Assignment-4

Why JDBC is Essential in Building Database-Driven Applications

JDBC is essential because it provides a standard API for java applications to interact with databases.

Achieving JDBC Connection Pooling Using JDBC Data Source and JNDI in Apache Tomcat

Configure DataSource

```
<Resource name="jdbc/MyDB"
          auth="Container"
          type="javax.sql.DataSource"
          maxTotal="20"
          maxIdle="10"
          maxWaitMillis="10000"
          username="dbuser"
          password="dbpassword"
          driverClassName="com.mysql.cj.jdbc.
                           Driver"
          url="jdbc:mysql://localhost:3306/
               mydatabase"/>
```

LookUp DataSource in Java Code Using JN

```
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.sql.DataSource;
import java.sql.Connection;
public class DatabaseUtil {
    public static Connection getConnection()
                    throws Exception {
```

Conte
Data
retu
}
}
Executing
① Using a
try(connec
Statemen
String q
Results
while(
Sys
}
}
② Using
try(con
prepa
pstm
Resu
whi
}
}

```
        Context initContext = new InitialContext();
        DataSource ds = (DataSource) initContext.
                                    lookup("java:/comp/
                                    env/jdbc/MyDB");
        return ds.getConnection();
    }
}
```

Executing SQL Queries Using JDBC Statement

① Using a Statement

```
try(Connection conn = DatabaseUtil.getConnection();
    Statement stmt = conn.createStatement()){
    String query = "SELECT * FROM users";
    ResultSet rs = stmt.excuteQuery(query);
    while(rs.next()){
        System.out.println("User ID:" + rs.getInt
                ("id") + " + Name:" + rs.getString("name"));

    }
}
```

② Using a PreparedStatement

```
try(Connection conn = DatabaseUtil.getConnection();
    PreparedStatement pstmt = conn.prepareStatement
        ("SELECT * FROM users WHERE id = ?")){
    pstmt.setInt(1, 1);
    ResultSet rs = pstmt.executeQuery();
    while(rs.next()){
        System.out.println("User ID:" + rs.getInt
            ("id") + ", Name:" + rs.getString("name"));

    }
}
```

③ Using a CallableStatement for stored procedures:

```
try(Connection conn=DatabaseUtil.getConnection();
   CallableStatement cstmt=conn.prepareCall("{
            call getUserById(?)}")){

   cstmt.setInt(1,1);
   ResultSet rs=cstmt.executeQuery();
   while(rs.next()){
      System.out.println("User ID:"+rs.getInt("id")
         +", Name:"+rs.getString("name"));

   }
}
```

Output:

Statement Example output:

User ID:1, Name: Vishnu

User ID:2, Name: Priya

Prepared Statement

User ID:1, Name: Vishnu

CallableStatement

User ID:1, Name: Vishnu

② Lifecycle Phases of a JSP Page

1. Translation Phase
2. Compilation Phase
3. Initialization phase
4. Request Processing Phase
5. Destruction phase

Embedding Java Code

① Scriptlets

```
<% int sum=5+10;%>
<P>The sum is:<%= sum
```

Output:

The sum is:15

② Declarations

```
<%! int add(int a
<P> The result is:
```

Output:

The result is:10

③ Expressions

```
<P>Current tim
```

Output:

Current time:

Scriptlets:

Advantages:

Java logic.

Disadvantage

to maintain

Declaration

Advantage

methods an

Disadvant

Java code

concerns.

Embedding Java Code in JSP

① Scriptlets

```
<% int sum = 5+10; %>
<P> The Sum is: <%= sum %>
```

Output:

The Sum is: 15

② Declarations

```
<%! int add(int a, int b){ return a+b; } %>
<P> The result is: <%= add(3,7) %></P>
```

Output:

The result is: 10

③ Expressions

```
<P>Current time: <%= new java.util.Date() %></P>
```

output:

Current time: Mon Sep 09 09:30:00 PDT 2024

Scriptlets:

Advantages: Easy to use for embedding simple Java logic.

Disadvantages: Leads to messy code, difficult to maintain

Declarations:

Advantages: Useful for declaring reusable methods and variables across multiple requests.

Disadvantages: Can clutter JSP with Java code, leading to poor separation of concerns.

Expressions:

Advantages: simplifies outputting dynamic content directly in JSP.

Disadvantages: limited to expressions

③ Generates a chessboard using HTML tables. width of 400px(total) and each cell h a height and width of 30px.

PHP Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Chessboard</title>
    <style>
        table{
            width: 400px;
            border-collapse: collapse;
        }
        td {
            width: 30px;
            height: 30px;
        }
    </style>
</head>
<body>
    <table>
        <?php
        //Loop for 8 rows
        for ($row=0; $row<8; $row++){
```

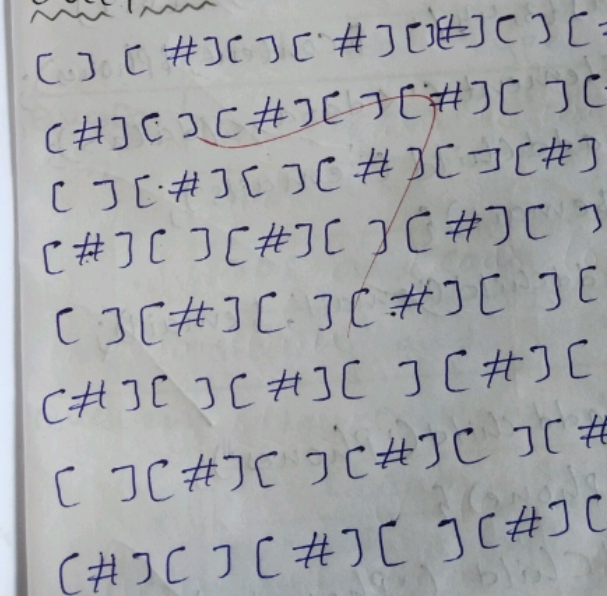
```
            echo "<tr>";
            for ($col=0; $col
                if (($row+
                    echo "
                }else{
                    echo
                }
            }
            echo "</tr
        }
        ?>
    </table>
</body>
</html>
```

Output:

[ ][#][ ][#
[#][ ][#][
[ ][#][ ][
[#][ ][#][
[ ][#][ ]
[#][ ][#]
[ ][#][
[#][ ][

```
        echo "<tr>";
        for ($col=0; $col<8; $col++){
            if (($row+$col)%2==0){
                echo "<td style='background-color:
                    white;'></td>";
            }else{
                echo "<td style='background-color:
                    black;'></td>";
            }
        }
        echo "</tr>";
    }
    ?>
</table>
</body>
</html>
```

Output:



③ Using a CallableStatement for Stored procedures:

```
try(Connection conn=DatabaseUtil.getConnection();
   CallableStatement cstmt=conn.prepareCall("{
            call getUserById(?) }")){

   cstmt.setInt(1, 1);
   ResultSet rs=cstmt.executeQuery();
   while(rs.next()){
      System.out.println("User ID:"+rs.getInt(
         +", Name:"+rs.getString("name"));

   }
}
```

Output:

Statement Example Output:

User ID: 1, Name: Vishnu

User ID: 2, Name: Priya

Prepared Statement

User ID: 1, Name: Vishnu

CallableStatement

User ID: 1, Name: Vishnu

② Lifecycle phases of a JSP Page

① Translation Phase

② Compilation Phase

③ Initialization phase

④ Request Processing phase

⑤ Destruction phase

Embedding

① Scriptlets

<% int sum=

<P> The Sum is

Output:

The Sum is

② Declarati

<%! int add

<P> The resu

Output:

The result

③ Expresio

<P>Current

Output:

Current tim

Scriptlet

Advantages

Java logic.

Disadvanta

to maintai

Declaratio

Advantages

methods and

Disadvanta

Java code,

Concerns.

Embedding Java Code in JSP

① Scriptlets

```
<% int sum = 5+10; %>
<P> The Sum is: <%= sum %>
```

Output:

The Sum is: 15

② Declarations

```
<%! int add(int a, int b){ return a+b; } %>
<P> The result is: <%= add(3,7) %></P>
```

Output:

The result is: 10

③ Expressions

```
<P>Current time: <%= new java.util.Date() %></P>
```

output:

Current time: Mon Sep 09 09:30:00 PDT 2024

Scriptlets:

Advantages: Easy to use for embedding simple Java logic.

Disadvantages: Leads to messy code, difficult to maintain

Declarations:

Advantages: Useful for declaring reusable methods and variables across multiple requests.

Disadvantages: Can clutter JSP with Java code, leading to poor separation of concerns.

Expressions:

Advantages: simplifies outputting dynamic content directly in JSP.

Disadvantages: limited to expressions

(3) Generates a chessboard using HTML tables. width of 400px(total) and each cell has a height and width of 30px.

PHP Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Chessboard</title>
    <style>
        table{
            width: 400px;
            border-collapse: collapse;
        }
        td {
            width: 30px;
            height: 30px;
        }
    </style>
</head>
<body>
    <table>
        <?php
        //Loop for 8 rows
        for ($row=0; $row<8; $row++){
```

```
        echo "<
        for ($co
            if (
                }
            }
            ech
        }
        ?>
    </table
</body>
</html>
```

Output:

[ ][#][
[#][ ][
[ ][#][
[#][ ][
[ ][#]
[#][ ][
[ ][#]
[#][ ]

```
        echo "<tr>";
        for ($col=0; $col<8; $col++){
            if (($row + $col)%2 ==0){
                echo "<td style='background-color:
                    White;'></td>";
            }else{
                echo "<td style=='background-color:
                    black;'></td>";
            }
        }
        echo "</tr>";
    }
    ?>
</table>
</body>
</html>
```

Output:

[ ][#][ ][#][ ][#][ ][#]
[#][ ][#][ ][#][ ][#][ ]
[ ][#][ ][#][ ][#][ ][#]
[#][ ][#][ ][#][ ][#][ ]
[ ][#][ ][#][ ][#][ ][#]
[#][ ][#][ ][#][ ][#][ ]
[ ][#][ ][#][ ][#][ ][#]
[#][ ][#][ ][#][ ][#][ ]

④ PHP Application to Extract Data and Store in XML

Steps:

① Read content from a Text File
② Extract patterns Using Regular.
③ Create and Store Results in an XML File.
④ Define XML schema.

PHP code:

```
<?php
$filename = 'input.txt';
$content = file_get_contents($filename);
preg_match_all('/[a-zA-Z0-9.-%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}/', $content, $emails);
preg_match_all('/\+?[0-9]{1,3}[-.\s]?[0-9]{1,4}[-.\s]?[0-9]{3,4}/', $content, $phones);
$xml = new SimpleXMLElement('<data></data>');
$emailElement = $xml->addChild('emails');
foreach ($emails[0] as $email) {
    $emailElement->addChild('email', $email);
}
$phoneElement = $xml->addChild('phones');
foreach ($phones[0] as $phone) {
    $phoneElement->addChild('phone', $phone);
}
$xml->asXML('output.xml');
echo "Data extracted and saved to output.xml";
?>
```

Output:

```
<data>
    <emails>
        <email>ex
        <email>exa
    </emails>
    <phones>
        <phone>
        <phone>
    </phones>
</data>
```

Assignment Rubrics	~~Marks~~
Question-1	Explan
	conne
	SQL
	State
Question-2	lifecy
	Embe
	Pros
	Clari
Question-3	cod
	HTI
	Alt
	Ex
Question-4	co
	Pa
	X
	C

Output:

```
<data>
    <emails>
        <email>example1@example.com</email>
        <email>example2@example.com</email>
    </emails>
    <phones>
        <phone>+123-456-7890</phone>
        <phone>987-654-3210</phone>
    </phones>
</data>
```

Assignment Rubrics	~~Marks~~ split up	Marks Score	Total marks
Question-1	Explanation of JDBC	5	5
	Connection Pooling	6	6
	SQL Queries	5	5
	Statement Types	4	4
Question-2	lifecycle phases Explanation	6	6
	Embedding Java code	5	5
	Pros and cons	5	5
	Clarity and Depth	4	4
Question-3	code Implementation	8	8
	HTML Table Structure	5	5
	Alternating Colors Logic	4	4
	Explanation	3	3
Question-4	Code Implementation	8	8
	Pattern Extraction	5	5
	XML file Generation	4	4
	DTD vs. XML schema	3	3