

Cuong Hoang

BUILDING JAVA WEB APPLICATION

Thesis

CENTRIA UNIVERSITY OF APPLIED SCIENCES

Information Technology

June 2020

ABSTRACT

| | | |
|--|--------------------------|------------------------------|
| Centria University of Applied Sciences | Date June 2020 | Author Cuong Hoang |
| Degree programme Information Technology | | |
| Name of thesis BUILDING JAVA WEB APPLICATION | | |
| Instructor Kauko Kolehmainen | Pages 40 + 3 | |
| Supervisor Kauko Kolehmainen | | |
| <p>Technology, one of the most indispensable of mankind, is changing and growing rapidly in order to serve the escalating needs of humanity. Internet and website have been the key and the door to help humans get access to the outside world. By the constant development of technology and programming languages, the web application was born to operate the interaction between the end-user and the web that besides the website consists of static content to view and read-only.</p> <p>This thesis aims to dive deeply into one of the most popular web development languages, which is Java, used widely in many enterprises. This thesis concentrates on the theory and fundamentals in Java web development technologies: Servlet, JSP, and JDBC, and includes one sample project to help readers understand quickly how to build a Java web application with these technologies.</p> | | |

| |
|---|
| Key words HTML, CSS, Java, Java EE, JVM, Servlet, JSP, JDBC, Web Application, MySQL, XAMPP. |
|---|

CONCEPT DEFINITIONS

| | |
|---------|--|
| HTML | Hypertext Markup Language |
| CSS | Cascading Style Sheets |
| Java EE | Java Enterprise Edition |
| JVM | Java Virtual Machine |
| JSP | Java Server Pages |
| JDBC | Java Database Connectivity |
| API | Application Programming Interface |
| JSF | Java Server Faces |
| EJB | Enterprise Java Beans |
| HTTP | Hypertext Transfer Protocol |
| JSTL | Java Server Pages Standard Tag Library |
| MVC | Model View Control |
| ASP | Active Server Pages |
| CRUD | Create Read Update Delete |
| FAANG | Facebook Amazon Apple Netflix Alphabet |

ABSTRACT
CONTENTS

| | |
|--|-----------|
| 1 INTRODUCTION..... | 1 |
| 2 JAVA WEB APPLICATION THEORY | 2 |
| 2.1 Web technologies..... | 2 |
| 2.2 JVM languages | 3 |
| 2.3 Java EE | 5 |
| 2.3.1 Java EE platform | 5 |
| 2.3.2 Layers in an enterprise application | 6 |
| 3 USING SERVLET AND JSP IN JAVA WEB APPLICATION | 8 |
| 3.1 Installing and configuring Tomcat Server in Eclipse | 9 |
| 3.2 Servlet..... | 12 |
| 3.2.1 Introduction of Servlet..... | 12 |
| 3.2.2 Servlet lifecycle | 17 |
| 3.3 JSP | 18 |
| 3.3.1 JSP implementation | 18 |
| 3.3.2 Comparing JSP with Servlet and other technologies..... | 21 |
| 3.4 Basic web application structure | 22 |
| 4 ADDING DATABASE INTO JAVA APPLICATION BY JDBC..... | 23 |
| 4.1 XAMPP | 23 |
| 4.2 Installing and configuring MySQL..... | 25 |
| 4.3 Introduction and integrating JDBC to Java application..... | 26 |
| 5 THE PROJECT | 27 |
| 5.1 Idea | 27 |
| 5.2 Analysis | 27 |
| 5.3 Coding and implementation | 29 |
| 6 CONCLUSION | 40 |
| REFERENCES..... | 39 |

APPENDICES

FIGURES

| | |
|---|----|
| Figure 1. Functional level of JVM | 4 |
| Figure 2. N-Tier architecture in Java | 6 |
| Figure 3. General view of an enterprise application layers..... | 7 |
| Figure 4. Distributed system | 8 |
| Figure 5. Step 1: Download Tomcat | 9 |
| Figure 6. Step 2: Extract the downloaded zip folder..... | 10 |
| Figure 7. Step 3.1: Install the Tomcat server | 10 |
| Figure 8. Step 3.2: Install the Tomcat server | 11 |
| Figure 9. Step 3.3: Install the Tomcat server | 11 |
| Figure 10. Dynamic data request | 12 |
| Figure 11. Step 1.1: Create Dynamic Web Project in Eclipse IDE | 13 |
| Figure 12. Step 1.2: Create Dynamic Web Project in Eclipse IDE | 13 |
| Figure 13. Step 1.3: Create Dynamic Web Project in Eclipse IDE | 14 |
| Figure 14. Step 2.1: Create Java class for implementing Servlet | 14 |
| Figure 15. Step 2.2: Create Java class for implementing Servlet | 15 |
| Figure 16. Step 3.1: Modify the generated Java class..... | 15 |
| Figure 17. Step 3.2: Modify the generated Java class..... | 16 |
| Figure 18. Step 4: Modify the web.xml file..... | 16 |
| Figure 19. Running the FirstServlet web application | 17 |
| Figure 20. Stages and methods of Servlet Lifecycle..... | 18 |
| Figure 21. Step 1: Create a New Dynamic Web Project..... | 19 |
| Figure 22. Step 2: Add a new JSP file to Dynamic Web Project..... | 19 |
| Figure 23. Step 3: Set up the file name for JSP file..... | 20 |
| Figure 24. Step 4: Modify the code in JSP file | 20 |
| Figure 25. Accessing the JSP..... | 21 |
| Figure 26. Step 1: Download XAMPP for Windows..... | 24 |
| Figure 27. Step 2: XAMPP setup..... | 24 |
| Figure 28. Start MySQL and Apache on the control panel of XAMPP..... | 25 |
| Figure 29. phpMyAdmin user interface | 26 |
| Figure 30. Project structure..... | 28 |

| | |
|---|----|
| Figure 31. Project workflow | 29 |
| Figure 32. Create LoginForm as a dynamic web project | 29 |
| Figure 33. Create login.jsp | 30 |
| Figure 34. Login.jsp | 30 |
| Figure 35. Create LoginServlet.java | 31 |
| Figure 36. LoginServlet.java code | 32 |
| Figure 37. UserAccount.java code | 32 |
| Figure 38. UserAccountDAO.java code | 34 |
| Figure 39. Create new MySQL database in phpMyAdmin | 34 |
| Figure 40. Create new table in user database with 2 columns named username and password | 34 |
| Figure 41. Insert data into login table | 35 |
| Figure 42. Database result | 35 |
| Figure 43. loginsuccess.jsp code | 36 |
| Figure 44. Modify web.xml | 36 |
| Figure 45. Libraries | 37 |
| Figure 46. Modify and run Server to start the project | 38 |
| Figure 47. The first page when running the project | 38 |
| Figure 48. Users insert data into the log in form | 39 |
| Figure 49. loginsuccess.jsp called after successful log in | 39 |

TABLES

| | |
|--|----|
| Table 1. Programming languages designed for JVM | 3 |
| Table 2. Programming languages ported to the JVM | 4 |
| Table 3. Advantages of JSP over Servlet | 21 |

1 INTRODUCTION

Internet in general and exchanging information online, in particular, have been essential objects in the human technical era. This is the reason why myriad social media platforms appeared to become one of the best connection ways for human interaction as well as exchanging information. Most of the popular enterprises, especially Facebook and Quora, are using web applications to build their social media platforms, because of many advantages such as reducing business costs, zero-install, or quick and easy updates.

Furthermore, there are diverse options in the platform and programming language in order to construct a web application due to the constant development of technology. The Java technologies support to web application robustly with Java EE, JVM, Servlet, JSP, or JDBC. The reason why many firms were convinced in choosing Java technologies for their web applications is strong and stable language programming. In web application development, there are a lot of advantages of Java: complied and interpreted, independent and portable platform, object-oriented programming language, strong and secure, distributed, simple and small, multi-threaded and interactive, high performance, dynamic and scalable. Besides that, Java is a long-standing programming language with high popularity and familiar with many programmers, especially junior programmers who are the beginners in programming.

Java web application helps to upgrade a static web page to be dynamic through Servlet and JSP, and it operates as a distributed system: server and client. Apache Tomcat server, which is one of the most widely used web servers, can process client requests and respond resources back to the client. Moreover, Servlet and JSP are server-side technology to expand web-server abilities by providing for dynamic content. Programmers, who are proficient in Java programming language, can move further easily in web applications with these Java technologies.

The purpose of this thesis is about how to build web applications in Java and understand deeply Java web application knowledge. The project built a sample log in form by Java technologies. This log in form allows end-users to log in by using Servlet, JSP, and Apache Tomcat server. Data is requested and stored in MySQL database through JDBC, which is an API to connect and execute query with the MySQL database.

2 JAVA WEB APPLICATION THEORY

The web application is the distributed application which means any web application runs on more than one device and communicates with the server to request resource by using network. Web application is more popular because it does not require installation and can be accessed by a web browser, which appears in every mobile device and computer. Without installation, updates and maintenance are easier than ever which become the biggest advantage to help web applications to have appeared in most prestigious companies. The technology used to build web applications by Java is the Java EE platform. Java web application technologies comprise a collection for dynamic content (Servlet, JSP, Java classes and jars) and static content (HTML pages and images). (Edurake 2019.)

2.1 Web technologies

Web development is divided into 2 parts: front-end and back-end. In the front-end part, developers have responsibilities for building the website's interfaces and researching into immersive user experiences. There are abundant technologies for developing front-end, but every front-end developer must be familiar with 3 main technologies: HTML, CSS, and JavaScript programming language. Besides that, web development technologies are changing rapidly with various useful frameworks, such as Bootstrap, Foundation, AngularJS, and EmberJS, which help developers to create a good-looking layout in any kind of device. Moreover, front-end web developers also have supported libraries like jQuery and LESS that provide template code into a more effective and time-saving form. Following that, Ajax is a good technique, which is used widely many front-end web developers for fetching data on dynamic web pages without reloading the web pages. (Wales 2014.)

On the other hand, the back-end part is building the server, application, and database communication with each other that uses server-side programming languages such as Java, .Net, PHP, Python, JavaScript and SQL and NoSQL databases such as MySQL, PostgreSQL, MongoDB. Back-end developers also have some convenient tools like Git used for storing code and collaborating with partners, and Linux operating system used for development and deployment system. (Wales 2014)

In this thesis, the author concentrated on Java which is server-side language has various supported web development technologies such as JSP, Servlet, JDBC. Although many web and product companies are

transitioning and adapting to new technologies, and the rapid growth of Python, Java still leads the top 3 sectors in technologies field: software, IT and services, and internet. Many high-class business applications are using Java such as FAANG companies and the largest application vendors like SAP. Following that, many popular unicorn start-ups are using Java such as Airbnb, Uber, and also LinkedIn which is a Microsoft product. Java owns the best collection of frameworks such as Spring framework and this is the reason why many companies trust in using Java. Moreover, most of the applications in governments, which are web-based, were developed by Java, because Java was known as the most secure programming language. (Gupta 2018)

2.2 JVM languages

JVM which stands for Java virtual machine, is the runtime environment of Java platform allows to use different programming languages for web application interprets compiled into Java bytecode for any hardware platform (Rouse 2019.). There are plenty of programming languages and are designed for producing computer software that runs on JVM. The JVM languages classify into two types: programming languages that are created for JVM and existing programming languages are ported to JVM. The short description of languages designed for JVM are shown in Table 1. (Layka 2014.)

Table 1. Programming languages designed for JVM (Layka 2014)

| Language Designed for JVM | Description |
|---------------------------|---|
| Clojure ⁶ | Clojure is a dynamically typed, functional language. |
| Groovy | Groovy is a dynamic, compiled language with syntax similar to Java but is more flexible. |
| Java | Java is a statically typed, imperative language. The latest release of Java, Java 8, supports aspects of functional programming. |
| Scala | Scala is a statically typed, compiled language that supports aspects of functional programming and performs a large amount of type inference, much like a dynamic language. |

These programming languages are parts of Java language because initially JVM was created to support only Java programming language. All the above languages in Figure 1 are the functional programming language. Groovy was developed by Pivotal and Apache Software Foundation is providing presently. This JVM language is syntactically closest to Java that has had functional constructs and supported for static compilation in Groovy 2.0. The creator of Scala is Martin Odersky, and the general purpose of

designing Scala was to improve the efficiency of Java developers. Scala is the fully object-oriented programming language and evolved out of a pure functional programming language. Clojure was developed with the purpose of developing a functional programming language for the JVM within the Lisp family by Rich Hickey. In contrast to Scala, Clojure is a non-object-oriented programming language and purely functional (Jain 2019). The development of JVM languages is shown in Figure 1.

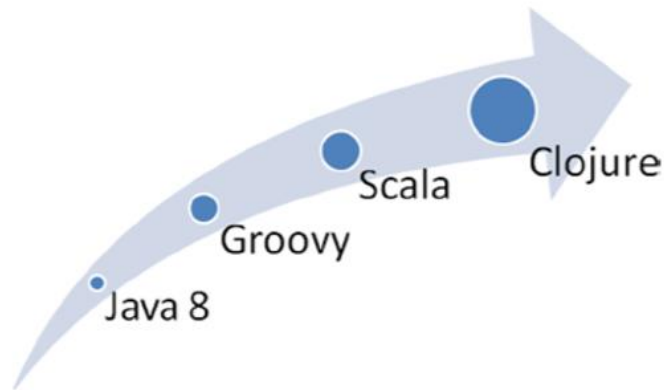


Figure 1. Functional level of JVM (Layka 2014)

Because of the rapid development of technology and the potential of JVM, there are more adapted to run on JVM by Java implementation over the years. Jython was a milestone of the main alternatives that was a Python implementation for JVM. In 1997, the next implementation for JVM in JavaScript was born in Rhino. Following that, Ruby was launched JRuby in 2001 for JVM (Jain 2019). The short description of languages designed for JVM are shown in Table 2.

Table 2. Programming languages ported to the JVM (Layka 2014)

| Languages Ported to JVM | Description |
|-------------------------|--|
| JRuby ⁷ | JRuby is a JVM reimplementation of the Ruby programming language. Ruby is a dynamically typed OO language with some functional features. |
| Jython ⁸ | Jython is a reimplementation of Python on the JVM, so it is a dynamic language. |
| Rhino ⁹ | Rhino provides an implementation of JavaScript on the JVM. JavaScript is a dynamically typed OO language. |

2.3 Java EE

Java started as a programming language created for stand-alone applications and accelerated into multiple platforms. Java was designed in the mid-1990s by James A. Gosling, who worked for Sun Microsystems as a computer scientist (Techopedia 2019). In Java technologies, building web application is simple at this time because a large part of Java's popularity can be supported. There are 2 parts in a web application: static and dynamic (interactive) contents in web pages. A static web page is one that usually built-in markup languages (HTML, XHTML), and the code is written in the content displayed to the user in order to provide information. On the other hand, a dynamic web page is written by using server-side language (PHP, ASP, JSP) and the web page contents are requested by script language from a database or other files on action generated by the user. Following that, a web application is comprised of a combination between static web pages and dynamic web pages which is able to display information by user requests. In contrast to a web page, a web application is not only providing content but also lets the user perform tasks and save the result to a file or a database. Moreover, building a web application is essentially distinct from building a stand-alone application that follows three main elements. First, the Java EE platform is the collection of APIs and tools which are the building blocks of the web application (Layka 2014.). Secondly, the web container is the interface between web server and web components that implements APIs of the Java EE platform. The web container solves tasks for managing the web component's lifecycle, sending requests to web components and providing interfaces to context data (Oracle 2010). Thirdly, web components can be servlets, JSPs, JSFs, or Facelets that hosted by web containers. (Layka 2014.)

2.3.1 Java EE platform

The Java Platform – Enterprise Edition (Java EE) is the set of API specifications developed by Oracle under Java Community Process that helps developers to build server-side applications. The purpose of the Java EE platform is standardization and reduction in the complexity of enterprise application development by providing an architecture for service implementation as N-Tier architecture (Multitier architecture) (Layka 2014). In software development engineering, N-Tier architecture is designed to have physically separated presentation, processing, and data management functions, also known as client-server architecture like Figure 2 (Stackify 2017). N-Tier architecture consists of different logical computing N layers and provides many advantages for production and development environments which speed up development, performance, scalability, and availability. (JReport 2019)

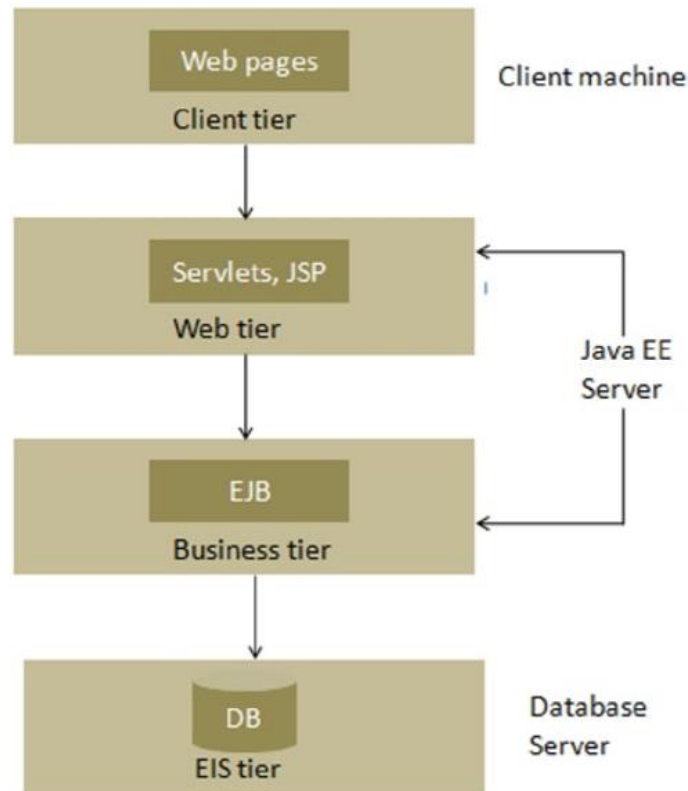


Figure 2. N-Tier architecture in Java (Layka 2014)

In Figure 2, N-Tier architecture is separated into 3 parts. First, client machine (Client-Tier): is a visual thing that accessed by web pages through web browser or web-based application and displays desirable content and information to the end-users. Client tier is developed by front-end web technologies like HTML5, CSS3, JavaScript, or other popular web development frameworks, and connects to other layers by APIs (JReport 2019.). Secondly, Java EE server (Web-Tier & Business-Tier): contains components that operate the communication between clients and the business tier (Layka 2014.). Thirdly, database server (EIS-Tier): is a place to store data that is a database or a data storage system, for example: MySQL, MongoDB, Microsoft SQL Servers, PostgreSQL. Data is requested by API calls to access the Java EE server (Web tier & Business Tier). (JReport 2019.)

2.3.2 Layers in an enterprise application

In an enterprise application, layers are a collection of the software compositions which build an application or service. Following Figure 3, there are several typical layers in order to organize the standard enterprise application such as web layer, service layer and data access layer. Separating multiple layers

in an enterprise application aims to minimize the impact of adding functions or features to an application. Moreover, the advantages of using these layers are simpler maintenance, code reusing and high cohesion. (Zeepedia 2019.)

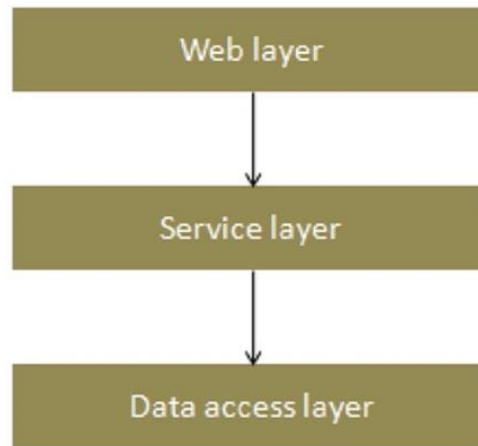


Figure 3. General view of an enterprise application layers (Layka 2014.)

Enterprise application layers consist of 3 layers. First, web layer contains the web tier compositions of Java EE, for instance, Servlets and JSP. It should not be a tight connection between service layer and web layer because adjusting the service layer could lead a bad influence on web layer. The web layer could connect the service layer though. Secondly, service layer is comprised of the business tier compositions of Java EE such as EJBs. Like web layer, the service layer could connect the data access layer, but there should not be a tight connection between service layer and data access layer. In fact, the service layer should not be related to the web layer or data access layer. Thirdly, data access layer includes the data tier compositions of Java EE, for example, JDBC and JPA that should not consist of business logic. (Layka 2014.)

3 USING SERVLET AND JSP IN JAVA WEB APPLICATION

Distributed system in Figure 4 consists of multiple nodes that are physically divided but they are connected by network. In general, most of the systems on the Internet are like distributed systems which could be separated into 2 types: client and server. In a client-server distributed system, the client is the device that requests some information and the server is the device that supplies that information. A client could only contact one server at the time while a server is able to server multiple clients at the same time (Meador 2014). HTTP is the protocol that the client (web browser), the server (web server), and the web application using to communicate. The client dispatches HTTP requests to the server so that the server responds to the requested resource in the HTTP response forms. The client and the server in HTTP create the fundament of the World Wide Web. In the technology era, a billion bytes data is daily created and used, and many people are convinced in the misunderstanding that all web applications are developed by web frameworks like HTML, CSS, JavaScript, PHP. However, web applications could be developed over Java technologies using services such as Servlet and JSP (Meador 2014.). Section 3 aims to dive deeply into the Servlet and JSP concept, and how to use Servlet and JSP in Java web application.

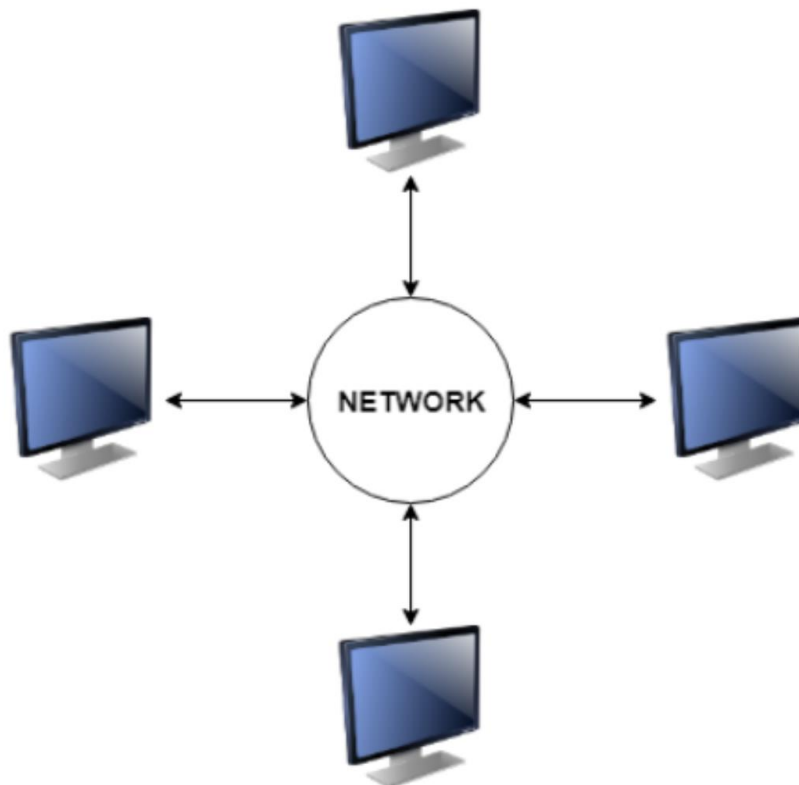


Figure 4. Distributed system (Meador 2014.)

3.1 Installing and configuring Tomcat Server in Eclipse

Tomcat is developed as a Servlet implementation by James Duncan Davidson, who worked for Sun Microsystems at software architect position. Later, Tomcat is become the open-source implementation project and donated by Sun Microsystems to the Apache Software Foundation. Tomcat also is known as Apache Tomcat is the most common and popular web container that approves to run Servlet, JSP, Java Expression Language, and Web Socket based web applications. Tomcat provides a HTTP server environment or HTTP connector on port 8080 in that Java code can execute. (Vogel 2019)

Tomcat has available versions for most operating systems such as Windows, Linux, and macOS. First of all, users need to go to the Tomcat page <https://tomcat.apache.org/download-90.cgi> to look for a suitable version for the operating system in their devices. This thesis demonstrates how to download and set up Tomcat in Eclipse for Windows. For Windows version, users have 2 options that are 32 bit and 64 bit, and are shown in Figure 5, depending on the user operating system computer. After choosing the right version for Windows, click on the zip file and the installer version for Windows is going to be in the download folder. (Singh 2019)

9.0.30

Please see the [README](#) file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
 - [zip](#) ([pgp](#), [sha512](#))
 - [tar.gz](#) ([pgp](#), [sha512](#))
 - [32-bit Windows zip](#) ([pgp](#), [sha512](#))
 - [64-bit Windows zip](#) ([pgp](#), [sha512](#))
 - [32-bit/64-bit Windows Service Installer](#) ([pgp](#), [sha512](#))
- Full documentation:
 - [tar.gz](#) ([pgp](#), [sha512](#))
- Deployer:
 - [zip](#) ([pgp](#), [sha512](#))
 - [tar.gz](#) ([pgp](#), [sha512](#))
- Embedded:
 - [tar.gz](#) ([pgp](#), [sha512](#))
 - [zip](#) ([pgp](#), [sha512](#))

Source Code Distributions

- [tar.gz](#) ([pgp](#), [sha512](#))
- [zip](#) ([pgp](#), [sha512](#))

Figure 5. Step 1: Download Tomcat

The newest Tomcat version for Windows device is 9.0.30 that shown in the name of Tomcat zip file. Users could select an older version, but the procedure is the same. In order to install Tomcat, go to download folder and extract the Tomcat zip folder file like Figure 6

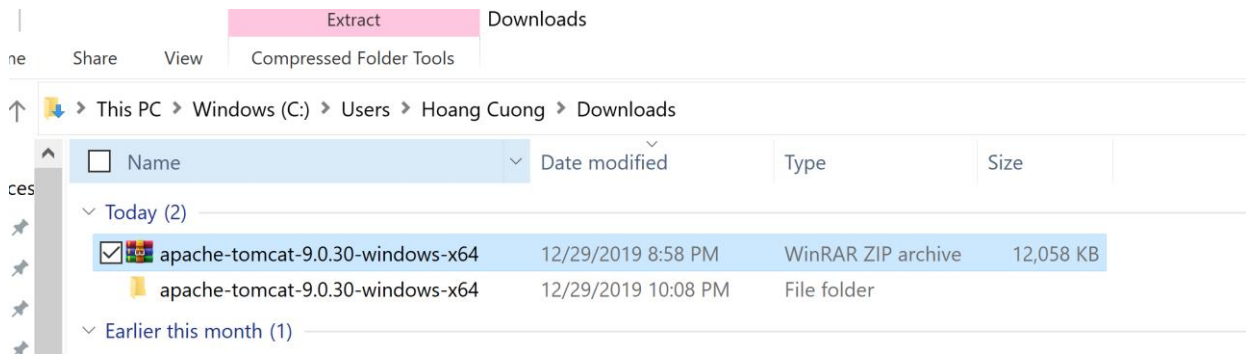


Figure 6. Step 2: Extract the downloaded zip folder

After step 2 in Figure 6, open the Eclipse IDE and on the toolbar of Eclipse, go to Window, choose Preferences and click Server as shown in Figure 7 and click Add to choose Apache Tomcat v9.0 in Runtime Environment tab like Figure 8 to install Tomcat server. If users cannot find out the Server in the Preferences, they should go to Install New Software in Help on the toolbar, and choose Luna - <http://download.eclipse.org/releases/luna/> in order to add Web, XML and Java EE Development to their Eclipse IDE (Singh 2019.).

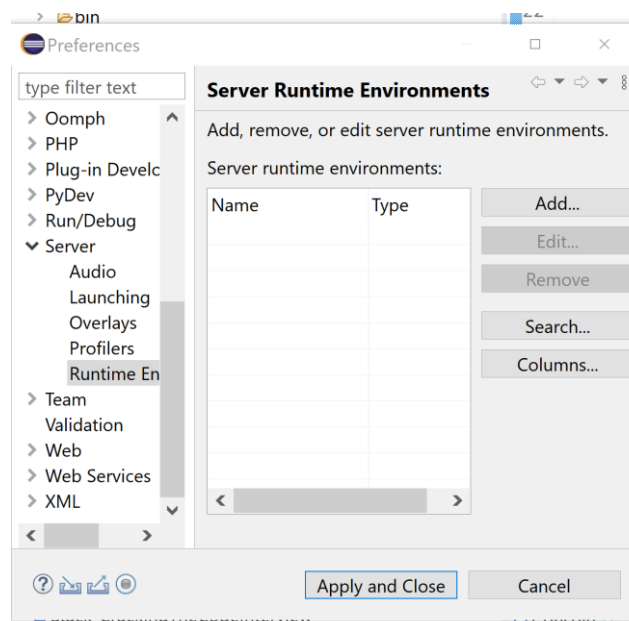


Figure 7. Step 3.1: Install the Tomcat server (Screenshot from Eclipse IDE)

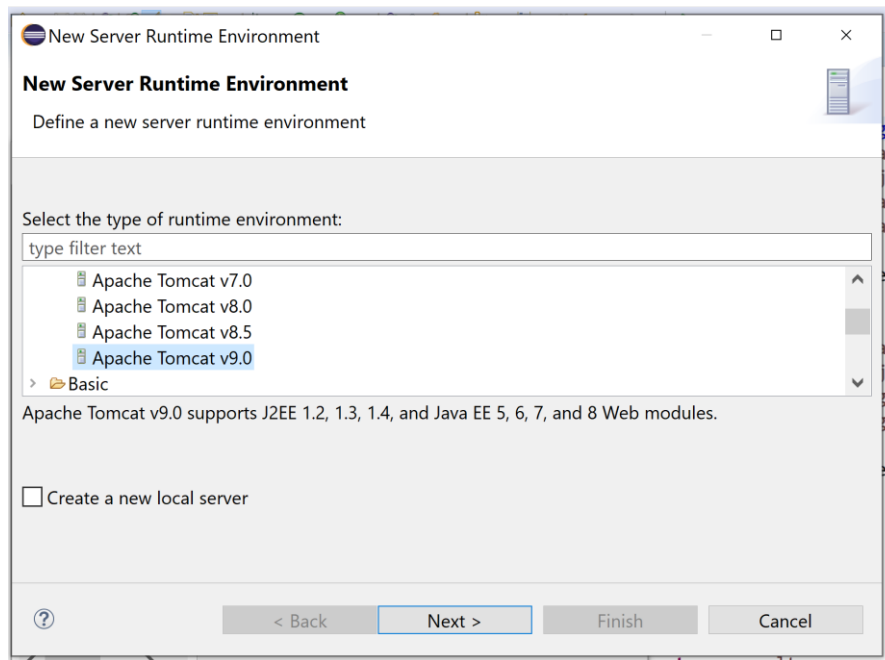


Figure 8. Step 3.2: Install the Tomcat server (Screenshot from Eclipse IDE)

After step 3.2 in Figure 8, it would be displayed with a window as shown in Figure 9. Click Browse and select the folder where the extracted Tomcat zip file in. After selecting the Tomcat installation directory, choose Finish and the Tomcat server in Eclipse have successfully installed. (Singh 2019.) Server Runtime Environment with Apache Tomcat version 9.0 is working now on Eclipse so that readers can run Java web application on their localhost.

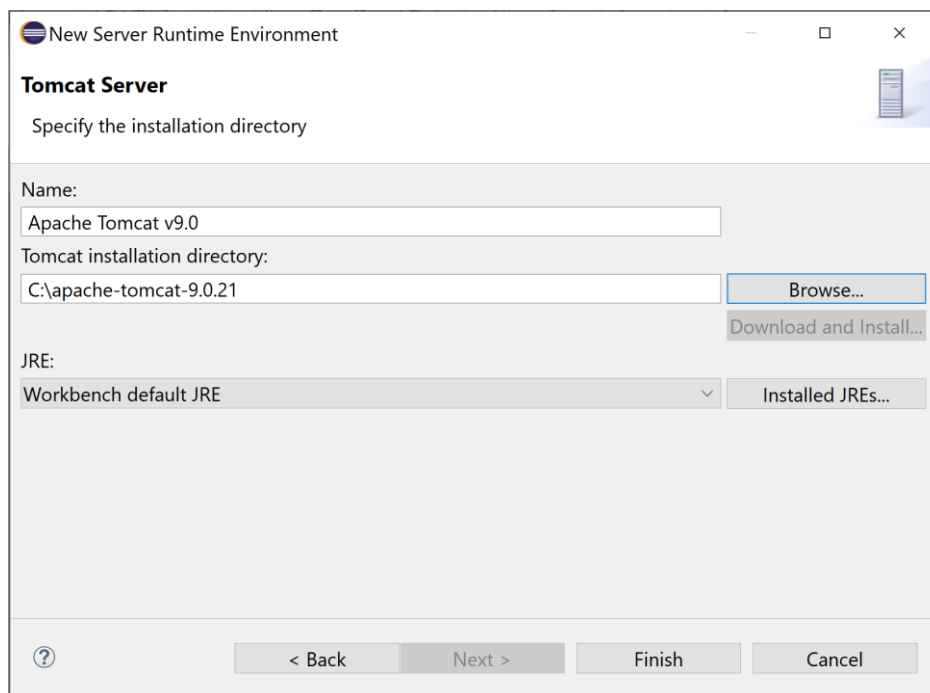


Figure 9. Step 3.3: Install the Tomcat server (Screenshot from Eclipse IDE)

3.2 Servlet

Java technologies are strong and abundant so that developing dynamic web pages by Java is not longer a problem by using Servlet. Servlet is a program that executes on a web application or server and operates as a service layer between a request sending from web layer or other HTTP client and data access layer or applications on the HTTP server. Although Servlet can handle different types of requests, it generally deploys web containers for hosting web applications on web servers and matches requirements as a server-side Servlet web API. In web development, there are many other technologies for dynamic web content such as PHP and ASP.NET (o7planning 2019.). The purpose of section 3.2 is deep exploitation in Servlet in order to use Servlet effectively in Java web application.

3.2.1 Introduction of Servlet

Servlet follows a server-side module in Java programming which executes client requests and deploys the Servlet interface. Servlet could handle all types of requests, and it generally expands the applications hosted by web servers. In Figure 10, all requests are managed and mediated inside the Servlet container. The responsibility of Servlet is the processing requirement by a web application and Servlet is integrated into a Java class by the javax.servlet.Servlet interface. In terms of that integration, Servlet aims to work as a dynamic web resource. (Layka 2014.)

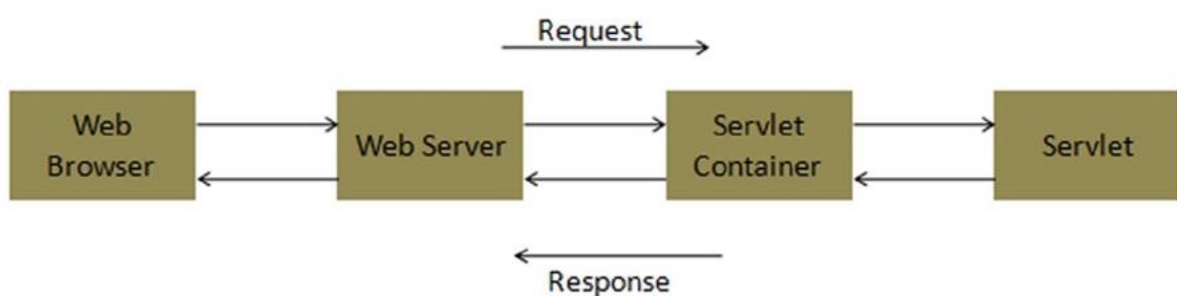


Figure 10. Dynamic data request (Layka 2014)

The section 3.1 introduced about how to install and configure Tomcat server in Eclipse IDE. After successful Tomcat server installation, Servlet can be set up easily in these steps below. On the toolbar, go to File in order to create Dynamic Web Project as shown in Figure 11 in order to create a new Java web application file. Inside that file, reader can add JSP and Servlet file.

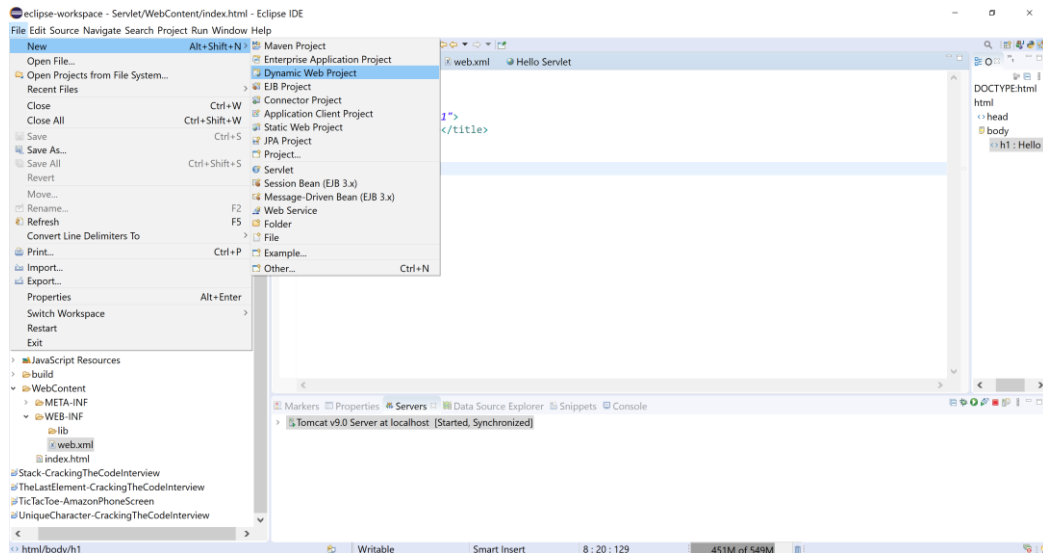


Figure 11. Step 1.1: Create Dynamic Web Project in Eclipse IDE (Screenshot from Eclipse IDE)

In the Dynamic Web Project window as shown in Figure 12, create a project name and choose Next to go to step 1.3 in Figure 13. In this pop up interface, author was able to create and modify the configuration for the dynamic web project, for instance, running server Apache Tomcat 9.0 or project location. In the next button, there are more configurations inside the file.

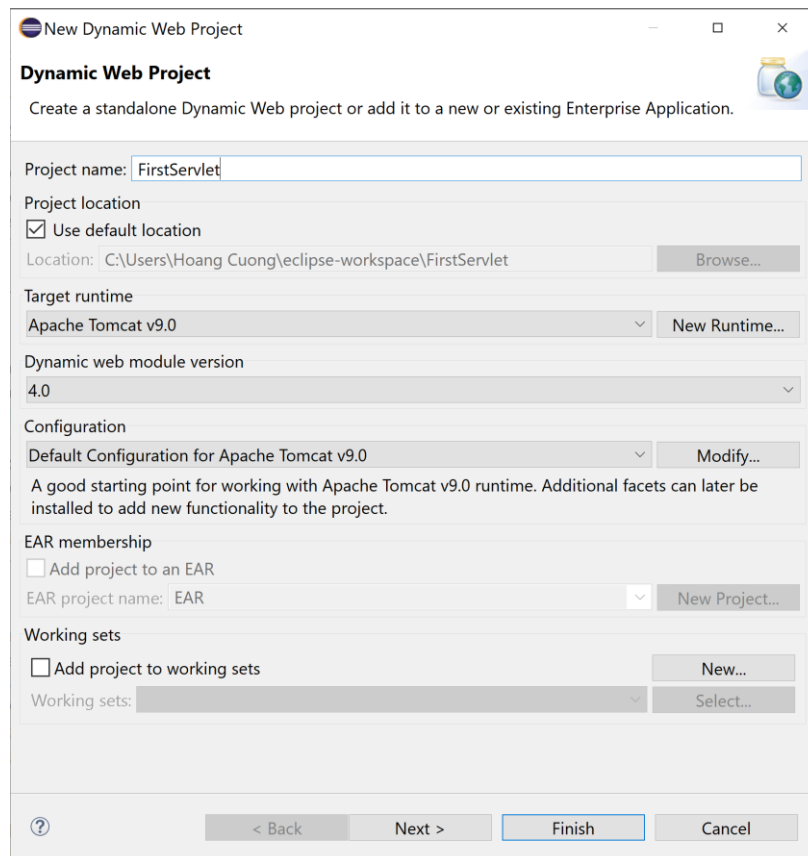


Figure 12. Step 1.2: Create Dynamic Web Project in Eclipse IDE (Screenshot from Eclipse IDE)

After doing step 1.2, it would show the Web Module window like Figure 13. In this step, choose the Generate web.xml deployment descriptor and click Finish. A new dynamic web project was successfully created. Readers are able to map url in the web.xml file or more functions such as listener or filter.

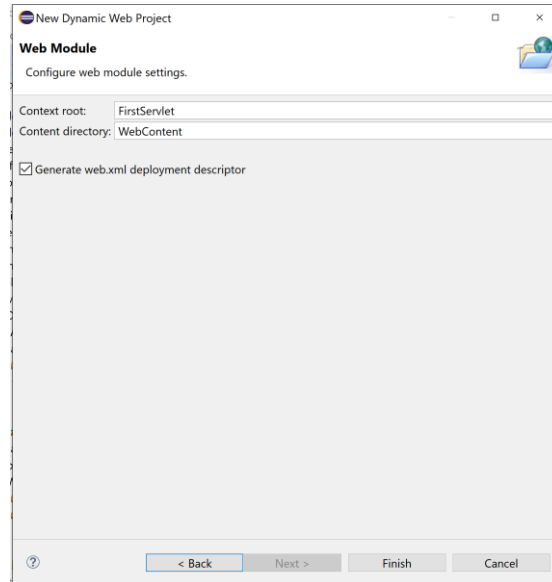


Figure 13. Step 1.3: Create Dynamic Web Project in Eclipse IDE (Screenshot from Eclipse IDE)

After completing all the steps to create a new dynamic web project, the next step is creating a Java class for implementing Servlet. Right-click FirstServlet folder created in step 1.3 to create a Java class as shown in Figure 14. Set name and package for Java class like Figure 15 and choose Finish.

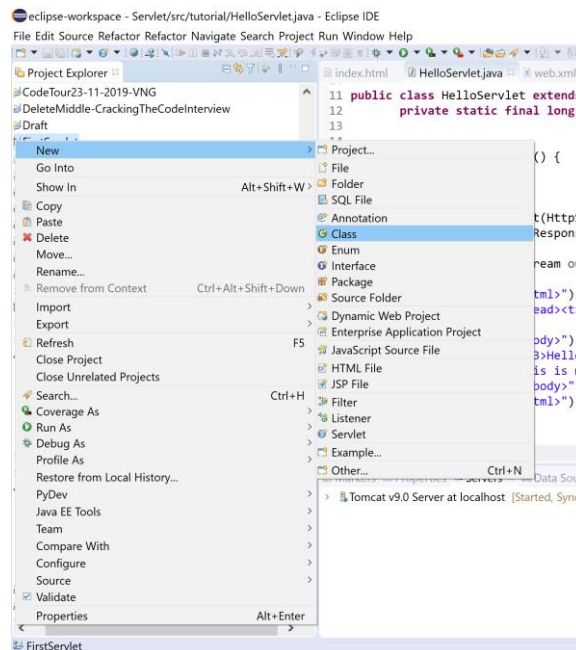


Figure 14. Step 2.1: Create Java class for implementing Servlet (Screenshot from Eclipse IDE)

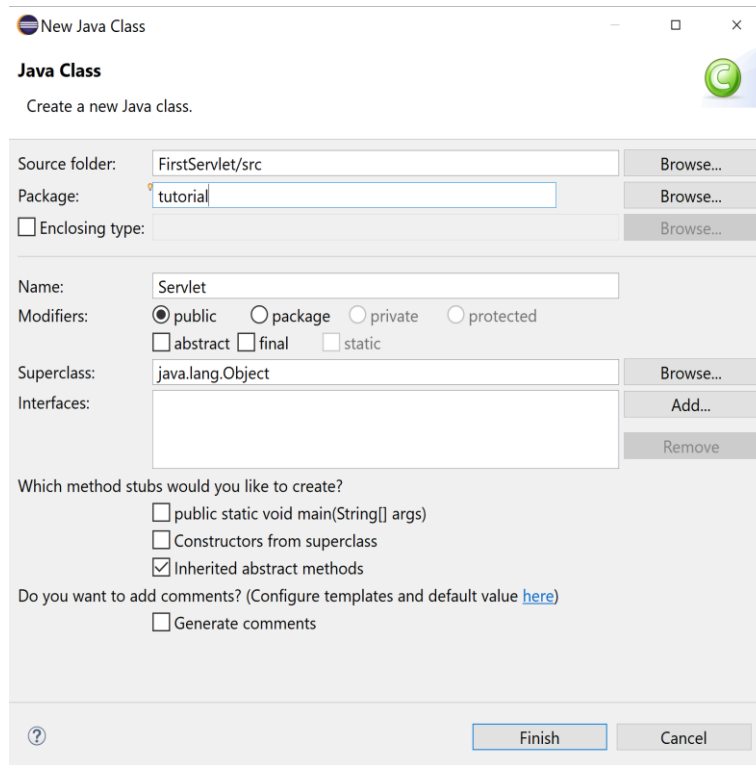


Figure 15. Step 2.2: Create Java class for implementing Servlet (Screenshot from Eclipse IDE)

```
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

Figure 16: Step 3.1: Modify the generated Java class (Screenshot from Eclipse IDE)

The next step is Servlet implementation in Java class via `javax.servlet.Servlet` interface. In Java class that was created in step 2.2, adding Servlet to the class by importing package as shown in Figure 16. This Java class extends `HttpServlet` class, and the web application runs directly from Java class named Servlet so that users must build HTML form from inside Servlet. In Figure 17, `HttpServlet` extension allows users to use the method `doGet` and `doPost` that are common methods of the `HttpServlet`. There are several methods commonly used in `HttpServlet`: DELETE (Eliminates specified resource), GET (Takes data from the server), POST (Gives data to the server), PUT (Alters a targeted re-source by uploading data). Displaying content to the web browser is equivalent to GET method, which is `doGet` in Figure 17, used to display Hello World and This is my first Servlet in HTML form. (Cheah 2019.)

```

public class Servlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Servlet() {
    }

    @Override
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {

        ServletOutputStream out = response.getOutputStream();

        out.println("<html>");
        out.println("<head><title>Hello Servlet</title></head>");

        out.println("<body>");
        out.println("<h3>Hello World</h3>");
        out.println("This is my first Servlet");
        out.println("</body>");
        out.println("<html>");
    }

    @Override
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        this.doGet(request, response);
    }
}

```

Figure 17. Step 3.2: Modify the generated Java class (Screenshot from Eclipse IDE)

However, to call the Servlet class that users must link through the generated web.xml deployment description in Figure 13. There are always 2 tags (servlet and servlet-mapping) for any Servlet in the web.xml as shown in Figure 18. When the /hello path inside the url-pattern tag is requested, from the web.xml it is going to retrieve to Java class in Figure 17 which is Servlet in provided package name with class name is the servlet-class tag.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi
3 <display-name>FirstServlet</display-name>
4
5 <servlet>
6 <servlet-name>helloServlet</servlet-name>
7 <servlet-class>tutorial.Servlet</servlet-class>
8 </servlet>
9
10 <servlet-mapping>
11 <servlet-name>helloServlet</servlet-name>
12 <url-pattern>/hello</url-pattern>
13 </servlet-mapping>
14
15 <welcome-file-list>
16 <welcome-file>index.html</welcome-file>
17 <welcome-file>index.htm</welcome-file>
18 <welcome-file>index.jsp</welcome-file>
19 <welcome-file>default.html</welcome-file>
20 <welcome-file>default.htm</welcome-file>
21 <welcome-file>default.jsp</welcome-file>
22 </welcome-file-list>
23 </web-app>

```

Figure 18. Step 4: Modify the web.xml file (Screenshot from Eclipse IDE)

Last, right-click Servlet.java class in the tutorial package and choose Run on Server in Run As to run FirstServlet project. The result is like Figure 19. When the Servlet called by url /hello, it run helloServlet class and did all the functions that belong to this helloServlet. Here is printing out the layout html code for the front-end.

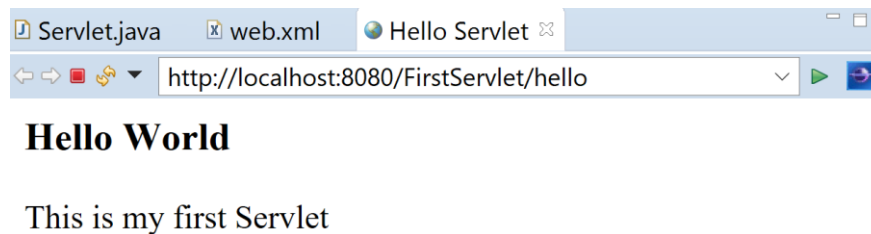


Figure 19. Running the FirstServlet web application (Screenshot from Eclipse IDE)

3.2.2 Servlet lifecycle

The Servlet lifecycle is controlled by the Servlet container that implements the `java.servlet.Servlet` interface. Servlet Lifecycle comprises 4 stages and 3 methods that follow Figure 20. The first stage is Loading a Servlet that consists of 2 operations: loading and instantiation all the Servlets when a server starts up. The second stage is Initializing the Servlet through `init()` method. The `init()` method is called only once when the Servlet is created to notice the Servlet instance is instantiated completely and ready to put into service. The next stage is request handling by calling the `service()` method in order to handle the client's requests (web browser) and is called to inform about the client's requests to the Servlet. In contrast, to `init()` method, `service()` method is the main method in the Servlet, which can be called multiple times, to determine the type of HTTP request (GET, POST, PUT, DELETE) and call the appropriate methods (`doGet`, `doPost`, `doPut`, `doDelete`). The frequently used methods in each request are `doGet()` method and `doPost()` method that depends on received client's request. The last stage is De-destroying the Servlet that the Servlet calls `destroy()` method to notify the end of the Servlet instance. The `destroy()` method is called only once during the Servlet lifetime as well as `init()` method. (Tutorials point 2019)

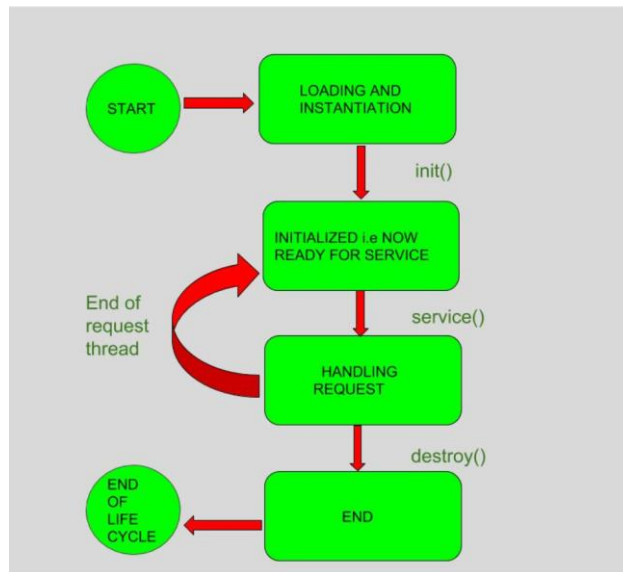


Figure 20. Stages and methods of Servlet Lifecycle (Thakral 2019)

3.3 JSP

Servlet allows the web server to generate dynamic content, but there is one problem in Servlet that users must convert HTML code to be hardwired in Java programming language. Because of that inconvenience, JSP was created, which includes HTML tags and JSP tags, is easier to maintain than Servlet. JSP is a combination of static content and dynamic content that divided into the designing and development parts. Section 3.3 aims to show more useful knowledge and skills in JSP to help thesis readers understand deeply about Java web development technologies. (Javatpoint 2019.)

3.3.1 JSP implementation

JSP is a server-side programming technology that allows generating dynamic content like Servlet technology. Some users think that JSP is as an extension of Servlet that gives more additional features than Servlet, for example: JSTL, Expression Language, Custom Tags (Javatpoint 2019.). JSP is an exact integral part of Java EE that makes a complete platform for enterprise applications. The most complex and demanding web applications become simpler by using JSP. JSP is able to access the whole Java APIs, especially JDBC API to access the enterprise database. (Tutorialspoint 2019.)

This section introduces thesis readers about how to implement JSP in a Java Web Application. Like Servlet implementation, users need to go to File to create a New Dynamic Web Project on the toolbar as shown in Figure 21. After this step, set up a name for the project name and click Finish to complete step 1.

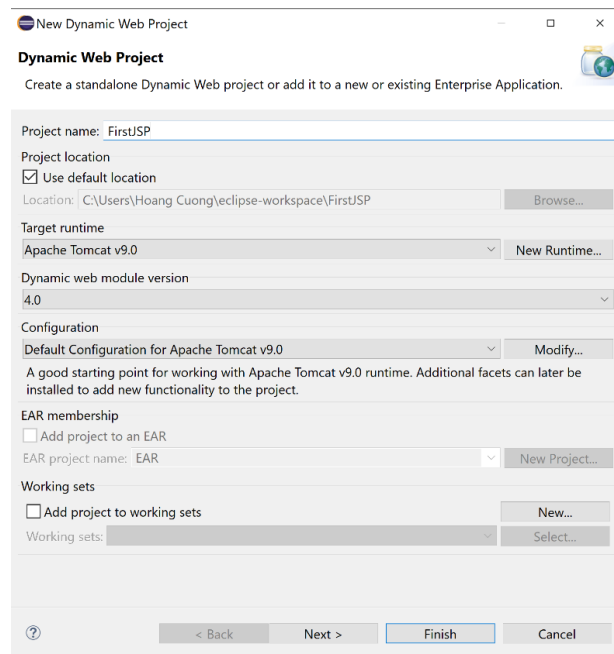


Figure 21. Step 1: Create a New Dynamic Web Project (Screenshot from Eclipse IDE)

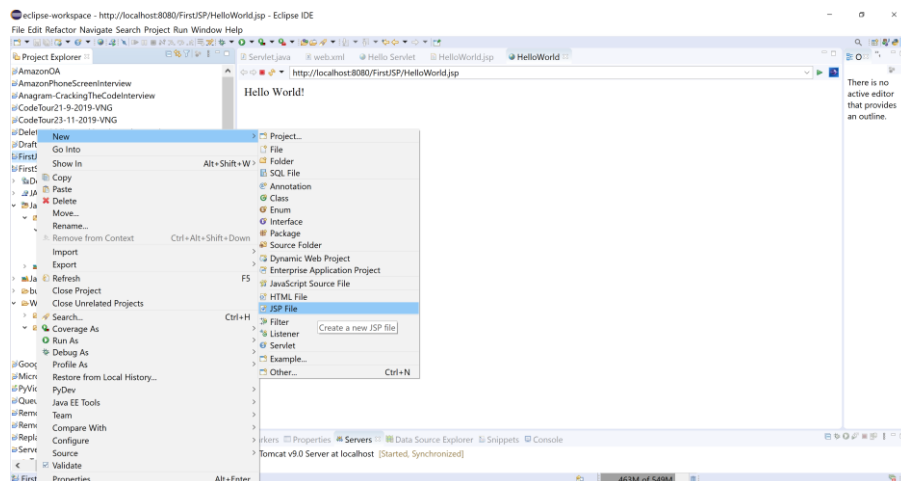


Figure 22. Step 2: Add a new JSP file to Dynamic Web Project (Screenshot from Eclipse IDE)

In the next step, right-click the created Dynamic Web Project in step 1 in order to create a JSP file as shown in Figure 22. In the JSP file, users allow to add and modify JSP code. In creating the JSP file to Dynamic Web Project step, users have to establish a file name for the JSP file as shown in Figure 23.

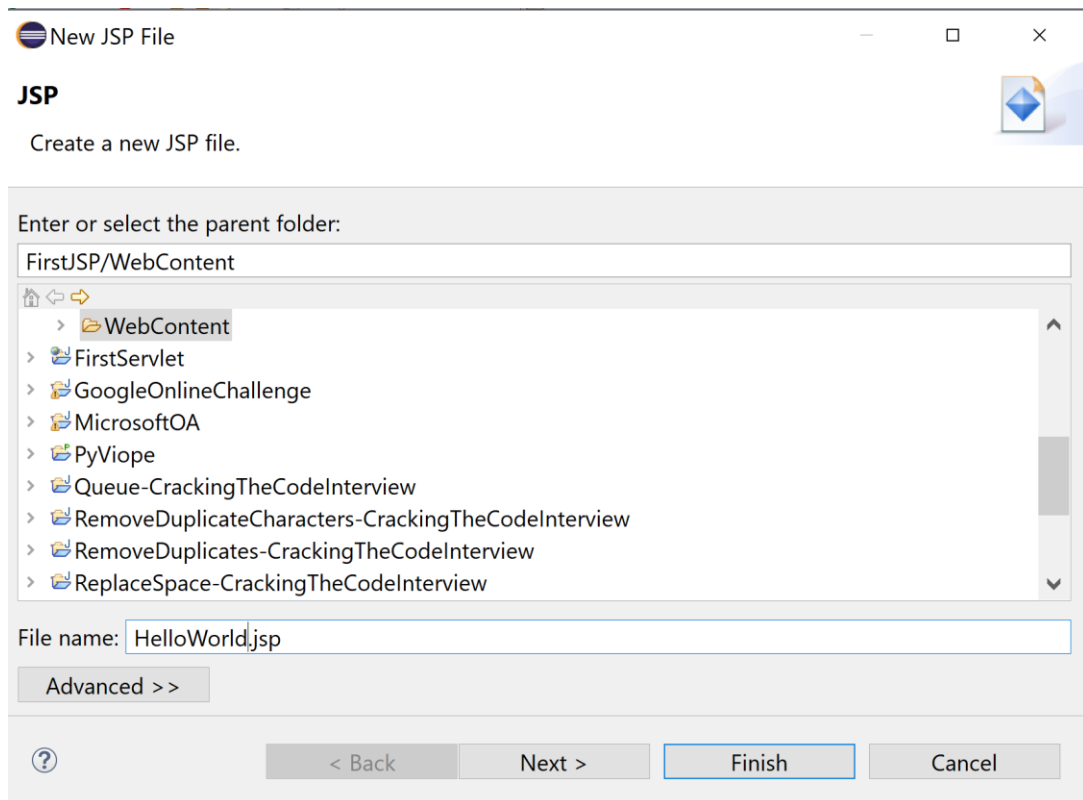


Figure 23. Step 3: Set up the file name for JSP file (Screenshot from Eclipse IDE)

After adding a JSP file to Dynamic Web Project, users can modify the code in JSP file like Figure 24. The layout of the JSP file is similar to HTML so that it is easier to familiarize it with JSP than Servlet. Developers also can put the CSS and JavaScript code here or create on other files and import into JSP file.

```

Servlet.java  web.xml  Hello Servlet  HelloWorld.jsp  HelloWorld
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2     pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="ISO-8859-1">
7  <title>HelloWorld</title>
8  </head>
9  <body>
10     Hello World!
11 </body>
12 </html>

```

Figure 24. Step 4: Modify the code in JSP file (Screenshot from Eclipse IDE)

Last, right-click HelloWorld.jsp file in WebContent folder and choose Run on Server in Run As to run the FirstJSP project. The result is like Figure 25 and that is the front-end layout like other HTML file when they run.

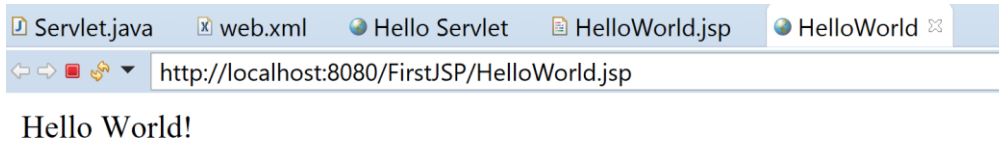


Figure 25. Accessing the JSP (Screenshot from Eclipse IDE)

3.3.2 Comparing JSP with Servlet and other technologies

As mentioned at the beginning of section 3, because of separating the designing and development part, the JSP page is simpler to maintain than Servlet. Moreover, there are many advantages from JSP that makes JSP is stronger and more efficient than Servlet technology as shown in Table 3. First of all, users can use all the functions from the Servlet in JSP, because JSP is an extension of Servlet. Moreover, users are able to use JSTL, Expression Language, Custom Tags in JSP that makes JSP development easier. The second advantage of JSP is mentioned above that is easy maintenance. Thirdly, when the JSP changed, users do not require to compile and deploy the project again. However, Servlet must compile again and update when the code changed. The last advantage is less code than Servlet. (Vaidya 2019.)

| JSP | Servlets |
|----------------------------------|---------------------------------|
| Extension to Servlet | Not an extension to servlet |
| Easy to Maintain | Bit complicated |
| No need to recompile or redeploy | The code needs to be recompiled |
| Less code than a servlet | More code compared to JSP |

Table 3. Advantages of JSP over Servlet (Vaidya 2019.)

Following that, there are many advantages of JSP over other technologies. Initially, JSP is stronger and simpler to use in dynamic than ASP, which is a Microsoft technology and popular in web development. ASP is not an independent platform, but JSP is. Next, compared to JavaScript, JSP is similar to JavaScript in being able to create dynamic HTML content on the client. However, JSP follows server-side programming technology that can access databases, catalogs, pricing information. Pure Servlet is more flexible to write HTML than using `println` syntax to create HTML. Web developers can design the front-end web page by JSP and insert the dynamic content by Servlet separately. (Hubberspot 2020)

3.4 Basic web application structure

A Java web application is built from many components. Servlet is the main key of any Java web application, that mentioned in section 3.2, has responsibility for handling and responding to HTTP requests. That means every client request to the Java web application goes through Servlet. Moreover, a Java web application has a filter, which is a Java class, can prevent requests to Servlet to classify by needs, such as data formatting, response compression, authentication, or authorization. Java web application provides diverse types of listeners that notice code in multiple events. However, JSP is the most powerful tool in Java web development technology. As mentioned in section 3.3, JSP generates dynamic content easily. There are several features in Java web development technology than Servlet, filter, listener, and JSP, but these features mentioned above are the most popular and important for developing Java web applications (Williams 2014). However, any Java web application follows the structure: static contents (HTML), client-side files (CSS & JavaScript), dynamic content generated by JSP, Servlets, external library or jar files, Java utility classes, deployment descriptor (`web.xml`). (Wide-skills 2020)

4 ADDING DATABASE INTO JAVA APPLICATION BY JDBC

Data that is stored and maintained in a database, is shared between web components and is persistent between web application requests. Java web application uses JDBC, which is a Java API to access and execute the query with the relational database. All the steps to use JDBC in a Java web application were mentioned in section 4. Section 4 shows how to MySQL database in section 4.1 after introducing about XAMPP platform for hosting MySQL database. JDBC and the process of integrating JDBC to Java application are shown in section 4.3 and section 4.4.

4.1 XAMPP

XAMPP is an abbreviation for Cross-Platform (X), Apache (A), MySQL (M), PHP (P), Perl (P). XAMPP is a simple and lightweight solution for Apache distribution on Windows, Linux and macOS, that offers MySQL (Database Server) & Apache (Web Server) in only one set-up and developers can manipulate them on the XAMPP dashboard. It is extremely basic for developers in order to create the local server (Apache) and database for web applications (MySQL) for testing purposes. Because of being a Cross-Platform, XAMPP works equally well on different kinds of operation systems (Linux, macOS, Windows). (Udemy 2020)

As mentioned above, XAMPP is available on Windows, macOS, Linux operation system with the small file size (around 149MB). To download and install XAMPP, developers need to access to the XAMPP website: <https://www.apachefriends.org/index.html>. In this page, developers can find out the XAMPP version following their operating system. This thesis introduces how to download and install XAMPP for Windows operation system. After choosing the right install version for Windows, the installer version for Window is going to be in the download folder like Figure 25.

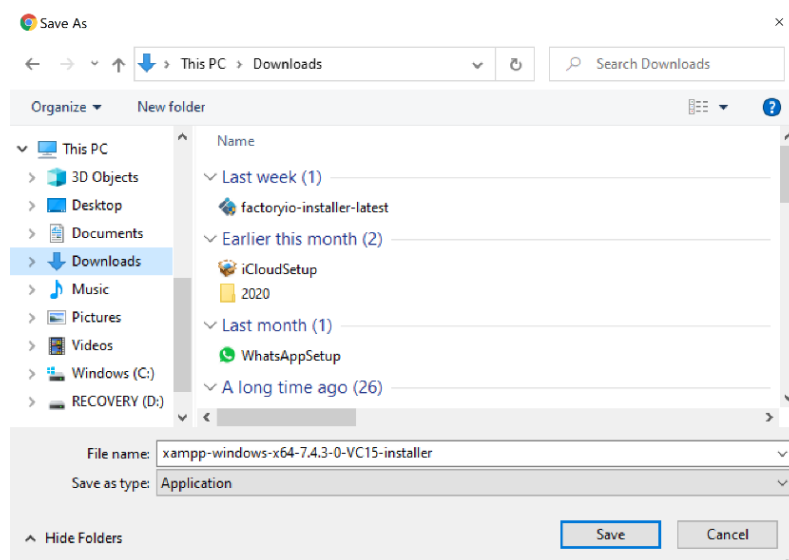


Figure 26. Step 1: Download XAMPP for Windows

The latest XAMPP version for Windows is 7.4.3 that shows in the name of the XAMPP file as shown in Figure 26. Next, click the XAMPP installer file and select install components and choose the folder C:\xampp like Figure 27 and finish the installation. After that, developers can use MySQL database with Apache server.

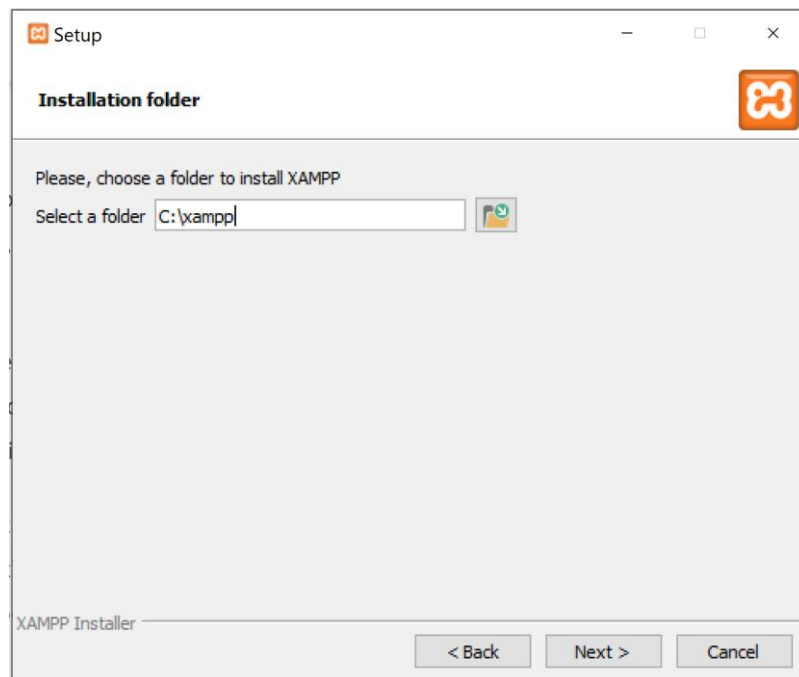


Figure 27. Step 2: XAMPP setup

4.2 Installing and configuring MySQL

MySQL is a popular open-source database management system (RDBMS) based on Structured Query Language (SQL). MySQL is also one of the best RDBMS used for web-based application development. The advantages of using MySQL that help developers structure data, show the information in an organized, update, delete, edit, and retrieve data when developers request in any content management system. MySQL is created for handling big database rapidly that based on a client-server model. (Chahal 2019)

XAMPP is compatible with Apache distribution as mentioned in section 4.1. Following that, MySQL works smoothly on different operating systems which can be integrated with PHP simply. MySQL operates as a database component that must run a database-enabled website and servers on XAMPP. Initially, Apache and MySQL must be run first to start the website on the local server as shown in Figure 28.

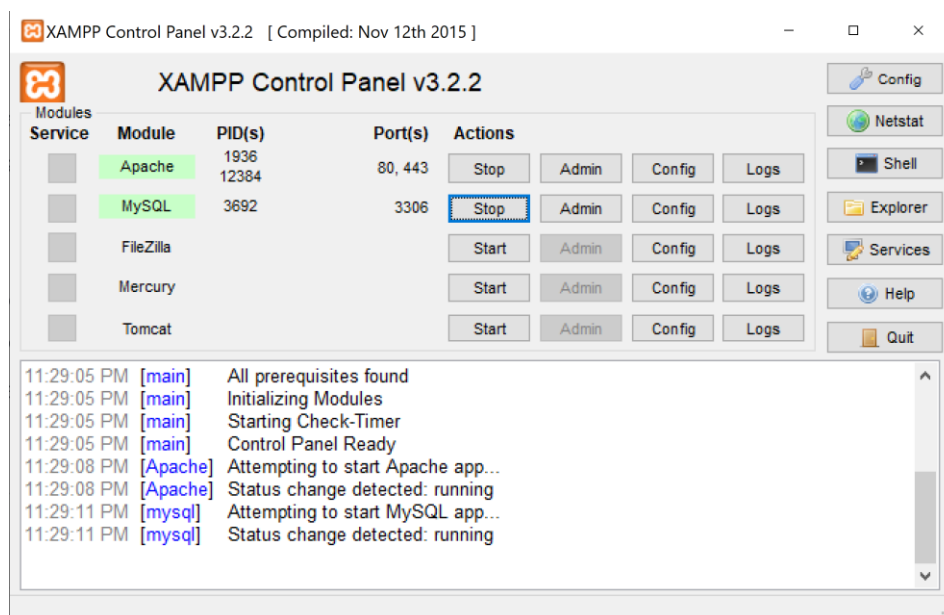


Figure 28. Start MySQL and Apache on the control panel of XAMPP

After starting Apache and MySQL, go to this link <http://localhost/phpmyadmin/index.php?lang=en#> via web browsers such as Google Chrome or Mozilla Firefox to use the control panel of MySQL as shown in Figure 29. On the phpMyAdmin user interface, developers are able to see and structure MySQL database easily. With phpMyAdmin, developers are able to create or modify their databases by SQL syntax or integrated table.

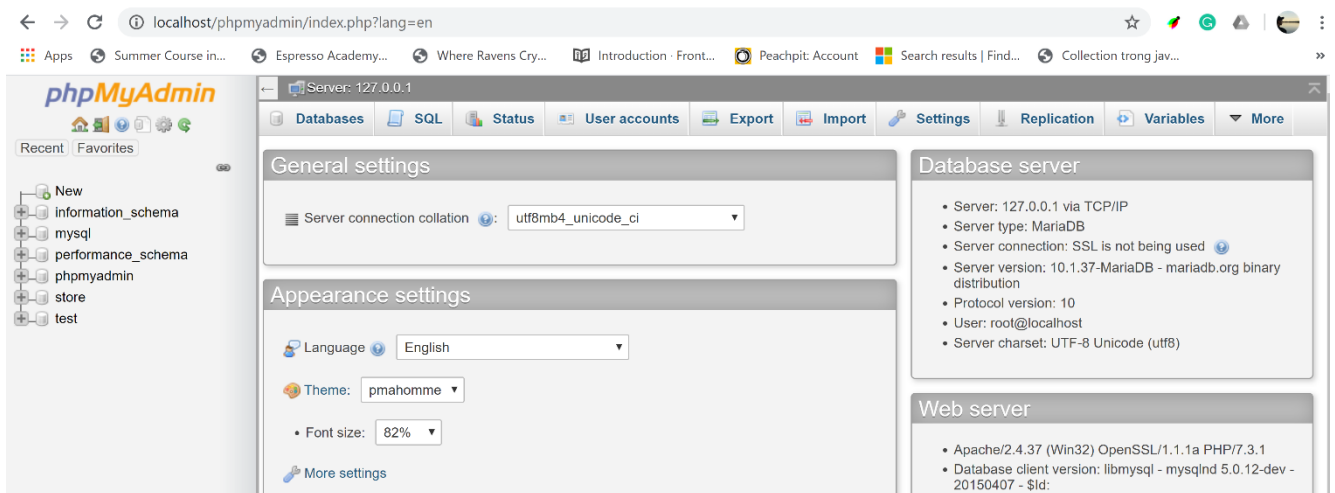


Figure 29. phpMyAdmin user interface

4.3 Introduction and integrating JDBC to Java application

JDBC, which is an abbreviation of Java Database Connectivity, is a Java API to access and execute the query with any kind of tabular data, especially relational database. To access the database, JDBC users JDBC drivers, which fit into one of the four-driver categories, JDBC-ODBC bridge, Native-API driver, Network-Protocol driver, Database-Protocol driver. Because of using JDBC API, developers can connect the Java programming language to a wide range of databases in order to save, update, delete, and retrieve data from the database. The main missions of integrating JDBC for connectivity between Java and databases are making a connection with a database, writing SQL or MySQL statements, executing SQL or MySQL queries in the database, viewing and modifying the data records (Javatpoint 2020). There are 5 steps for connectivity between Java application with MySQL database: loading the driver, creating connection, creating statements, executing the query and closing connection that mentions concretely in section 5.3. (Gupta 2020)

5 THE PROJECT

In order to understand clearly and have a concrete example about Java web application, the author built a sample Java web application with Servlet, JSP, and JDBC in this section 5. This project was developed on Eclipse which is a popular integrated development environment for Java applications. In this project, readers are able to see a Java web application on MVC architecture, which was built by a combination of Servlet, JSP, and JDBC.

5.1 Idea

The purpose of this project is to develop a sample web page that combines all of Java web technology and was researched above. This website is a basic login form which allows readers to understand how Servlet, JSP, and JDBC work together on Java web application based on MVC architecture. Because of frequent appearance on web development, readers can learn and build their own log in forms after reading this thesis in their own projects.

5.2 Analysis

The author decided to use MVC pattern to create the login form website. In the MVC pattern, there are 3 layers to separate and manage the application structure: model, view, controller. The model layer, which is the data layer, contains the business logic of the layer and delegates the state of the application. The controller layer works as an interface between view layer and model layer, which receives the user requests from the view layer and executes those requests, including the necessary validations. Following that, the user requests are sent to the model for data processing. The data is sent back to the controller layer again and displayed on the view layer after processing. On the MVC pattern, the view layer is the user interface that displays the output from the application. This presentation layer shows the data retrieved from the model layer by the controller layer and displays the data to the user (Choudary 2019). Based on the MVC architecture definition, the author decided to divide this login form website file structure like Figure 30.

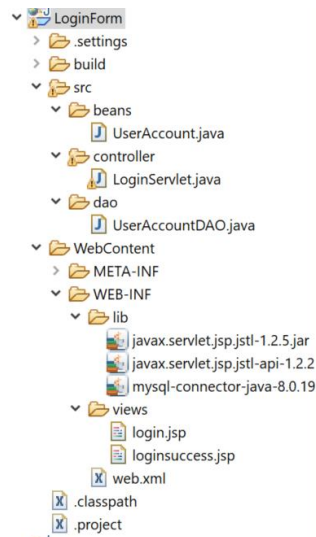


Figure 30. Project structure (Screenshot from Eclipse IDE)

Following Figure 30, beans folder in src represents model layer that stores data to be displayed or treated. Controller file is controller layer that included Servlet file to intercept the HTTP request and return the HTTP response. JSP files, which are login.jsp and loginsuccess.jsp, are the view layers to organise output or display. Based on Figure 31, this login form web begins at login.jsp which users could input their username and password from the web browser in the login form. After filling the username and password, these data are sent to LoginServlet.java in order to process. Inside the LoginServlet.java, username and password which are input from users in the view layer login.jsp, are converted to an object follows the model of UserAccount.java in beans package. Then that object is checked with the data in the database and return to loginsuccess.jsp on the web browser if the input data from the users match with data from the database. UserAccountDAO.java class has the responsibility to requests and return data from the database to LoginServlet.java. In the next section, the author shows how to create login form website step by step by Java in order to give the readers a detailed instruction about creating Java web application.

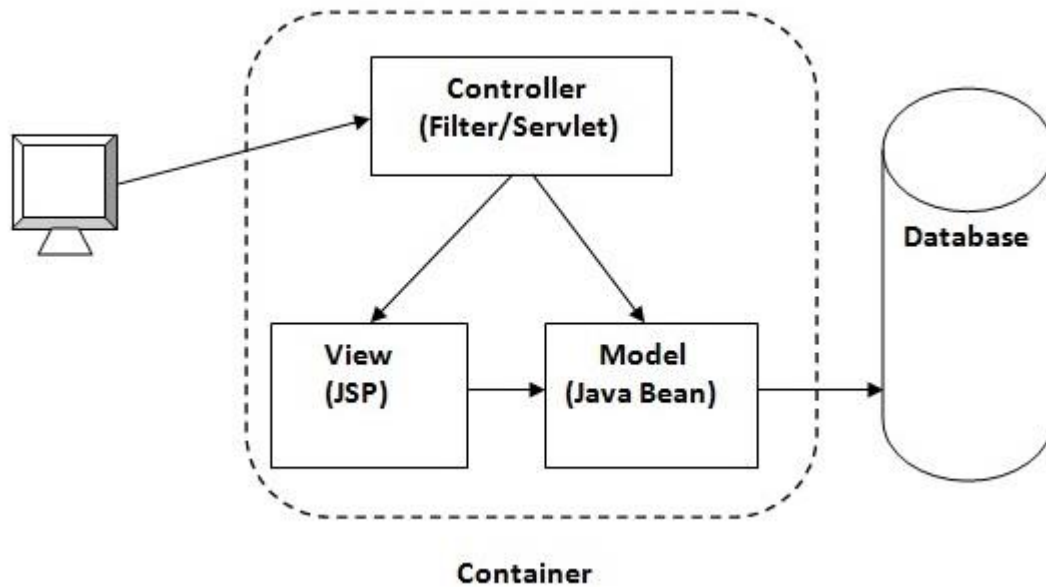


Figure 31. Project workflow (Javatpoint 2020)

5.3 Coding and implementation

Following the analysis in section 5.2 and the project structure in Figure 30, the author created a new dynamic web project on Eclipse named LoginForm and organized the folder and file as well as Figure 30. The first step in creating a new dynamic web project is similar to the introduction in section 3.2.1. After following these steps like the introduction in section 3.2.1, the author created LoginForm by choosing the finish button as shown in Figure 32.

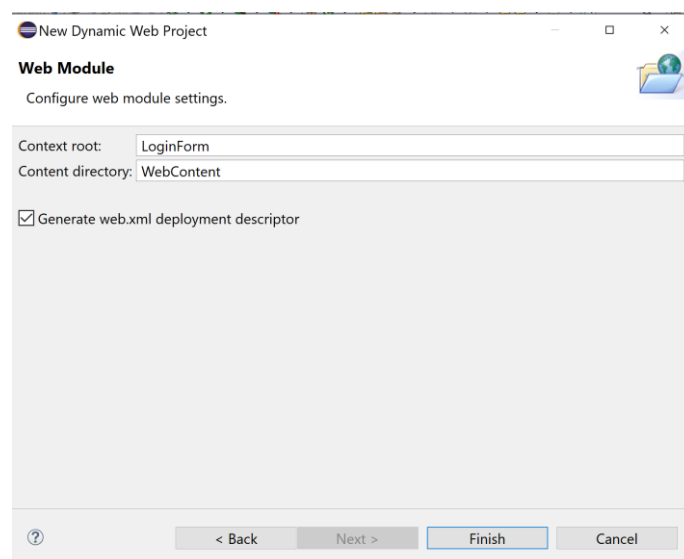


Figure 32. Create LoginForm as a dynamic web project (Screenshot from Eclipse IDE)

Next, the author created login.jsp file which is the first layer view in the web browser and users could input the username and password there. This step was like the introduction of adding a new JSP file to a dynamic web project in section 3.3.1. The author added login.jsp to LoginForm by clicking the finish button as shown in Figure 33.

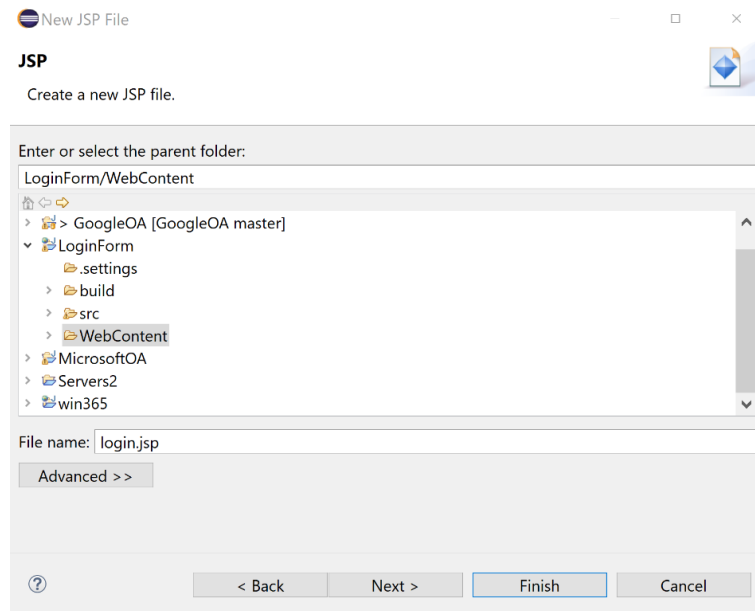


Figure 33. Create login.jsp (Screenshot from Eclipse IDE)

```

login.jsp  loginsuccess.jsp
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2  pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="ISO-8859-1">
7  <title>Log In</title>
8  </head>
9  <body>
10 <div align="center">
11 <h1>Login Form</h1>
12 <form action="<%=request.getContextPath()%>/Login" method="post">
13 <table style="width: 100%">
14 <tr>
15 <td>UserName</td>
16 <td><input type="text" name="username" /></td>
17 </tr>
18 <tr>
19 <td>Password</td>
20 <td><input type="password" name="password" /></td>
21 </tr>
22 </table>
23 <input type="submit" value="Submit" />
24 </form>
25 </div>
26 </body>
27 </html>
28

```

Figure 34. Login.jsp (Screenshot from Eclipse IDE)

In the login.jsp as shown in Figure 34, it includes the form for logging in where users are able to fill in username and password. That log in form is written in HTML and CSS which is similar to many other

front-end web developments. Following that, readers can understand and modify the form easily if they want to do that. In the HTML part of the login form, it used normally form and input tags to retrieve data (username and password) from users. Inside the form tag, it has an action attribute to call the Servlet file in order to execute request log in when users click submit. After clicking the submit input tag, the username parameter and password parameter are sent to Servlet, and Servlet is going to handle that request which is checking the username and password with the data on the database. The next step was creating the Servlet file that carries out the request from users in the login.jsp. The Servlet file named LoginServlet.java was created by clicking new and other, and choosing create a new Servlet like Figure 35. This step is the same as creating JSP file step.

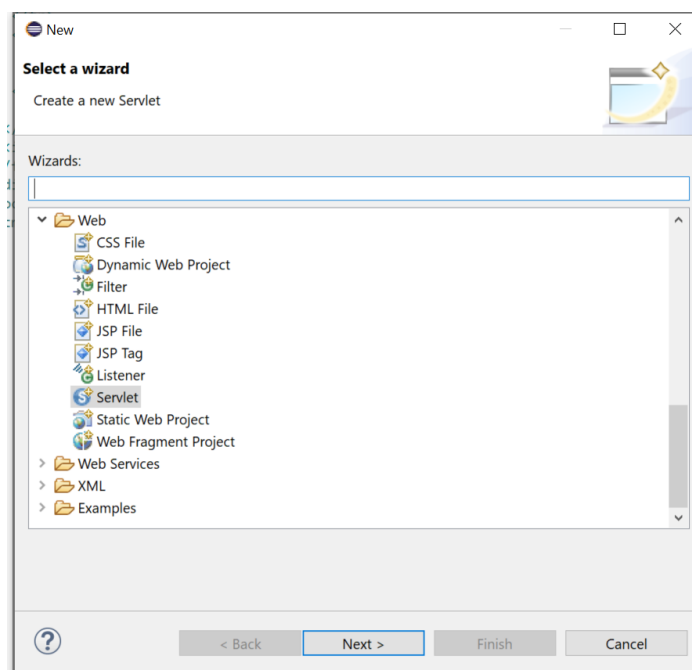
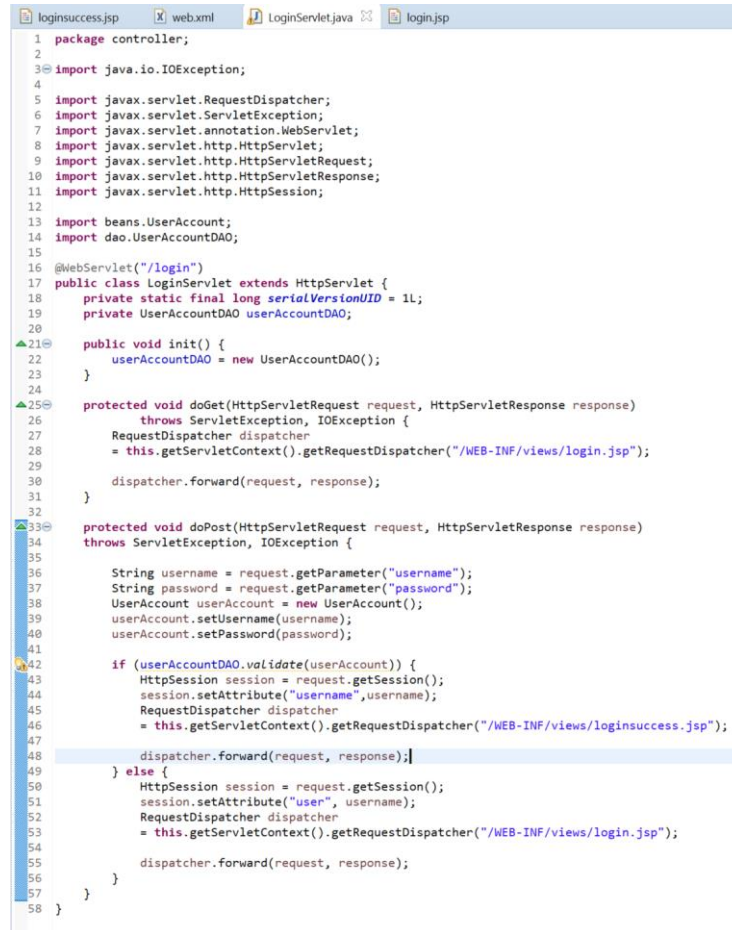


Figure 35. Create LoginServlet.java (Screenshot from Eclipse IDE)

After creating the LoginServlet.java, the author has modified that file like Figure 36. Line 16 is the link to map Servlet file that could be called from the JSP file. From line 25 to line 31, this doGet function is used for calling login.jsp page. Because of the login form using method post as shown in Figure 34, author used doPost function inline 33 to handle the request from the Login.jsp. Line 36 and 37, the data of username and password are caught by request.getParameter following attribute name for input tag in Login.jsp. The author declared an object as UserAccount in order to store data which received from handling requests from Login.jsp in between line 38 and line 40. From line 42 to line 56, there is if-else condition to check the UserAccount with the database. In the first condition, if UserAccount validated the data from the database, the author would declare the session and used setAttribute for username to

send data inline 43 and line 44. Line 45, it would redirect the page to loginsuccess.jsp and the data set with setAttribute also send to loginsuccess.jsp. In the second condition, if UserAccount did not validate to data in the database, it would redirect the page to login.jsp as shown line 52.



```

1 package controller;
2
3 import java.io.IOException;
4
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import javax.servlet.http.HttpSession;
12
13 import beans.UserAccount;
14 import dao.UserAccountDAO;
15
16 @WebServlet("/login")
17 public class LoginServlet extends HttpServlet {
18     private static final long serialVersionUID = 1L;
19     private UserAccountDAO userAccountDAO;
20
21     public void init() {
22         userAccountDAO = new UserAccountDAO();
23     }
24
25     protected void doGet(HttpServletRequest request, HttpServletResponse response)
26         throws ServletException, IOException {
27         RequestDispatcher dispatcher
28             = this.getServletContext().getRequestDispatcher("/WEB-INF/views/login.jsp");
29
30         dispatcher.forward(request, response);
31     }
32
33     protected void doPost(HttpServletRequest request, HttpServletResponse response)
34         throws ServletException, IOException {
35
36         String username = request.getParameter("username");
37         String password = request.getParameter("password");
38         UserAccount userAccount = new UserAccount();
39         userAccount.setUsername(username);
40         userAccount.setPassword(password);
41
42         if (userAccountDAO.validate(userAccount)) {
43             HttpSession session = request.getSession();
44             session.setAttribute("username", username);
45             RequestDispatcher dispatcher
46                 = this.getServletContext().getRequestDispatcher("/WEB-INF/views/loginsuccess.jsp");
47             dispatcher.forward(request, response);
48         } else {
49             HttpSession session = request.getSession();
50             session.setAttribute("user", username);
51             RequestDispatcher dispatcher
52                 = this.getServletContext().getRequestDispatcher("/WEB-INF/views/login.jsp");
53             dispatcher.forward(request, response);
54         }
55     }
56 }
57
58 }

```

Figure 36. LoginServlet.java code (Screenshot from Eclipse IDE)

The author created an object class in folder beans named UserAccount.java. This java class is an object-oriented design for UserAccount with 2 parameters: name and password. In Figure 37, this is the encapsulation in object-oriented programming that would hide the data with users.

```

1 package beans;
2
3 import java.io.Serializable;
4
5 public class UserAccount implements Serializable {
6
7     private static final long serialVersionUID = 1L;
8     private String username;
9     private String password;
10
11     public String getUsername() {
12         return username;
13     }
14
15     public void setUsername(String username) {
16         this.username = username;
17     }
18
19     public String getPassword() {
20         return password;
21     }
22
23     public void setPassword(String password) {
24         this.password = password;
25     }
26 }
27

```

Figure 37. UserAccount.java code (Screenshot from Eclipse IDE)

After these steps above, the author continued to start with the connection and retrieve data from the database in UserAccountDAO.java which is inside the dao folder. The code in UserAccountDAO.java follows Figure 38. From line 13 to line 16, there are declarations that are status for checking the validate function and return a Boolean value, Connection, PreparedStatement, and ResultSet for retrieving data. Between line 18 and line 22, these string variables are the information for connecting the MySQL database which is called inline 26. Next line, this is the SQL syntax to query data from the database. Inline 34, this code would execute the SQL syntax inline 29 and the Boolean value was going to be returned inline 35. If the username and password match with the data in the database, the status will be true and vice-versa.

```

1 package dao;
2
3 import java.sql.Connection;
4
5
6
7
8
9
10
11 public class UserAccountDAO {
12     public static boolean validate(UserAccount userAccount) {
13         boolean status = false;
14         Connection conn = null;
15         PreparedStatement pst = null;
16         ResultSet rs = null;
17
18         String url = "jdbc:mysql://localhost:3306/";
19         String dbName = "user";
20         String driver = "com.mysql.jdbc.Driver";
21         String userName = "root";
22         String password = "";
23
24         try {
25             Class.forName(driver).newInstance();
26             conn = DriverManager
27                 .getConnection(url + dbName, userName, password);
28
29             pst = conn
30                 .prepareStatement("select * from login where username=? and password=?");
31             pst.setString(1, userAccount.getUsername());
32             pst.setString(2, userAccount.getPassword());
33
34             rs = pst.executeQuery();
35             status = rs.next();
36
37         } catch (Exception e) {
38             System.out.println(e);
39         } finally {
40             if (conn != null) {
41                 try {
42                     conn.close();
43                 } catch (SQLException e) {
44                     e.printStackTrace();
45                 }
46             }
47             if (pst != null) {
48                 try {
49                     pst.close();
50                 } catch (SQLException e) {
51                     e.printStackTrace();
52                 }
53             }
54             if (rs != null) {
55                 try {
56                     rs.close();
57                 } catch (SQLException e) {
58                     e.printStackTrace();
59                 }
60             }
61         }
62         return status;
63     }
64 }
65

```

Figure 38. UserAccountDAO.java code (Screenshot from Eclipse IDE)

In order to connect to the database, it requires a MySQL database in the localhost. The author was going to create a new MySQL in the next step. To start working with MySQL, readers can read section 4.2: installing and configuring MySQL and follow these steps. Creating a new database by choosing new button in the left top corner as shown in Figure 39 and modify database name is user and language box is utf8_unicode_ci. Next, click user database in order to create a table named login with 2 columns and modify these columns as Figure 40.

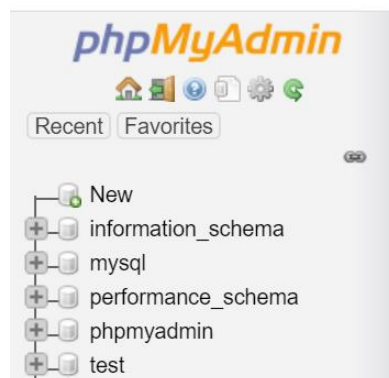


Figure 39. Create new MySQL database in phpMyAdmin

Table name: Add column(s)

| Name | Type | Length/Values | Default | Collation | Attributes | Null | Index |
|---|---------|----------------------------------|---------|-----------|------------|--------------------------|-------|
| <input type="text" value="username"/> <small>Pick from Central Columns</small> | VARCHAR | <input type="text" value="255"/> | None | | | <input type="checkbox"/> | --- |
| <input type="text" value="password"/> <small>Pick from Central Columns</small> | VARCHAR | <input type="text" value="255"/> | None | | | <input type="checkbox"/> | --- |

Table comments:
Collation:
Storage Engine:

PARTITION definition:
Partition by: ()
Partitions:

Figure 40. Create new table in user database with 2 columns named username and password

After completing the login table in user database, the author inserted 1 row to the table as well as Figure 41 by choosing insert button on the toolbar. The table result would be the same as Figure 42 after doing all of these steps above. Author set up 1 account which has username and password like Figure 41.

| Column | Type | Function | Null | Value |
|----------|--------------|----------------------|------|-------------------------------------|
| username | varchar(255) | <input type="text"/> | | <input type="text" value="admin"/> |
| password | varchar(255) | <input type="text"/> | | <input type="text" value="abc123"/> |

Figure 41. Insert data into login table

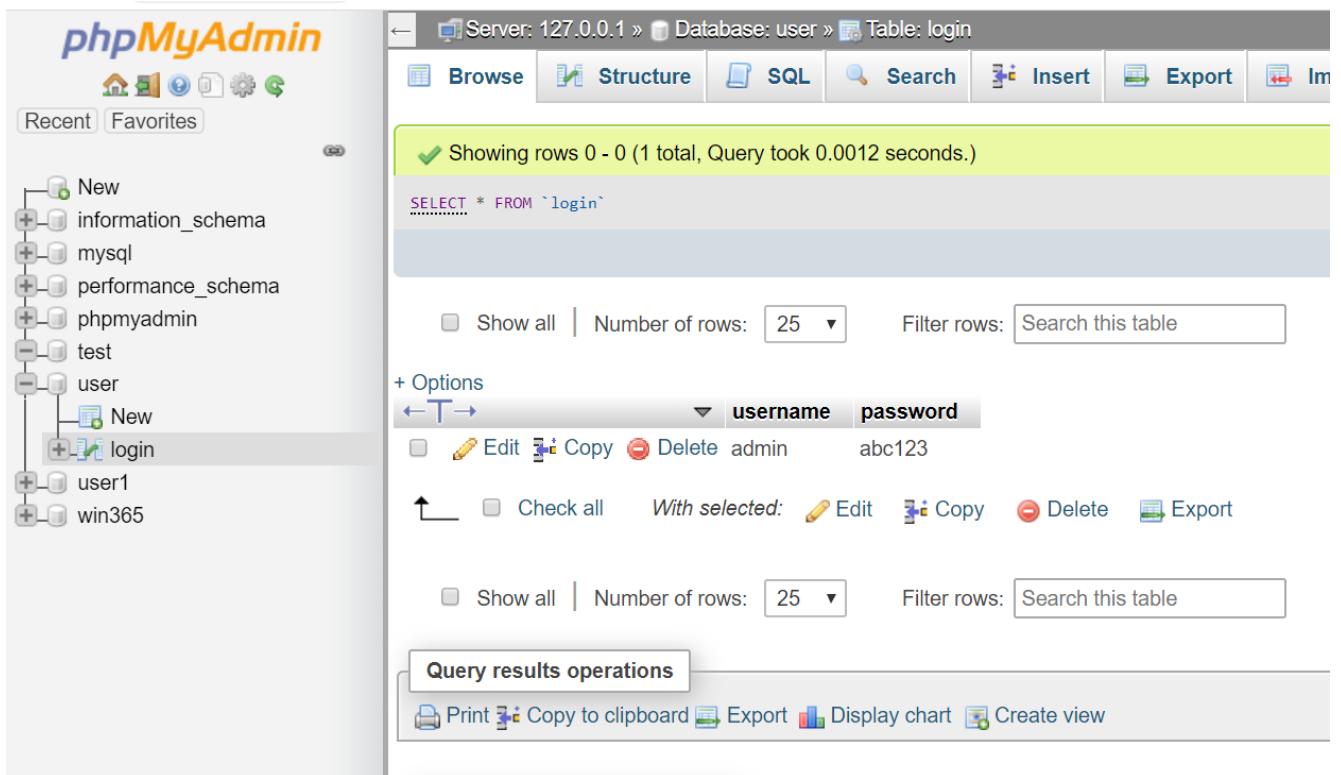


Figure 42. Database result

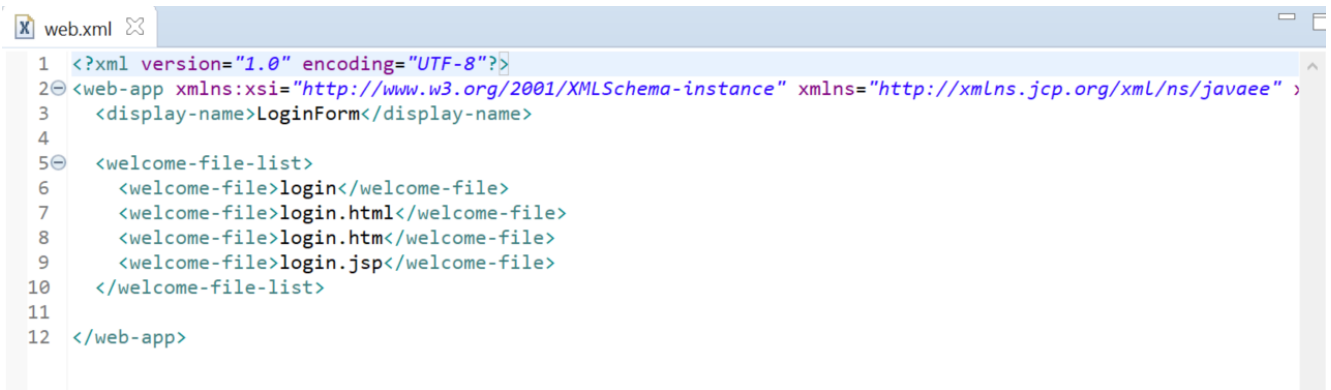
After that, the author created one more JSP file named `loginsuccess.jsp` for redirect page when users log in successfully. The code inside `loginsuccess.jsp` file looks like Figure 43. Inline 12, `session.getAttribute("username")` would catch the data which is set at `username` parameter and the text inside `<h1></h1>` would be `Hello, admin`. The value of attribute that named `username` would be sent by Servlet.

```

loginsuccess.jsp  web.xml  LoginServlet.java  login.jsp
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2  pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="ISO-8859-1">
7  <title>Main Page</title>
8  </head>
9  <body>
10 <div align="center">
11 <h1>Hello,
12     <%=session.getAttribute("username")%></h1>
13 </div>
14 </body>
15 </html>

```

Figure 43. `loginsuccess.jsp` code (Screenshot from Eclipse IDE)



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" >
3   <display-name>LoginForm</display-name>
4
5   <welcome-file-list>
6     <welcome-file>login</welcome-file>
7     <welcome-file>login.html</welcome-file>
8     <welcome-file>login.htm</welcome-file>
9     <welcome-file>login.jsp</welcome-file>
10  </welcome-file-list>
11
12 </web-app>

```

Figure 44. Modify web.xml (Screenshot from Eclipse IDE)

The last steps are modifying web.xml and adding the library to the lib folder in WEB-INF. The author modified the web.xml file like Figure 44 in order to call login.jsp as index page. In the lib folder, the author inserted 3 libraries into this folder as shown in Figure 45 which are MySQL connector library, Servlet JSP library, and Servlet JSP API library. Readers can find out that easily on this page <https://mvnrepository.com/> with these name libraries in Figure 45. Download jar files and copy these files to the lib folder.



Figure 45. Libraries (Screenshot from Eclipse IDE)

To run the project, right-click LoginForm folder, choose Run As, and choose Run On Server. In Run On Server pop up, choose Manually define a new server and modify that server like Figure 46 and click Finish to start running the project. After this step, Servlet and JSP are able to run on Tomcat v9.0.

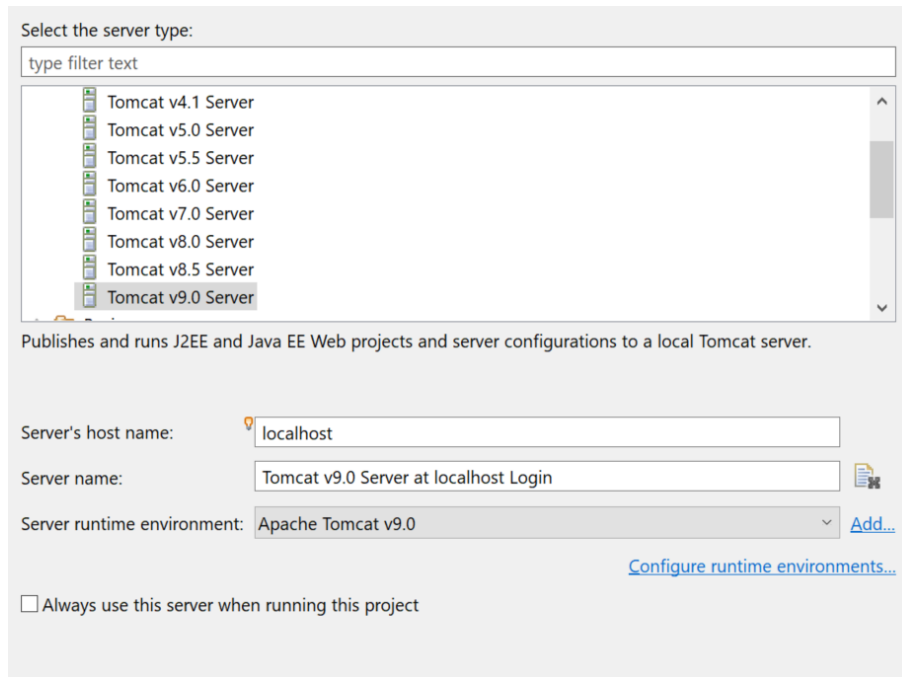


Figure 46. Modify and run Server to start the project (Screenshot from Eclipse IDE)

The result of this project would follow these below Figures (Figure 47, Figure 48, Figure 49). The user would insert username (admin) and password (abc123) into username and password box like Figure 48. After a successful login, this first page - login.jsp - would be redirected to loginsuccess.jsp and say hello to the user by username as shown in Figure 49.

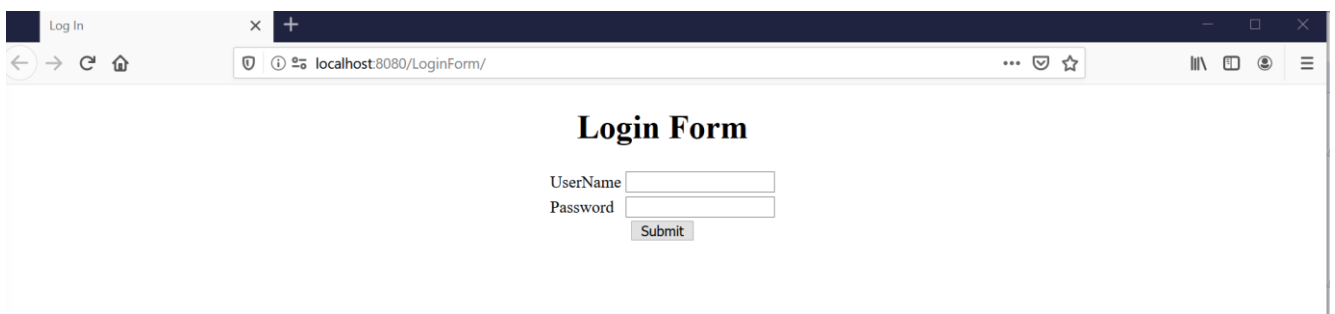
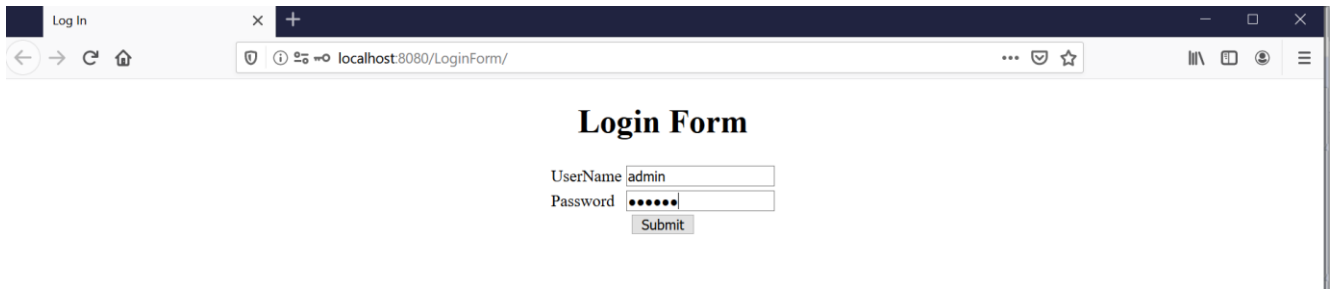


Figure 47. The first page when running the project



The screenshot shows a web browser window with a single tab titled "Log In". The address bar displays "localhost:8080/LoginForm/". The main content area features a centered heading "Login Form". Below the heading is a form with two input fields: "UserName" containing the text "admin" and "Password" containing masked characters (dots). A "Submit" button is positioned below the password field.

Figure 48. Users insert data into the log in form

When the users submit data from the login form like Figure 48, the Servlet would receive and retrieve data from users. The data would be executed to check with the data on the MySQL database and if it was validated so that Servlet would send loginsuccess.jsp to web browser like Figure 49.



Figure 49. loginsuccess.jsp called after successful log in

6 CONCLUSION

The main goal of the project was to build the sample Java web application in order to have a concrete instance for understanding and learning how to use and combine Servlet, JSP, and JDBC in a Java web application. The important parts were creating JSP files, Servlet files, MySQL database, and connecting the database to Servlet in login validation function. Many techniques such as XAMPP, MySQL, JSP, Servlet, JDBC are exposed in this thesis.

This sample project can be expanded easily in such as registration or also a book-store application with these explained Java technologies above. The book-store application is comprised of CRUD, which are read, update, and delete functions in order to build some features such as add books, delete books, or also modify the user information. However, it requires more knowledge in web development, for instance: AJAX, SQL, JavaScript, or jQuery to build a complete website. This thesis project can be a material for readers who want to learn and understand the fundamental Java web application.

REFERENCES

- Chahal, P. 2019. XAMPP MySQL: How to Install, Configure and Use. Available at: <https://blog.templateoaster.com/xampp-mysql/>. Accessed 24 March 2020.
- Cheah, D. 2019. How to work with Servlet, JSP and JDBC? Available at: <https://medium.com/@tattwei46/how-to-work-with-servlet-jsp-and-jdbc-fcc568a6a57b>. Accessed: 27 December 2019.
- Choudary, A. 2019. How to Implement MVC Architecture in Java? Available at: <https://www.edureka.co/blog/mvc-architecture-in-java/>. Accessed 31 March 2020.
- Edureka. 2019. What is a Java Web Application? Available at: <https://www.edureka.co/blog/java-web-application/>. Accessed: 21 December 2019.
- Gupta, K. 2018. What big companies still code in Java: Do major corporations still use Java? Available at: <https://www.freelancinggig.com/blog/2018/08/29/what-big-companies-still-code-in-java-do-major-corporations-still-use-java/>. Accessed 8 June 2020.
- Gupta, S. 2020. Establishing JDBC Connection in Java. Available at: <https://www.geeksforgeeks.org/establishing-jdbc-connection-in-java/>. Accessed 28 March 2020.
- Hubberspot. 2020. Java Server Pages (JSP): Advantages over Servlet technologies. Available at: <https://www.hubberspot.com/2012/04/java-server-pages-jsp-advantages-over.html>. Accessed: 1 January 2020.
- Jain, N. 2019. A Complete Guide to JVM Languages. Available at: <https://www.whizlabs.com/blog/jvm-languages/>. Accessed: 21 December 2019.
- Javatpoint. 2019. JSP Tutorial. Available at: <https://www.javatpoint.com/jsp-tutorial>. Accessed: 29 December 2019.
- Javatpoint. 2020. Java JDBC Tutorial. Available at: <https://www.javatpoint.com/java-jdbc>. Accessed 26 March 2020.

Javatpoint. 2020. MVC in JSP. Available at: <https://www.javatpoint.com/MVC-in-jsp>. Accessed 31 March 2020.

JReport. 2019. 3-Tier Architecture: A complete overview. Available at: <https://www.jinfony.com/resources/bi-defined/3-tier-architecture-complete-overview/>. Accessed: 24 December 2019.

Layka, V. 2014. Learn Java for Web Development. America: Apress Media.

Meador, D. 2018. Distributed System. Available at: <https://www.tutorialspoint.com/Distributed-Systems>. Accessed: 25 December 2019.

o7planning. 2019. Java Servlet Tutorial for Beginners. Available at: <https://o7planning.org/en/10169/java-servlet-tutorial>. Accessed: 26 December 2019.

Oracle. 2010. The web container. Available at: <https://docs.oracle.com/cd/E19226-01/820-7759/gcrmb/index.html>. Accessed: 23 December 2019.

Rouse, M. 2019. Java virtual machine (JVM). Available at: <https://www.theserverside.com/definition/Java-virtual-machine-JVM>. Accessed: 21 December 2019.

Singh, C 2019. How to configure Apache Tomcat Server in Eclipse IDE. Available at: <https://beginnersbook.com/2017/06/how-to-configure-apache-tomcat-server-in-eclipse-ide/>. Accessed: 25 December 2019.

Stackify. 2017. What is N-Tier Architecture? How it works, examples, tutorials, and more. Available at: <https://stackify.com/n-tier-architecture/>. Accessed: 24 December 2019.

Thakral, K. 2019. Life Cycle of a Servlet. Available at: <https://www.geeksforgeeks.org/life-cycle-of-a-servlet/>. Accessed: 28 December 2019.

Techopedia. 2019. Definition – What does Java mean? Available at: <https://www.techopedia.com/definition/3927/java>. Accessed: 22 December 2019.

Tutorialspoint. 2019. Servlets - Life Cycle. Available at: <https://www.tutorialspoint.com/servlets/servlets-life-cycle.htm>. Accessed: 29 December 2019.

Tutorialspoint. 2019. JSP Tutorial. Available at: <https://www.tutorialspoint.com/jsp/index.htm>. Accessed: 30 December 2019.

Udemy. 2020. Development XAMPP Tutorial: How to Use XAMPP to Run Your Own Web Server. Available at: <https://www.udemy.com/blog/xampp-tutorial/>. Accessed 24 March 2020.

Vaidya, N. 2019. Servlet and JSP Tutorial – How to Build Web Applications in Java? Available at: <https://www.edureka.co/blog/servlet-and-jsp-tutorial/#Introduction>. Accessed: 28 December 2019.

Vogel, L. 2019. Apache Tomcat – Tutorial. Available at: <https://www.vogella.com/tutorials/ApacheTomcat/article.html>. Accessed: 25 December 2019.

Wales, M. 2014. 3 Web Dev Careers Decoded: Front-end vs Back-end vs Full Stack. Available at: <https://blog.udacity.com/2014/12/front-end-vs-back-end-vs-full-stack-web-developers.html>. Accessed 2 June 2020.

Wideskills. 2020. Web Application Structure. Available at: <https://www.wideskills.com/servlets/web-app-structure>. Accessed 22 March 2020.

Williams, N. 2014. Professional Java for Wev Application. India: John Wiley & Sons, Inc.

Zeepedia. 2019. Java: Layers and Tiers. Available at: https://www.zeepedia.com/read.php?java_layers_and_tiers_web_design_and_development&b=20&c=41. Accessed: 24 December 2019.

