Experimental Report: Configuration and Deployment of a Java Web Application

Student Name: Mohammad Abdullah
Student ID: 202322240356
Course: Java Web
Date: 13 October 2025

1.0 Objective

The primary objective of this experiment was to establish a fully functional Java Enterprise development environment by configuring and integrating an Apache Tomcat server within the Visual Studio Code IDE. The success of this configuration was to be validated by building, deploying, and running a dynamic JavaServer Pages (JSP) web application created with Apache Maven. The process aimed to demonstrate proficiency in server setup, project management with Maven, and systematic troubleshooting of common development environment issues.

2.0 Environment Setup & Tools

The following software components were utilized to conduct this experiment:

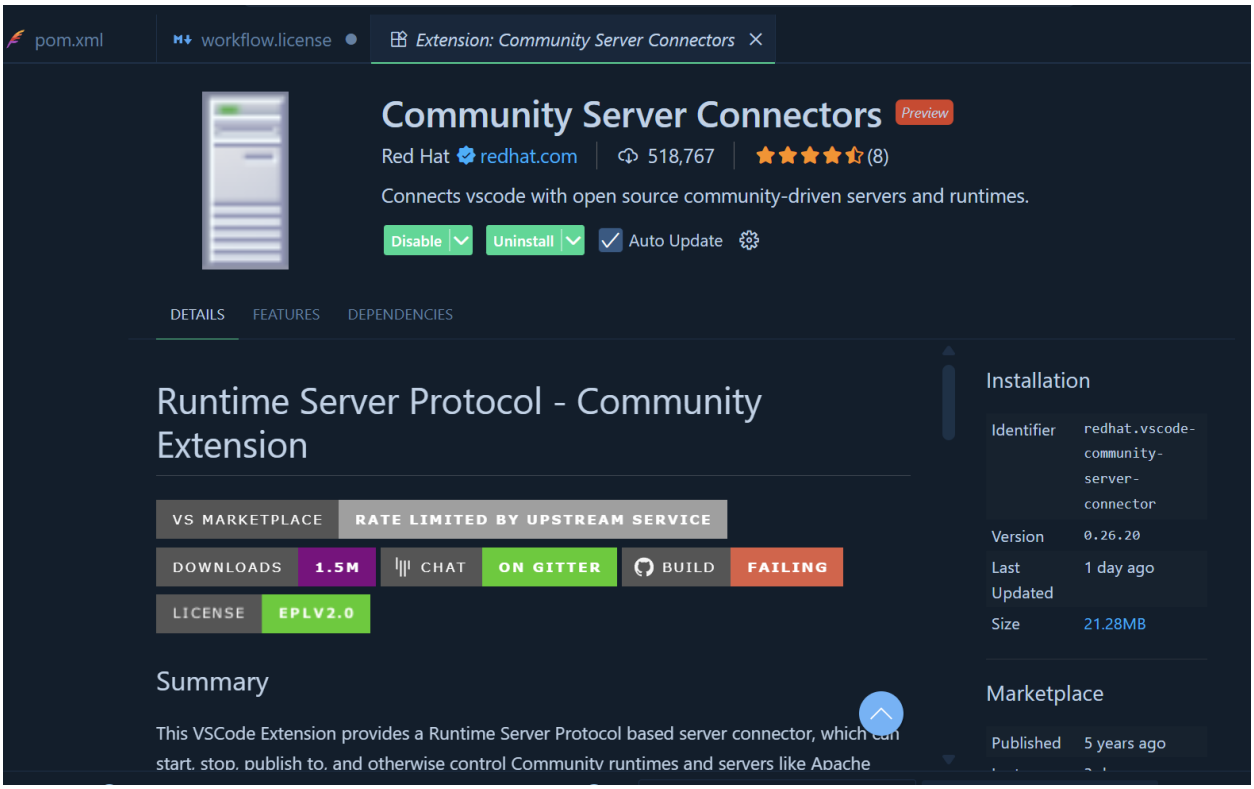
1. **Operating System:** Windows 11
2. **Integrated Development Environment (IDE):** Visual Studio Code (Version 1.94 or similar)
3. **Key VS Code Extensions:**
    a. Extension Pack for Java (Microsoft)
    b. Community Server Connectors (Red Hat)
4. **Java Development Kit (JDK):** Eclipse Adoptium JDK 17.0.15
5. **Application Server:** Apache Tomcat 11.0.0-M6
6. **Build Automation Tool:** Apache Maven 3.9.11

3.0 Configuration & Deployment Procedure

The configuration was executed through a systematic, multi-stage process, from environment preparation to final application deployment.

3.1 IDE and Toolchain Configuration

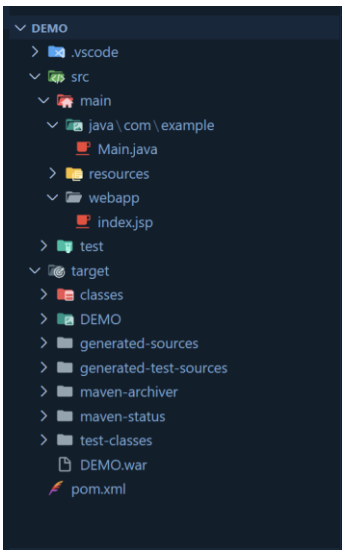
The initial step involved preparing VS Code for Java web development. The Extension Pack for Java provided core language support, while the Community Server Connectors extension enabled the integration of the Tomcat application server directly into the IDE's workflow. The Apache Tomcat server was downloaded and extracted to a local directory, and its path was registered within the Community Server Connectors extension.



(VS Code Extension Setup and SERVERS View showing the configured Tomcat server)

3.2 Maven Project Scaffolding and Correction

A new project was created using the maven-archetype-webapp archetype. During initial setup, a structural error was identified: the index.jsp file was incorrectly placed in src/main/java. This was rectified by creating the correct src/main/webapp directory and relocating the file, adhering to the standard directory layout for Java web applications.



(Final Project Structure in VS Code Explorer)

3.3 pom.xml Dependency Management

To enable the compilation of JSP and Servlet code, the project's pom.xml file was modified to include necessary dependencies. The javax.servlet-api and javax.servlet.jsp-api artifacts were added with a <scope>provided</scope>, instructing Maven that these libraries would be supplied by the Tomcat container at runtime and should not be bundled in the final artifact. The packaging type was explicitly set to war.

```
<?xml version="1.0" encoding="UTF-8"?>

<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.example</groupId>
  <artifactId>DEMO</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>DEMO Maven Webapp</name>
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>4.0.1</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet.jsp</groupId>
      <artifactId>javax.servlet.jsp-api</artifactId>
      <version>2.3.3</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>

  <build>
```

```
    <finalName>DEMO</finalName>
    <pluginManagement>
      <plugins>
        <plugin>
          <artifactId>maven-war-plugin</artifactId>
          <version>3.2.2</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

3.4 Building the Web Application Archive (WAR)

The project was built using the Maven install lifecycle goal, executed from the MAVEN view within VS Code. This process compiled the Java classes, processed resources, and packaged the application into a deployable DEMO.war file located in the /target directory.

```
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus
/plexus-utils/4.0.1/plexus-utils-4.0.1.jar (193 kB at 310 kB/s)
[INFO] Installing c:\Users\tUp\demo\target\DEMO.war to C:\Users\abdul\.m2\reposit
ory\com\example\DEMO\1.0-SNAPSHOT\DEMO-1.0-SNAPSHOT.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:19 min
[INFO] Finished at: 2025-10-13T01:10:44+08:00
[INFO] ------------------------------------------------------------------------
```

(The VS Code Terminal showing the "BUILD SUCCESS" message from Maven)

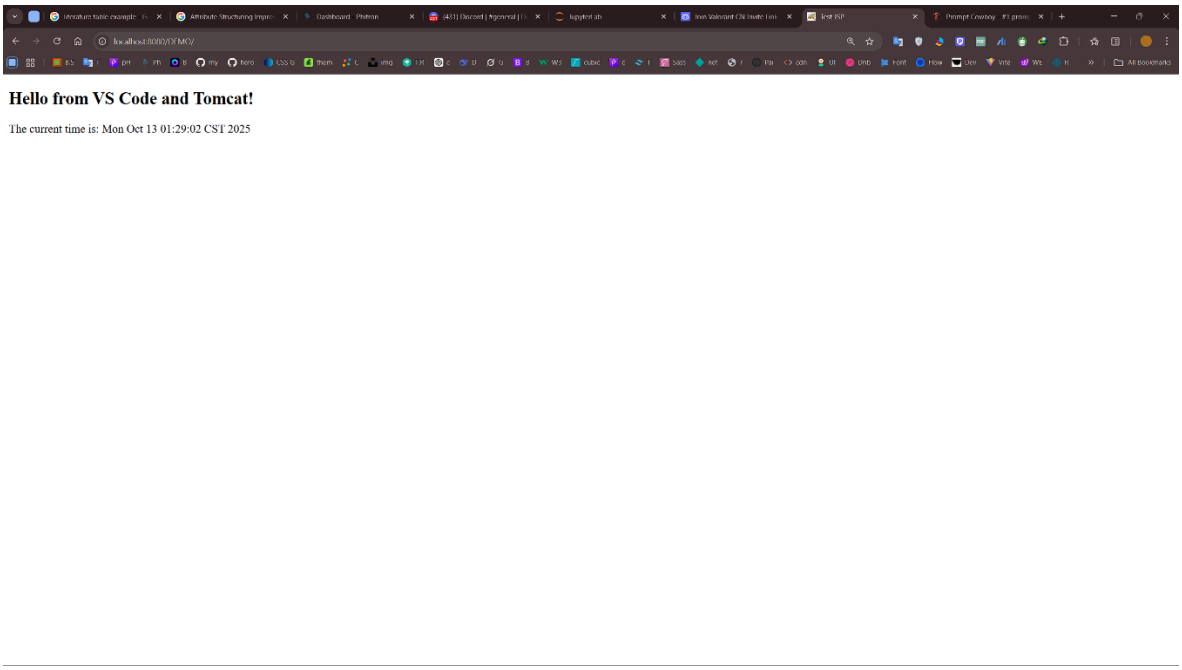
3.5 Deployment and Execution

The generated DEMO.war file was deployed to the running Tomcat server using the "Run on Server" command in VS Code. This action automatically started the server and published the web application.

4.0 Testing and Results

Upon server startup, a web browser was automatically launched, initially pointing to the Tomcat server's root URL (http://localhost:8080/), which displayed the default Tomcat homepage. By manually navigating to the application's correct context path, **http://localhost:8080/DEMO/**, the index.jsp page was successfully accessed.

The page correctly rendered the static "Hello from VS Code and Tomcat!" message and dynamically displayed the current server date and time, confirming that the JSP engine was processing the scriptlet tag (<%= new java.util.Date() %>). The outcome was a complete success, verifying that the end-to-end configuration of the IDE, build tool, and server was correct.



(The final running application in the web browser, clearly showing the URL and the dynamic content)

5.0 Problems Encountered and Resolutions

The configuration process presented several critical technical challenges that required systematic troubleshooting.

1. **Problem 1: Maven Executable Not Found**
    a. **Symptom:** The Maven build commands were unavailable within VS Code, and the right-click context menu did not show lifecycle goals like install.
    b. **Resolution:** Apache Maven was not installed as a standalone program. The issue was resolved by downloading Maven, extracting it to a local directory, and configuring the M2_HOME and Path system environment variables. After restarting VS Code, the Maven extension successfully detected the executable.
2. **Problem 2: pom.xml ModelParseException**
    a. **Symptom:** The Maven build failed immediately with a ModelParseException, indicating a syntax error on line 1 of the pom.xml file.
    b. **Resolution:** The file contained extraneous characters and was missing essential structural tags. This was resolved by completely replacing the file's content with a standardized, well-formed pom.xml template. This ensured correct XML syntax and the inclusion of all necessary sections, such as <packaging>, <dependencies>, and <build>.
3. **Problem 3: Application Not Found (HTTP 404 Error)**
    a. **Symptom:** After a successful deployment, the browser displayed the default Tomcat page instead of the application's index.jsp.

b. **Resolution:** This was identified as an incorrect URL. The server root (/) was being accessed instead of the application's context path. The issue was resolved by manually appending the application's final name (/DEMO) to the URL, successfully routing the request to the deployed web application.

6.0 Conclusion

This experiment successfully achieved its objective of configuring a robust Java web development environment. The process provided invaluable hands-on experience in integrating essential development tools, including VS Code, Maven, and the Apache Tomcat server. The successful resolution of build, configuration, and deployment errors demonstrated a practical understanding of Maven dependency management, standard project structures, and the fundamentals of web application deployment. The environment is now fully operational and suitable for future Java Enterprise development projects.