Here's a step-by-step guide to creating a Dynamic Web Project in Eclipse, configuring Maven, and implementing the required features:

Step 1: Create Dynamic Web Project

1. Open Eclipse.

2. Go to File > New > Other.

3. Select "Dynamic Web Project" under the "Web" category.

4. Name the project (e.g., "BatchManagementSystem").

5. Choose the target runtime environment (e.g., Apache Tomcat 9).

6. Click Finish.

Step 2: Configure Maven

1. Right-click the project > Maven > Enable Dependency Management.

2. Create a pom.xml file with the following content:

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.batchmanagement</groupId>

<artifactId>BatchManagementSystem</artifactId>

<version>1.0</version>

<packaging>war</packaging>

<dependencies>

<!-- JDBC dependency -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.21</version>

</dependency>

<!-- Servlet dependency -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

Step 3: Create Java Classes

Create the following Java classes:

Participant.java

public class Participant {

private int id;

private String name;

private String email;

// Getters and setters

}

Batch.java

public class Batch {

private int id;

private String name;

private Date startDate;

// Getters and setters

}

Step 4: Create Servlets

Create the following Servlets:

ParticipantServlet.java

@WebServlet("/participant")

public class ParticipantServlet extends HttpServlet {

private ParticipantDAO participantDAO;

public void init() {

participantDAO = new ParticipantDAO();

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) {

// Handle participant CRUD operations

}

}

BatchServlet.java

@WebServlet("/batch")

public class BatchServlet extends HttpServlet {

private BatchDAO batchDAO;

public void init() {

batchDAO = new BatchDAO();

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) {

// Handle batch CRUD operations

}

}

Step 5: Create HTML Pages

Create the following HTML pages:

participant.html

<!DOCTYPE html>

<html>

<head>

<title>Participant Management</title>

</head>

<body>

<h1>Participant Management</h1>

<form action="participant" method="post">

<label>Name:</label>

<input type="text" name="name"><br>

<label>Email:</label>

<input type="email" name="email"><br>

<input type="submit" value="Save">

</form>

</body>

</html>

batch.html

<!DOCTYPE html>

<html>

<head>

<title>Batch Management</title>

</head>

<body>

<h1>Batch Management</h1>

<form action="batch" method="post">

<label>Name:</label>

<input type="text" name="name"><br>

<label>Start Date:</label>

<input type="date" name="startDate"><br>

<input type="submit" value="Save">

</form>

</body>

</html>

Step 6: Perform CRUD Operations

Implement CRUD operations using JDBC in the ParticipantDAO and BatchDAO classes.

Step 7: Create JSP Pages

Create JSP pages for query parameters.

Step 8: Build and Run

Build and run the project on the Apache Tomcat web server.

Step 9: Package

Package the project as a Jar file using Maven Package Goal