1. JDBC in database -driver

connection pooling.

Define the data source in

context.xml:

<Resource name = "jdbc/mydb" auth = "Container"

type = "javax.sql.datasource" driverClassName = "com.mysql.jdbc.Driver"

url = "jdbc:mysql://localhost:3306/mydb"

username = "root" password = "password";

Executing SQL Queries:

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("select * from users");

Example code:

PreparedStatement pstmt = conn.prepareStatement

("SELECT * FROM users WHERE id = ?");

pstmt.setInt(1, 1);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

System.out.println("UserID: " + rs.getInt("id"));

System.out.println("Username: " + rs.getString("username"));

}

conn.close();

Output:

UserID: 1

Username: Bhaskar

2. JSP lifecycle phases and embedding Java code:

The JSP lifecycle consists of the translation phase, compilation phase, initialization phase, request processing phase, destroy phase.

Embedding Java code.

1. Scriptlets (<% .... %>);

<% int count = 0; %>

2. Declarations (<%! .... %>):

<%! int count = 0; %>

3. Expression (<%= .... %>)

<% "Hello world!" %>

Example output: JSP lifecycle phase

<% "welcome to our website!" %>

output: welcome to our website.

JSP using Scriplet:

<%

int visitor count = 10;

out.println ("visitor count: + visitor count);

%>

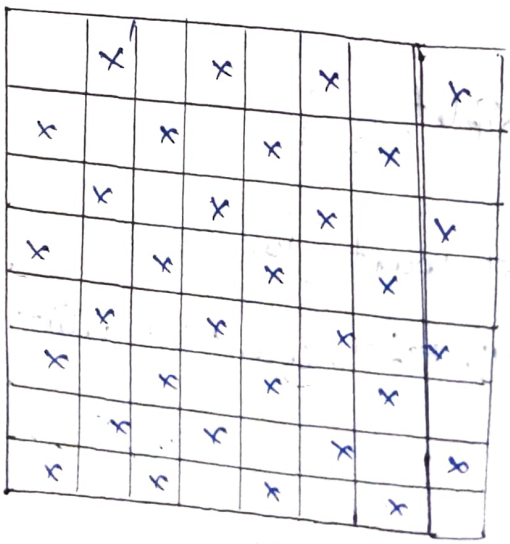
output:

3. PHP program to generate chessboard.

To generate chessboard with alternating colors using html tables in php

```
echo "<table width='400px' height='400px'
    cellpacing='0' cellpadding='0'>";
for ($row=0; $row<8; $row++)
    echo "<tr>";
for ($col=0; $col<8; $col++) {
$color = ($row+$col)%2 ==0? 'white' 'black';
echo "<style='width=width:30px; height=30px
        background-color:$color;></td>";
    }
    echo "</tr>";
}
echo "</table>";
```

output:



4. PHP application for extracting the pattern and xml question:

1. Read the file content:

```
$file content = file_get_content ('input.txt');
```

2. Use regular expressions to extract data;

```
preg_match_all ("/[a-z0-9._]+@[a-z0-9\.]+
    [a-z]{2,6}/", filetext, $ emails);
preg_match_all ("/\d{3}-\d{3}-\d{4}, filecontent,
$ phone number);
```

DTD vs xml schema:

DTD (document type definition):

→ simple & easier to define
→ does not support data types.

Eg code:

```
$ xml = new SimpleXML ('<data></data>');
foreach ($ emails [0] as $email){
  $ xml -> add child ('email', $email);
}
  foreach ($ phone number [0] as $phone)
  $ xml -> add child ('phone', $phone);
  }
  $ xml -> asXML ('output.xml');
```

output:

```
<email> john@example.com</email>
<phone> 987-283-143</phone>
```