    double balance = Double.parseDouble(request.getParameter("balance"));

    try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      Connection con = DriverManager.getConnection(
          "jdbc:mysql://localhost:3306/bankdb", "root", "password");

      PreparedStatement ps = con.prepareStatement(
          "INSERT INTO customers(name, email, mobile, accountType, balance)
VALUES(?,?,?,?,?)");

      ps.setString(1, name);
      ps.setString(2, email);
      ps.setString(3, mobile);
      ps.setString(4, accountType);
      ps.setDouble(5, balance);

      int result = ps.executeUpdate();

      if (result > 0) {
        out.println("<h3>Account Created Successfully!</h3>");
      } else {
        out.println("<h3>Error occurred while creating account.</h3>");
      }

      con.close();

    } catch (Exception e) {
      out.println("<h3>Error: " + e.getMessage() + "</h3>");
    }
```

```
  }
}
```

---

3. web.xml Configuration

```
<web-app>
  <servlet>
    <servlet-name>BankServlet</servlet-name>
    <servlet-class>BankServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>BankServlet</servlet-name>
    <url-pattern>/BankServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

Observations

1. JavaScript successfully validates form fields on the client side.
2. Servlet receives form data after validation.
3. JDBC connection inserts customer records into the database.
4. Incorrect inputs are rejected before reaching the server.
5. Records can be verified through SQL queries in MySQL.

Result

A Banking Management System web page was successfully developed using HTML, JavaScript validation, Servlet backend, and database connectivity. Users can register new bank accounts, and the data is stored securely in the database.

Conclusion

This experiment demonstrates how banking applications are built using Java Servlets and web technologies. Client-side validation ensures correctness of data, while server-side processing and JDBC ensure secure database operations.

Viva Questions

1. What is the role of JavaScript in form validation?
2. What is JDBC and why is it used?

3. Explain the life cycle of a Servlet.
4. Difference between GET and POST methods?
5. Why are PreparedStatements preferred over Statements?
6. How does a servlet interact with a database?

End of Experiment Document