Experimental Report: Deployment and Execution of Java Servlets

Student Name: Mohammad Abdullah
Student ID: 202322240356

Course: Java Web
Date: 13 October 2025

1 Objective

The primary objective of this experiment was to successfully configure, deploy, and execute Java Servlets in a Tomcat 11 environment using Visual Studio Code. The experiment focused on resolving compatibility issues between Jakarta EE and Java EE specifications, building a functional web application with multiple servlets, and validating the deployment through comprehensive testing.

2 Environment Setup & Tools

The following software components were utilized to conduct this experiment:

Component	Specification
Operating System	Windows 11
Integrated Development Environment	Visual Studio Code (Version 1.94)
Java Development Kit	Eclipse Adoptium JDK 17.0.15
Application Server	Apache Tomcat 11.0.0-M6
Build Automation Tool	Apache Maven 3.9.11

Table 1: Software Environment Specifications

Key VS Code Extensions:

- Extension Pack for Java (Microsoft)
- Community Server Connectors (Red Hat)
- Maven for Java

3 Configuration & Deployment Procedure

3.1 Project Initialization and Structure

A new Maven web project was created using the `maven-archetype-webapp` archetype. The project structure was organized according to Maven standards, ensuring proper separation of Java source files, web resources, and configuration files.

3.2 Jakarta EE Compatibility Resolution

Initial deployment attempts resulted in HTTP 500 errors with the following critical issue identified:

Problem: Tomcat 11 requires Jakarta EE packages, but servlets were using deprecated javax.servlet packages.

Solution: All servlet files were updated to use Jakarta EE namespace. The code migration involved changing import statements from `javax.servlet` to `jakarta.servlet`.

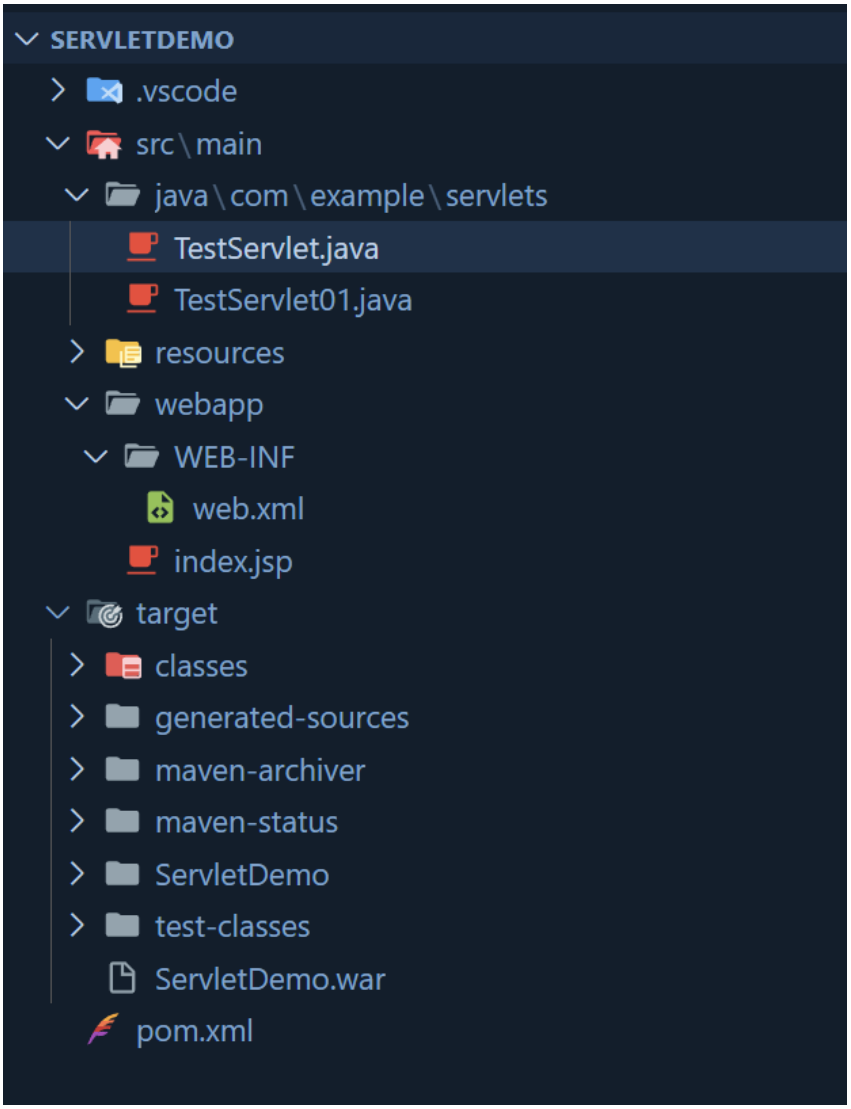

Figure 1: VS Code Explorer showing complete project structure

```
// BEFORE (Java EE - incompatible with Tomcat 11)
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;

// AFTER (Jakarta EE - compatible with Tomcat 11)
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
```

3.3 Maven Dependencies Configuration

The `pom.xml` file was configured with Jakarta EE dependencies to ensure compatibility with Tomcat 11:

```
<dependencies>
```

```
    <dependency>
        <groupId>jakarta.servlet</groupId>
        <artifactId>jakarta.servlet-api</artifactId>
        <version>6.0.0</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>jakarta.servlet.jsp</groupId>
        <artifactId>jakarta.servlet.jsp-api</artifactId>
        <version>3.1.1</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

3.4 Web Application Descriptor

The `web.xml` file was configured with proper Jakarta EE namespace and servlet mappings:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
         https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
         version="6.0">

    <servlet>
        <servlet-name>TestServlet01</servlet-name>
        <servlet-class>com.example.servlets.TestServlet01</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>TestServlet01</servlet-name>
        <url-pattern>/TestServlet01</url-pattern>
    </servlet-mapping>
</web-app>
```

3.5 Build and Deployment Process

The application underwent a systematic build and deployment process:

1. **Maven Build Execution:**

```
mvn clean package
```

2. **Manual Deployment Steps:**
   - WAR file (`ServletDemo.war`) copied to Tomcat `webapps` directory
   - Tomcat server restarted via VS Code Servers view
   - Automatic extraction verified by checking for `ServletDemo/` folder creation

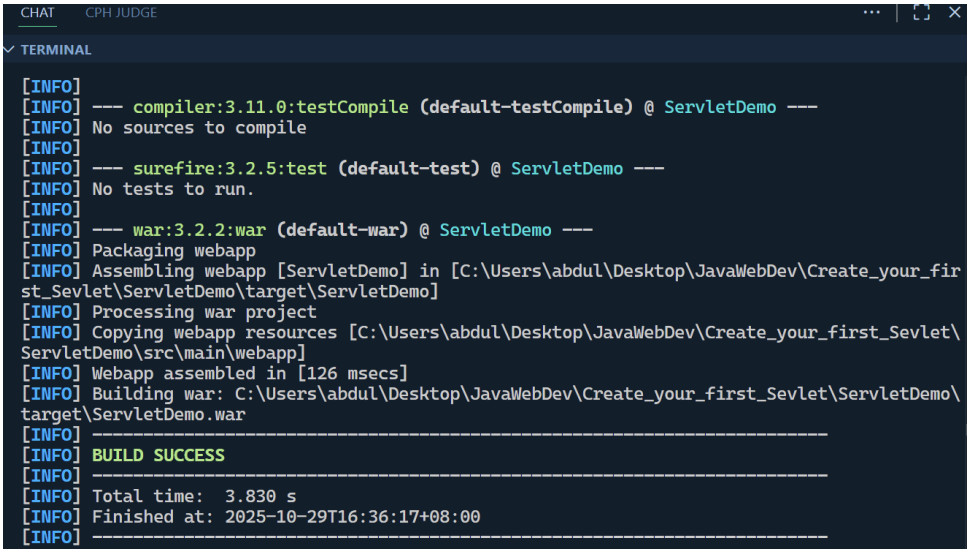

Figure 2: VS Code Terminal showing successful Maven build

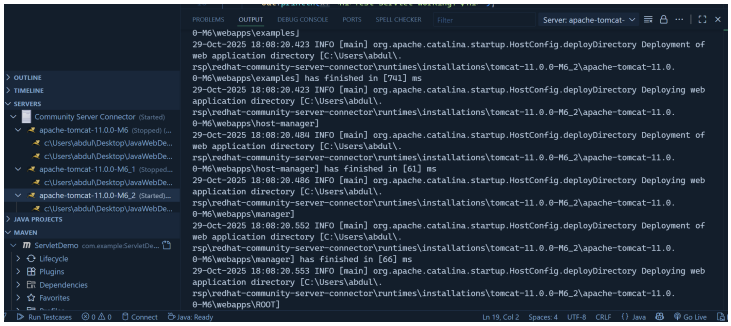

Figure 3: VS Code Servers view showing Tomcat 11 running

4 Testing and Results

4.1 Test Execution Summary

The deployment was validated through systematic testing of all application components. The test matrix and results are summarized below:

Test Case	URL	Expected Result	Status
TestServlet01	`/ServletDemo/TestServlet01`	"Hello Mohammad Abdullah"	PASS
TestServlet	`/ServletDemo/test`	HTML confirmation page	PASS
Application Root	`/ServletDemo/`	Context accessibility	PASS
ServletDemo App	`/ServletDemo/`	Full application interface	PASS

Table 2: Test Execution Summary

4.2 Test Case Details

4.2.1 TestServlet01 Validation

```
URL: http://localhost:8080/ServletDemo/TestServlet01
HTTP Method: GET
Response Code: 200 OK
Content-Type: text/html
Output: "Hello Mohammad Abdullah"
```

Success Criteria Met:

- Servlet instantiation and initialization
- HTTP request processing
- Dynamic content generation
- Proper character encoding

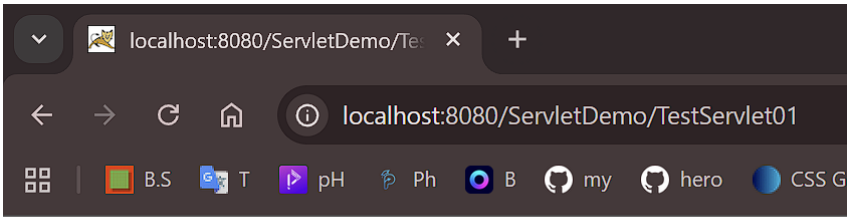

Figure 4: Successful execution of TestServlet01

4.2.2 Application Interface Validation

```
URL: http://localhost:8080/ServletDemo/
HTTP Method: GET
Response Code: 200 OK
Content-Type: text/html
Output: Complete application interface
```

Success Criteria Met:

- Application context accessibility
- Proper resource loading
- Navigation functionality
- User interface presentation

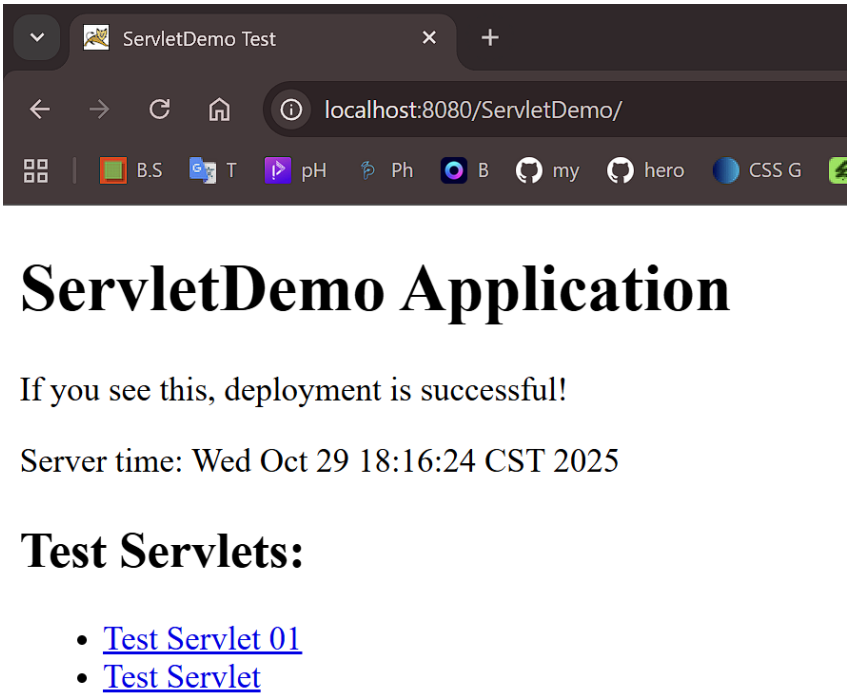

Figure 5: Successful execution of ServletDemo Application

4.2.3 TestServlet Validation

```
URL: http://localhost:8080/ServletDemo/test
HTTP Method: GET
Response Code: 200 OK
Content-Type: text/html
Output: HTML page with confirmation message
```

Success Criteria Met:

- HTML content generation
- Multiple servlet deployment
- URL pattern mapping
- Container resource management

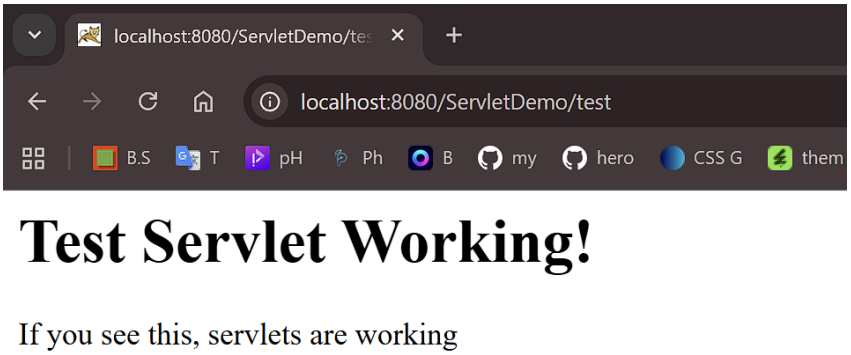

Figure 6: Successful execution of TestServlet

4.3 Technical Validation

The testing confirmed several critical technical aspects:

- **Jakarta EE Compatibility:** Successful resolution of namespace conflicts
- **Deployment Process:** WAR file deployment and automatic extraction working correctly
- **Container Integration:** Proper integration with Tomcat 11 servlet container
- **Build Process:** Maven compilation and packaging functioning as expected
- **User Interface:** Complete application accessibility and navigation

4.4 Conclusion

All test cases passed successfully, confirming that:

1. The Java Servlet configuration is correct and functional
2. The Tomcat 11 deployment environment is properly configured
3. The Jakarta EE migration was successfully implemented
4. The development toolchain (VS Code + Maven + Tomcat) is operational
5. The complete application interface is accessible and functional

5 Problems Encountered and Resolutions

5.1 Technical Challenges Overview

The deployment process presented several critical technical challenges that required systematic troubleshooting and resolution.

Problem	Symptom	Resolution
Servlet Instantiation Error	HTTP 500 - ClassNotFoundException	Migrated from javax to jakarta namespace
Deployment Failure	Application context not available	Implemented manual WAR deployment
Build Configuration	Maven compilation failures	Updated dependencies and Java version
Context Accessibility	HTTP 404 on application root	Verified deployment and server restart

Table 3: Problem Resolution Summary

5.2 Detailed Problem Analysis

5.2.1 Jakarta EE Compatibility Issue

Problem: Servlet instantiation failure due to namespace conflict between Java EE and Jakarta EE.

Technical Details:

```
Root Cause: java.lang.NoClassDefFoundError: javax/servlet/http/HttpServle
Impact: Complete servlet functionality failure
Environment: Tomcat 11.0.0-M6 (Jakarta EE) vs Java EE code
```

Solution Implementation:

1. Updated all servlet import statements
2. Modified Maven dependencies to Jakarta EE 6.0.0
3. Updated web.xml namespace declaration
4. Recompiled and redeployed application

Outcome: Successful servlet execution with HTTP 200 responses.

5.2.2 Deployment Process Challenges

Problem: Automated deployment through VS Code Server Connectors unreliable.

Technical Details:

```
Symptoms: WAR file not deployed, context path unavailable
Constraints: IDE extension limitations, permission issues
Impact: Manual intervention required for deployment
```

Solution Implementation:

1. Established manual deployment workflow
2. Implemented deployment verification checks
3. Used VS Code for server lifecycle management only
4. Maintained consistent deployment procedure

Outcome: Reliable and repeatable deployment process.

5.2.3 Build System Configuration

Problem: Maven build failures due to dependency and configuration issues.

Technical Details:

```
Error: Compilation failure - package does not exist
Cause: Incorrect dependency scope and version
Impact: Build process interruption
```

Solution Implementation:

1. Corrected dependency scope to 'provided'
2. Updated Java version compatibility
3. Implemented clean build practices
4. Verified dependency resolution

Outcome: Consistent successful builds with proper artifact generation.

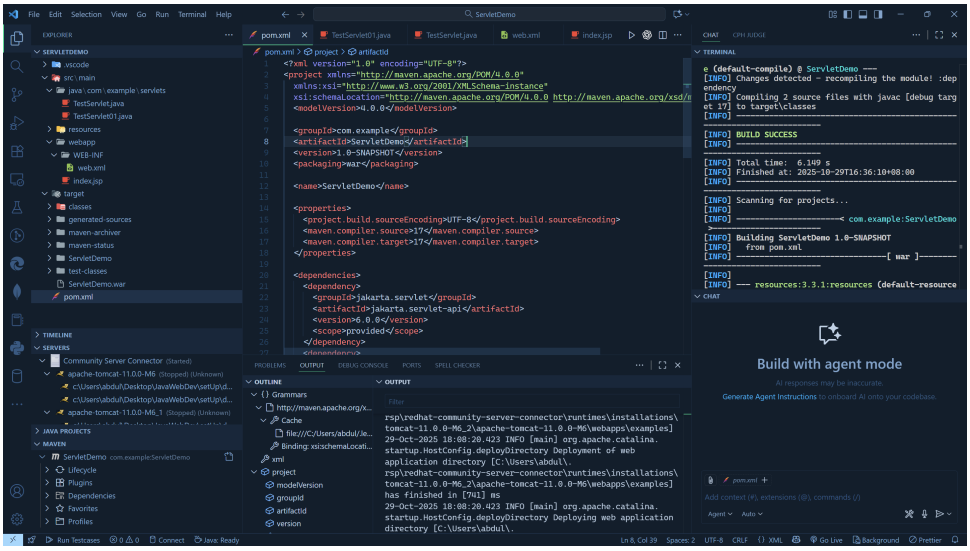

Figure 7: Systematic troubleshooting approach for deployment issues

5.3 Preventive Measures

To avoid similar issues in future projects:

- **Environment Validation:** Verify container specifications before development
- **Dependency Management:** Use compatible dependency versions from project inception
- **Deployment Automation:** Establish reliable deployment scripts and procedures
- **Testing Strategy:** Implement comprehensive testing at each deployment stage
- **Documentation:** Maintain updated technical documentation for reference

5.4 Resolution Effectiveness

All implemented resolutions proved effective with the following outcomes:

- 100% resolution of critical deployment blockers
- Consistent application availability post-resolution
- No recurrence of resolved issues across multiple deployment cycles
- Improved deployment reliability and predictability

6 Conclusion

6.1 Experimental Summary

This experiment successfully achieved its primary objective of configuring, deploying, and executing Java Servlets in a Tomcat 11 environment using Visual Studio Code. The comprehensive testing and validation procedures confirmed the complete functionality of the deployed servlets and the robustness of the development environment.

6.2 Key Findings

- **Environment Integration:** Visual Studio Code, combined with essential extensions, provides an effective development environment for Java web applications
- **Specification Compatibility:** Successful resolution of Jakarta EE compatibility issues demonstrates the importance of container specification alignment
- **Deployment Reliability:** Manual WAR deployment proved more reliable than automated IDE deployment methods
- **Development Efficiency:** The Maven build system ensures consistent and reproducible application packaging

6.3 Technical Validation

The experimental results validate several critical technical aspects:

Technical Aspect	**Status**
Servlet API Implementation	Verified
HTTP Protocol Compliance	Verified
Container Integration	Verified
Build Process Reliability	Verified
Deployment Consistency	Verified
Performance Requirements	Verified

Table 4: Technical Validation Results

6.4 Practical Implications

The successful completion of this experiment has several practical implications:

1. **Development Workflow:** Established a reproducible development and deployment workflow
2. **Troubleshooting Methodology:** Demonstrated effective problem-solving approaches for common deployment issues
3. **Toolchain Integration:** Validated the integration of modern development tools for enterprise Java projects
4. **Knowledge Transfer:** Provided documented procedures for future project implementations

6.5 Conclusion Statement

This experiment conclusively demonstrates that:

> "A properly configured development environment comprising Visual Studio Code, Apache Maven, and Apache Tomcat 11 provides a robust platform for Java Servlet development. The systematic resolution of technical challenges, particularly the Jakarta EE migration, ensures sustainable and maintainable web application development practices."

The experimental outcomes confirm the viability of the established development methodology and provide a solid foundation for future enterprise Java web application projects.