

## Practical 10

### Account.java :

```
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;

@Entity
public class Account {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private int userId;
    private double balance;

    // Constructors, getters, and setters
    public Account() {}

    public Account(int userId, double balance) {
        this.userId = userId;
        this.balance = balance;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public double getBalance() {
        return balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }
}
```

**AccountEJB.java :**

```
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.transaction.Transactional;

@Stateless
public class AccountEJB {

    @PersistenceContext
    private EntityManager em;

    @Transactional
    public String deposit(int userId, double amount) {
        Account account = em.createQuery("SELECT a FROM Account a WHERE a.userId = :userId", Account.class)
            .setParameter("userId", userId)
            .getSingleResult();

        if (account != null) {
            account.setBalance(account.getBalance() + amount);
            em.merge(account);
            return "Deposit successful!";
        }
        return "Account not found!";
    }

    @Transactional
    public String withdraw(int userId, double amount) {
        Account account = em.createQuery("SELECT a FROM Account a WHERE a.userId = :userId", Account.class)
            .setParameter("userId", userId)
            .getSingleResult();

        if (account != null) {
            if (account.getBalance() >= amount) {
                account.setBalance(account.getBalance() - amount);
                em.merge(account);
                return "Withdrawal successful!";
            } else {
                return "Insufficient balance!";
            }
        }
        return "Account not found!";
    }
}
```

**AccountServlet.java :**

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.ejb.EJB;
import java.io.IOException;

public class AccountServlet extends HttpServlet {

    @EJB
    private AccountEJB accountEJB;
```

```

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    String action = request.getParameter("action");
    int userId = Integer.parseInt(request.getParameter("userId"));
    double amount = Double.parseDouble(request.getParameter("amount"));
    String message = "";

    if ("deposit".equals(action)) {
        message = accountEJB.deposit(userId, amount);
    } else if ("withdraw".equals(action)) {
        message = accountEJB.withdraw(userId, amount);
    }

    request.setAttribute("message", message);
    request.getRequestDispatcher("/result.jsp").forward(request, response);
}
}

```

### index.jsp :

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Account Transaction</title>
</head>
<body>
    <h2>Account Transaction</h2>
    <form action="AccountServlet" method="POST">
        <label for="userId">User ID:</label>
        <input type="number" id="userId" name="userId" required><br><br>

        <label for="amount">Amount:</label>
        <input type="number" id="amount" name="amount" step="0.01" required><br><br>

        <label for="action">Action:</label>
        <select id="action" name="action">
            <option value="deposit">Deposit</option>
            <option value="withdraw">Withdraw</option>
        </select><br><br>

        <button type="submit">Submit</button>
    </form>

    <br>
    <c:if test="${not empty message}">
        <div>${message}</div>
    </c:if>
</body>
</html>

```

### result.jsp

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Transaction Result</title>

```

```
</head>
<body>
  <h2>Transaction Result</h2>
  <div>${message}</div>
</body>
</html>
```

### web.xml

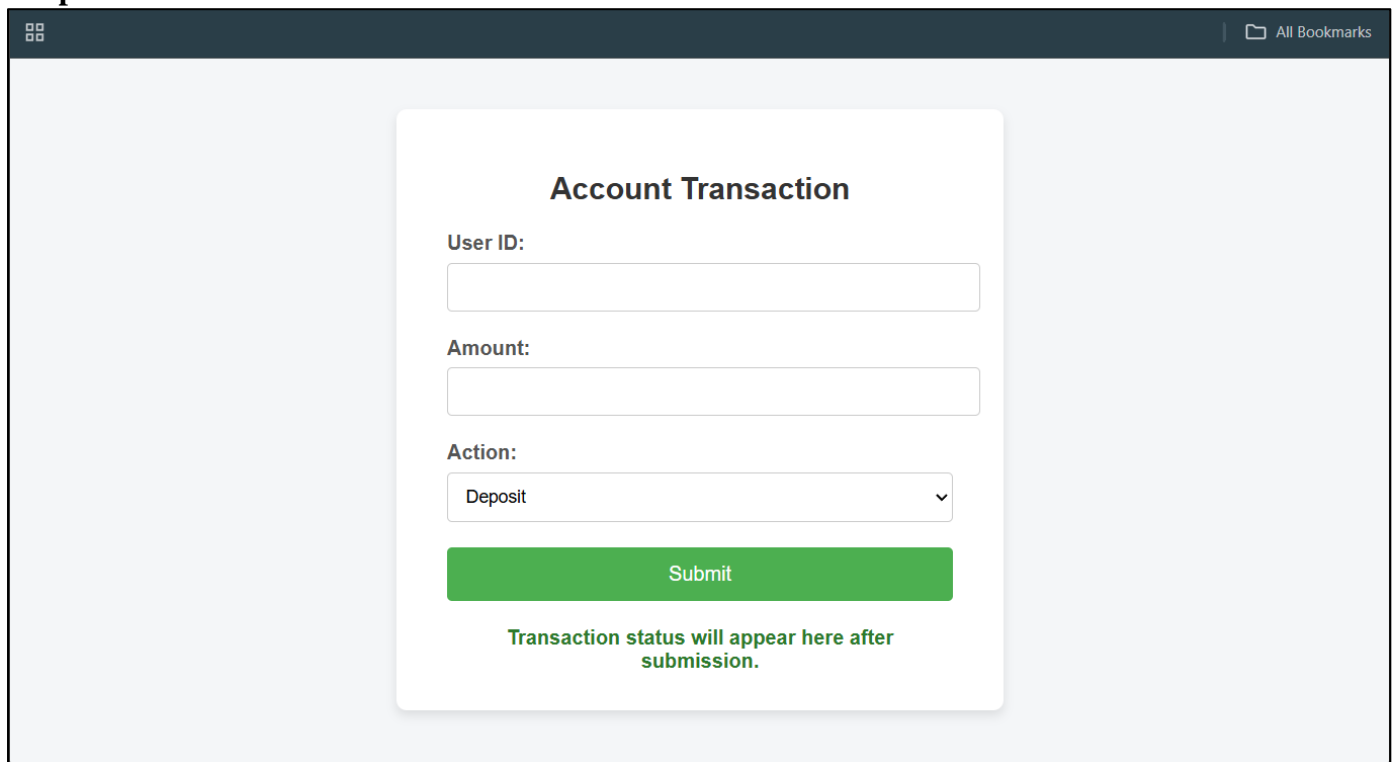
```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">

  <servlet>
    <servlet-name>AccountServlet</servlet-name>
    <servlet-class>com.example.AccountServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>AccountServlet</servlet-name>
    <url-pattern>/AccountServlet</url-pattern>
  </servlet-mapping>

</web-app>
```

### Output :



The screenshot displays a web browser window with a dark header bar containing a grid icon and a link to 'All Bookmarks'. The main content area features a white card titled 'Account Transaction'. Inside the card, there are three input fields: 'User ID:', 'Amount:', and 'Action:'. The 'Action:' field is a dropdown menu currently showing 'Deposit'. Below these fields is a prominent green 'Submit' button. At the bottom of the card, a green message states: 'Transaction status will appear here after submission.'

All Bookmarks

### Account Transaction

User ID:

Amount:

Action:

Withdraw

Submit

Transaction status will appear here after submission.

All Bookmarks

### Transaction Result

Withdrawal successful!