DIPLOMA OF
ASSOCIATE ENGINEER
3RD YEAR

A TEXT BOOK OF WEB DEVELOPMENT WITH JAVA CIT-303

COMPUTER
INFORMATION TECHNOLOGY
APPROVED BY TEVTA PUNJAB

Developed By :
Academics Wing
Technical Educations & Vocational
Training Authority Punjab

Web Development with JAVA

CIT-303

FOR DAE 3rd Year

TECHNICAL EDUCATION &
VOCATIONAL TRAINING AUTHORITY
PUNJAB

PREFACE

The text book has been written to cover the syllabus of Web Development with JAVA 3rd year D.A.E (CIT) according to the new scheme of studies. The book has been written in order to cater the needs of latest concepts and needs of the course i.e.Web Development with JAVA and to be able to attempt D.A.E Examination of PBTE Lahore.

The aim of bringing out this book is to enable the students to have sound knowledge of the subject. Every aspect has been discussed to present the subject matter in the most concise, compact lucid & simple manner to help the subject without any difficulty. Frequent use of illustrative figures has been made for clarity. Short Questions and Self-tests have also been included at the end of each chapter which will serve as a quick learning tool for students.

The author would like to thank the reviewers whose valuable recommendations have made the book more readable and understandable. Constructive criticisms and suggestions for the improvements in future are welcome.

AUTHORS

MANUAL DEVELOPMENT COMMITTEE

MUHAMMAD AZAM (Convener)

(GCT Kamalia)

SAMYA KANWAL (Member)

(GCT (w) Bahawalpur)

MAHVISH MAJEED (Member)

(GCT (w) Layyah)

Table of Contents

CHAPTER NO. 01 ---1

INTRODUCTION TO WEB DEVELOPMENT --1

1.1 WHAT IS A WEB APPLICATION? --1
1.1.1 HOW A WEB APPLICATION WORKS--2
1.1.2 BENEFITS OF A WEB APPLICATION--2
1.2 HTTP BASICS--3
1.2.1 HOW HTTP WORKS?---4
1.3 TYPES OF HTTP REQUESTS--5
1.3.1 HTTP REQUESTS AND RESPONSES --5
1.3.2 REQUEST METHODS: ---5
1.3.3 GET VS POST METHODS---6
1.4 SERVER SIDE PROGRAMMING---7
1.4.1 SERVER-SIDE USES --7
1.4.2 SERVER SIDE PROGRAMMING LANGUAGES ---7
1.5 CLIENT SIDE PROGRAMMING --8
1.5.1 CLIENT-SIDE USES---8
1.5.2 CLIENT SIDE PROGRAMMING LANGUAGE EXAMPLES-----------------------------------9
1.6 WEB APPLICATION LAYERS---10
1.6.1 THE THREE LAYER MODEL:- --10
1.6.2 BENEFITS OF THREE-TIER ARCHITECTURE--11
PART I: SHORT QUESTIONS ---12
PART II: LONG QUESTIONS---12
MULTIPLE CHOICE QUESTIONS ---13
REFERENCES--14

CHAPTER NO. 02 --15

SETTING UP DEVELOPMENT ENVIRONMENT--15

2.1 INSTALLATION OF APACHE TOMCAT SERVER---15
2.1.1 WHAT IS TOMCAT? ---15
2.1.2 INSTALLATION PROCESS OF APACHE TOMCAT SERVER ------------------------------15
2.1.3 ADVANTAGES OF TOMCAT--20
2.1.4 DISADVANTAGES OF TOMCAT ---20

2.2 INSTALATION OF IDE (ECLIPSE/ NETBEANS) ---------- 20
2.1.1 INSTALLATION PROCESS OF NETBEANS: ---------- 21
2.2 STANDARD DIRECTORY STRUCTURE OF JAVA WEB APPLICATION ---------- 24
2.3 DEVELOPMENT OF WEB APPLICATION ---------- 26
2.3.1 WEB APPLICATION DEVELOPMENT STAGES ---------- 26
PART I: SHORT QUESTIONS ---------- 29
PART II: LONG QUESTIONS ---------- 29
MULTIPLE CHOICE QUESTIONS ---------- 30
REFERENCES ---------- 31

CHAPTER NO. 03 ---------- 32

HTML AND CSS ---------- 32

3.1 HTML DOCUMENT STRUCTURE AND TAGS ---------- 32
3.1.1 HTML DOCUMENT STRUCTURE: ---------- 32
3.1.2 HTML TAGS ---------- 34
3.2 DESCRIBE CSS STYLES ---------- 40
3.2.1 INLINE CSS ---------- 41
3.2.2 INTERNAL CSS ---------- 41
3.2.3 EXTERNAL CSS ---------- 42
3.3 DESCRIBE TABLES ---------- 44
3.4 DESIGNING FORMS ---------- 46
3.5 ADVANCE PAGE LAYOUT ---------- 48
PART I: SHORT QUESTIONS ---------- 52
PART II: LONG QUESTIONS ---------- 52
MULTIPLE CHOICE QUESTIONS ---------- 53
REFERENCE: ---------- 54

CHAPTER. NO. 04 ---------- 55

JAVASCRIPT ---------- 55

4.1 WHAT IS JAVSCRIPT? ---------- 55
4.1.1 JAVASCRIPT CAN CHANGE HTML CONTENT ---------- 56
4.1.3 JAVASCRIPT CAN SHOW HTML ELEMENTS ---------- 56
4.1.4 JAVASCRIPT CAN CHANGE HTML STYLES (CSS) ---------- 56
4.3 STYLING ELEMENTS ---------- 57

4.4 USING JQUERY ---------- 59
4.4.1 jQUERY SYNTAX ---------- 59
4.4.2 jQUERY HIDE AND SHOW ---------- 59
4.5 jQUERY SELECTORS ---------- 60
4.5.1 THE ELEMENT SELECTOR ---------- 60
4.5.2 THE #ID SELECTOR ---------- 60
4.5.3 THE .CLASS SELECTOR ---------- 61
4.6 APPLY VALIDATION ON HTML FORM ---------- 61
PART I: SHORT QUESTIONS ---------- 63
PART II: LONG QUESTIONS ---------- 63
MULTIPLE CHOICE QUESTIONS ---------- 64
REFRENCES ---------- 66

CHEPTER NO. 5 ---------- 67

JAVA SERVLETS ---------- 67

5.1 WHAT IS A SERVLET? ---------- 67
5.2 ADVANTAGES OF SERVLET ---------- 68
5.3 SERVLET TYPES ---------- 68
5.3.1 GENERIC SERVLETS ---------- 68
5.3.2 HTTP SERVLETS: ---------- 69
5.4 WRITING BASIC HELLO WORLD SERVLET ---------- 70
5.5 SERVLET LIFE CYCLE ---------- 71
5.6 SERVLET AND FORMS ---------- 73
5.6 VALIDATING FORMS ---------- 77
PART I: SHORT QUESTIONS ---------- 78
PART II: LONG QUESTIONS ---------- 78
MULTIPLE CHOICE QUESTIONS ---------- 79
REFERENCES ---------- 80

CHAPTER NO. 6 ---------- 81

JAVA SERVER PAGES ---------- 81

6.1 INTRODUCTION TO JSP: ---------- 81
6.2 LIFE CYCLE OF JSP ---------- 82
6.3 JSP ELEMENTS ---------- 83

6.3.1 JSP SCRIPTING ELEMENTS ---- 83
6.3.2 JSP DIRECTIVE ELEMENTS ---- 84
6.3.3 JSP STANDARD ACTION ELEMENTS ---- 85
6.4 JSP STANDARD ACTIONS ---- 85
PART I: SHORT QUESTIONS ---- 87
PART II: LONG QUESTIONS ---- 87
MULTIPLE CHOICE QUESTIONS ---- 88
REFERENCES ---- 89

CHAPTER NO. 07 ---- **90**

SESSION MANAGEMENT ---- **90**

7.1 WHAT IS A SESSION? ---- 90
7.1.1 HOW SESSION WORKS ---- 90
7.2 COOKIE ---- 91
7.2.1 TYPES OF COOKIES ---- 92
7.2.2 WHAT IS THE DIFFERENCE BETWEEN SESSION COOKIES AND PERSISTENT COOKIES? ---- 93
7.3 URL REWRITING ---- 93
7.4 HIDDEN FIELD ---- 94
7.5 WORKING WITH HTTP SESSION ---- 95
PART I: SHORT QUESTIONS ---- 96
PART II: LONG QUESTIONS ---- 96
MULTIPLE CHOICE QUESTIONS ---- 97
REFERENCES ---- 100

CHAPTER NO. 08 ---- **101**

JDBC PROGRAMMING ---- **101**

8.1 INTRODUCTION JDBC PROGRAMMING ---- 101
8.1.1 WHY SHOULD WE USE JDBC ---- 102
8.1.2 WHAT IS API ---- 102
8.2 CONFIGURE MYSQL ---- 103
8.2.1 INTRODUCTION TO MY SQL ---- 103
8.2.2 FEATURES OF MY-SQL ---- 104
8.2.3 BENEFITS OF MYSQL: ---- 104
8.2.4 MYSQL INSTALLATION ON WINDOWS ---- 105

8.3 CONNECTING TO THE DATABASE ---------- 106
8.3.1 IMPORT JDBC PACKAGES ---------- 107
8.3.2 REGISTER JDBC DRIVER ---------- 107
8.3.3 DATABASE URL FORMULATION ---------- 108
8.3.4 CREATE CONNECTION OBJECT ---------- 109
8.3.5 CLOSING JDBC CONNECTIONS ---------- 109
8.4 ACCESSING DATA ---------- 110
8.4.1 SEQUENTIAL ACCESS ---------- 110
8.4.2 RANDOM ACCESS ---------- 110
8.5 THE EXECUTE QUERY METHOD ---------- 110
8.5.1 THE SQL SELECT STATEMENT ---------- 110
8.6 THE EXECUTE UPDATE METHOD ---------- 112
8.6.1 DML: ---------- 112
8.6.2 DDL ---------- 113
PART I: SHORT QUESTIONS ---------- 115
PART II: LONG QUESTIONS ---------- 115
MULTIPLE CHOICE QUESTIONS ---------- 116
REFERENCES ---------- 119

CHAPTER NO. 09 ---------- 120

CUSTOM TAGS ---------- 120

9.1 CUSTOM TAGS ---------- 120
9.1.1 CREATE "HELLO" TAG ---------- 121
9.2 TAG HANDLER ---------- 121
9.2.1 SIMPLE TAG HANDLERS ---------- 121
9.2.2 CLASSIC TAG HANDLER ---------- 122
9.3.1 TAGS WITH ATTRIBUTES ---------- 123
PART I: SHORT QUESTIONS ---------- 124
PART III: LONG QUESTIONS ---------- 124
MULTIPLE CHOICE QUESTIONS ---------- 125
REFERENCES ---------- 127

CHAPTER NO. 10 ---------- 128

JAVA SERVER FACES ---------- 128

10.1 INTRODUCTION -- 128
10.1.1 BENEFITS OF JAVA SERVER FACES --- 128
10.2 USING JSF WITH JSP --- 129
10.3 JSF LIFE CYCLE -- 130
10.4 EVENTS HANDLING -- 132
10.5 JSF TAG LIBRARY -- 133
PART I: SHORT QUESTIONS -- 136
PART II: LONG QUESTIONS--- 136
MULTIPLE CHOICE QUESTIONS -- 137
REFERENCES-- 139

Chapter No. 01

INTRODUCTION TO WEB DEVELOPMENT

At the end of this chapter students will be able to understand:

1.1- What is a Web Application?
1.2- HTTP Basics
1.3- Types of HTTP Requests
1.4- Server Side Programming
1.5- Client Side Programming
1.6- Web Application Layers

1.1 WHAT IS A WEB APPLICATION?

A web application or a web app is a software application that runs on server, a web application uses web browsers and web technology to perform tasks over internet. A web application does not run on a local computer operating system like other software applications.

A web application is a program that allows user to perform actions online. Many businesses use these programs to communicate with customers, sell products and improve work processes. Web apps are client-server programs so each program has a client-side and a server-side.

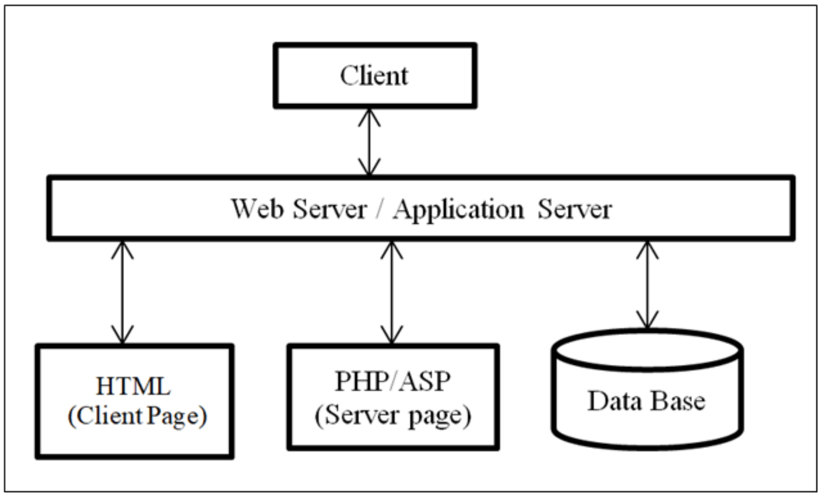


Figure 1: Web Application General Structure

1.1.1 How a web application works

Web applications are usually coded in browser-supported language such as JavaScript and HTML as these languages rely on the browser to make the program executable. Some applications are dynamic (requiring server-side processing) while others are completely static (with no processing required at the server).
The web application requires a web server (to manage requests from the client), an application server (to perform the tasks requested), and sometimes, a database (to store the information). Application server technology ranges from ASP.NET, ASP and ColdFusion, to PHP and JSP.

- **Example of a web application**

Web applications include online forms, shopping carts, word processors, spreadsheets, video and photo editing, file conversion, file scanning, and email programs such as Gmail, Yahoo and AOL. Popular applications include Google Apps and Microsoft 365. Google Apps for Work has Gmail, Google Docs, Google Sheets, Google Slides, online storage and more. Other functionalities include online sharing of documents and calendars. Online retail sales, online auctions, wikis, instant messaging services are some other web applications.

1.1.2 Benefits of a web application

- Web applications run on multiple platforms regardless of OS or device as long as the browser is compatible
- All users access the same version, removing any compatibility issues
- They are not installed on the hard drive, thus removing space limitations on a local computer
- They reduce software piracy in subscription-based web applications i.e. SaaS (Software as a Service)
- They reduce costs for both the business and end user as there is less support and maintenance required by the business and lower requirements for the end user's computer

Some terminologies regarding Web App are:

Web Page: A web page is a document that is suitable for the World Wide Web and web browsers. A web browser displays a web page on a monitor or mobile device. Web page is a document commonly written in Hyper Text Markup Language (HTML) that is accessible through the Internet or other network using an Internet browser.

Website: A set of related web pages located under a single domain name, and published on at least one web server.

For example www.tevta.gop.pk

HTML: HTML stands for Hyper Text Makrup Language. It is a standard markup language for creating Web Pages. It describe the structure of web pages using markups. HTML elements are the building blocks of HTML pages. HTML elements are represented by tags. HTML tags label pieces of content such as "heading", "paragraph", "table", and so on. Browsers do not display the HTML tags, but use them to render the content of the page.

For Example:

```
<!DOCTYPE.html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

JAVASCRIPT: JavaScript is a programming language commonly used in web development.

Client-Server: Client-server is a relationship in which one program (the client) requests a service or resource from another program (the server).

1.2 HTTP BASICS

HTTP (Hypertext Transfer Protocol) is the set of rules for transferring files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. As soon as a Web user opens their Web browser, the user is indirectly making use of HTTP. HTTP is an application protocol that runs on top of the TCP/IP suite of protocols (the foundation protocols for the Internet). Latest version of HTTP is HTTP/2.

- **World Wide Web**

The World Wide Web, commonly referred to as WWW, W3, or the Web is a system of interconnected webpages accessible through the Internet. **WWW** is about communication between web **clients** and **servers.** Communication between client computers and web servers is done by sending **HTTP Requests** and receiving **HTTP Responses.**

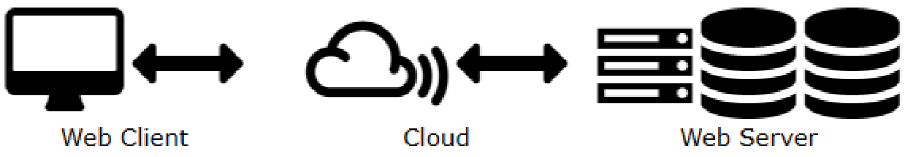


Figure 2: World Wide Web Communication

- **Protocol:** When computers communicate with each other, there needs to be a common set of rules and instructions that each computer follows. A specific set of communication rules is called a protocol. Some examples of these different protocols include PPP, TCP/IP, SLIP, HTTP, and FTP.

1.2.1 How HTTP works?

HTTP is a request response protocol to communicate asynchronously between client and server. Mostly in HTTP a browser acts as a client and a web-server like Apache or IIS acts as server. The server which hosts the files (like html, audio, video files, etc.) responses to the client. Depending on the request a response contains the status of the request.

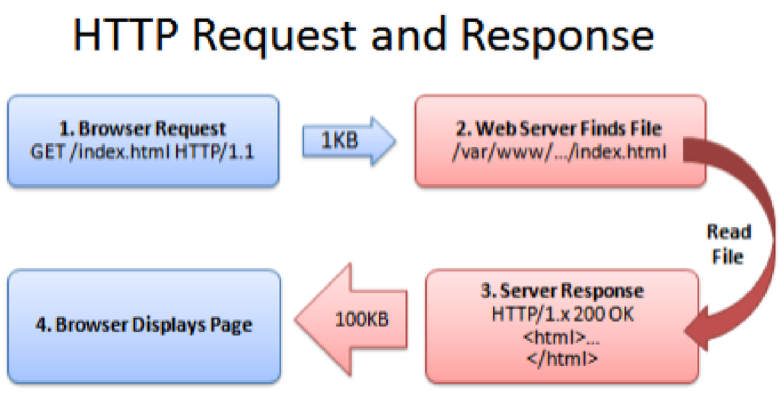


Figure 3: How HTTP Works?

A client (browser) submits an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

1.3 TYPES OF HTTP REQUESTS

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource. The method designates the type of request being made to the web server.

1.3.1 HTTP Requests and Responses

Each interaction between the client and server is called a message. HTTP messages are requests or responses. Client devices submit HTTP requests to servers, which reply by sending HTTP responses back to the clients.

1.3.2 Request Methods:

The most common and widely used request methods are GET and POST but there are many others, including HEAD, PUT, DELETE, CONNECT, and OPTIONS. HTTP Methods\Requests are:

1-**GET** is used to request data from a specified resource. GET is one of the most common HTTP methods. The query string (name/value pairs) is sent in the URL of a GET request: For example,
GET /test/demo_form.php?name1=value1&name2=value2

2-**POST** is used to send data to a server to create/update a resource. The data sent to the server with POST is stored in the request body of the HTTP request:

For example,

```
POST /test/demo_form.php HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

3-**PUT** is used to send data to a server to create/update a resource. The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.

4-**HEAD** is almost identical to GET, but without the response body. In other words, if GET /users returns a list of users, then HEAD /users will make the same request but will not return the list of users. HEAD requests are useful for checking what a GET request will return before actually making a GET request - like before downloading a large file or response body.

1.3.3 GET Vs POST Methods

Following table compares GET and POST Methods.

GET	**POST**
Parameters in this method are saved in the browser's history	Parameters are not archived in the browser history or other web server logs
Can be bookmarked	Cannot be bookmarked
Features a restriction on data length. This is because the GET method adds data to the URL for it to be sent, and we know the maximum URL length is 2048 characters	There are no restriction on data length
There is no impact when you hit the reload/back button.	Should you hit the reload/back button, sent data will be resubmitted
Has restriction on data type as the only allowed data type is ASCII	There is no restriction on data type, and binary data is also allowed

characters	
Information is visible to everyone in the URL	Information is not displayed in the URL thus not visible to everyone
Can be cached	Can't be cached

1.4 SERVER SIDE PROGRAMMING

It is the program that runs on server dealing with the generation of content of web page. Or we can say that server-side programming must deal with dynamic content. It runs on the server. Most web pages are not static since they deal with searching databases.

1.4.1 Server-side Uses

- It processes the user input. For example if user input is a text in search box, run a search algorithm on data stored on server and send the results
- Displays the requested pages
- Structure of web applications
- Interaction with servers/storages
- Interaction with databases
- Querying the database
- Encoding of data into HTML
- Operations over databases like delete, update.

1.4.2 Server Side Programming Languages

The Programming languages for server-side programming are:

- PHP
- C++
- Java and JSP

PHP: Hypertext Preprocessor (or simply PHP) is a server-side scripting language designed for Web development, but also used as a general-purpose programming language. PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks. For example:

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>PHP Test</title>
</head>
<body>
<?php echo '<p>Hello World</p>'; ?>
</body>

</html>
```

C++: C++ is a general purpose language in computer programming. It is a middle level language that can be used for several purposes in the computer industry. C++ language may not always be the most preferred programming language, but there are many aspects of software programming that cannot be done without the use of C++.

Java and JSP:

Java - Java is a programming language that produces software for multiple platforms. When a programmer writes a Java application, the compiled code (known as bytecode) runs on most operating systems (OS), including Windows, Linux and Mac OS.

JSP - Java Server Page (JSP) is a technology for controlling the content or appearance of Web pages through the use of servlets, small programs that are specified in the Web page and run on the Web server to modify the Web page before it is sent to the user who requested it.

Java Servlet - Servlet technology is used to create web application (resides at server side and generates dynamic web page).

1.5 CLIENT SIDE PROGRAMMING

It is the program that runs on the client machine (browser) and deals with the user interface/display and any other processing that can happen on client machine like reading/writing cookies.

We can say that client-side programming mostly deals with the user interface with which the user interacts in the web. It is mostly a browser, in the user's machine, that runs the code and is mainly done in any scripting language like JavaScript.

1.5.1 Client-side Uses

- Makes interactive web pages

- Make stuff work dynamically
- Interact with temporary storage
- Works as an interface between user and server
- Sends requests to the server
- Retrieval of data from Server
- Interact with local storage
- Provides remote access for client-server program

1.5.2 Client Side Programming Language Examples

There are many client-side scripting languages too.

- JavaScript
- HTML (Structure)
- CSS (Designing)
- jQuery etc.

JavaScript: JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

HTML: HTML is the standard markup language for creating Web pages.
HTML stands for Hyper Text Markup Language. HTML describes the structure of Web pages using markup. HTML elements are the building blocks of HTML pages

CSS: stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. External style sheets are stored in CSS files.

JQuery: It is a JavaScript library, not a programming language, but it is an essential element of client-side scripting. The creators say that JQuery behaves consistently across all browsers. This is the most significant benefit of this technique. JQuery is a JavaScript toolkit that makes coding easier and allows for faster web development.

1.6 WEB APPLICATION LAYERS

1.6.1 The Three Layer Model:-

- **What is three-tier architecture?**

Three-tier architecture is a well-established software application architecture that organizes applications into three logical and physical computing tiers: the presentation tier, or user interface; the application tier, where data is processed; and the data tier, where the data associated with the application is stored and managed.

The chief benefit of three-tier architecture is that because each tier runs on its own infrastructure, each tier can be developed simultaneously by a separate development team, and can be updated or scaled as needed without impacting the other tiers. The three tiers in detail:

- **Presentation tier/ Layer**

The presentation tier is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect information from the user. This top-level tier can run on a web browser, as desktop application, or a graphical user interface (GUI), for example. Web presentation tiers are usually developed using HTML, CSS and JavaScript. Desktop applications can be written in a variety of languages depending on the platform.

- **Application tier/ Layer**

The application tier, also known as the logic tier or middle tier, is the heart of the application. In this tier, information collected in the presentation tier is processed - sometimes against other information in the data tier - using business logic, a specific set of business rules. The application tier can also add, delete or modify data in the data tier.

The application tier is typically developed using Python, Java, Perl, PHP or Ruby, and communicates with the data tier using API calls.

- **Data tier/ Layer**

The data tier, sometimes called database tier, data access tier or back-end, is where the information processed by the application is stored and managed. This can be a relational database management system such as PostgreSQL, MySQL, MariaDB, Oracle, DB2, Informix or Microsoft SQL Server, or in a NoSQL Database server such as Cassandra, CouchDB or MongoDB.

In a three-tier application, all communication goes through the application tier. The presentation tier and the data tier cannot communicate directly with one another.

1.6.2 Benefits of three-tier architecture

- **Logical and physical separation of functionality:** Each tier can run on a separate operating system and server platform. And each tier runs on at least one dedicated server hardware or virtual server.
- **Faster development**: An organization can bring the application to market faster, and programmers can use the latest and best languages and tools for each tier.
- **Improved scalability**: Any tier can be scaled independently of the others as needed.
- **Improved reliability**: An outage in one tier is less likely to impact the availability or performance of the other tiers.
- **Improved security**: The presentation tier and data tier can't communicate directly, a well-designed application tier can function as a sort of internal firewall.

EXERCISE

PART I: SHORT QUESTIONS

1. What is a web Application?
2. What is a website and a web page?
3. What is HTML?
4. What is HTTP?
5. What is an HTTP Request?
6. What is a GET method?
7. Define POST method?
8. What is server side programming? Write names of server side programming languages.
9. What is client side programming? Write names of some client side programming languages.
10. Define JavaScript?
11. What is an application layer?
12. Write two benefits of three layer model of web application layers.

PART II: LONG QUESTIONS

1. What is a web application? Describe in detail.
2. What is HTTP? How it Works?
3. Compare HTTP GET and POST methods.
4. Describe about server side programming languages.
5. Describe about web application layers.

MULTIPLE CHOICE QUESTIONS

1. A _______ application is a software application that runs on server.

a. user application b. software application
c. web application d. graphics application

2. HTML stands for

a. Hypertext Transfer Language b. Hypertext Markup Language
c. Hyper Tech Markup Language d. Hyper Terminal Markup Language

3. What should be the first tag in any HTML document?

a. <head> b. <title> c. <html> d. <document>

4. HTTP stands for:

a. Hypertext Transmission Protocol b. Hyper Terminal Transfer Protocol
c. Hyper Tech Transfer Protocol d. Hyper Text Transfer Protocol

5. HTML web pages can be read and rendered by _________.

a. Compiler b. Server c. Web Browser d. Interpreter

6. Which of the following is not a browser?

a. WhatsApp b. Google Chrome
c. Mozilla Firefox d. Opera

7. Which of the following is a server side Programming Language.

a. C++ b. HTML c. VBScript d. JavaScript

8. Which of the following is not a web application layer.

a. Data Layer b. Presentation Layer
c. Application Layer d. Network Layer

9. Which of the following is a scripting language.

a. C++ b. JAVA c. PHP d. JavaScript

10. A _________ is a program that runs on the client machine (browser).

a. A client side program b. Server side Program
c. Web Program d. all

MCQ's Answers:

1	2	3	4	5	6	7	8	9	10
C	B	A	D	C	A	A	D	D	A

References

client-server model (client-server architecture). (n.d.). Retrieved from www.techtarget.com: https://www.techtarget.com/searchnetworking/definition/client-server#:~:text=Client%2Dserver%20is%20a%20relationship,another%20program%20(the%20server).

client-side-scripting-top-languages-to-learn. (n.d.). Retrieved from www.crampete.com: https://www.crampete.com/blogs/client-side-scripting-top-languages-to-learn/#:~:text=Javascript%2C%20HTML%2C%20CSS%2C%20VB,used%20for%20client%2Dside%20scripting.&text=It%20has%20full%20access%20to,files%20on%20the%20web%20server.

HTTP (Hypertext Transfer Protocol) . (n.d.). Retrieved from www.techtarget.com: https://www.techtarget.com/whatis/definition/HTTP-Hypertext-Transfer-Protocol

http request methods. (n.d.). Retrieved from rapidapi.com: https://rapidapi.com/blog/api-glossary/http-request-methods/

introduction to server side programming languages. (n.d.). Retrieved from www.crampete.com: https://www.crampete.com/blogs/introduction-to-server-side-programming-languages/

three-tier-architecture. (n.d.). Retrieved from www.ibm.com: https://www.ibm.com/topics/three-tier-architecture

Web Application: General Structure|Download Scientific Diagram. (n.d.). Retrieved from www.researchgate.net: https://www.researchgate.net/figure/Web-Application-General-Structure_fig1_339302445

What is a Web Application. (n.d.). Retrieved from www.stackpath.com: https://www.stackpath.com/edge-academy/what-is-a-web-application/

What is HTTP? (n.d.). Retrieved from www.w3schools.com: https://www.w3schools.com/whatis/whatis_http.asp

Chapter No. 02

SETTING UP DEVELOPMENT ENVIRONMENT

At the end of this chapter students will be able to:

2.1- Install Apache Tomcat Server
2.2- Install IDE (Eclipse / NetBeans)
2.3- Describe Standard Directory Structure of Java Web Application
2.4- Development of Web Application

2.1 INSTALLATION OF APACHE TOMCAT SERVER

2.1.1 What is Tomcat?

It is an open-source Java servlet container that implements many Java Enterprise Specs such as the Websites API, Java-Server Pages and the Java Servlet. The complete name of Tomcat is "Apache Tomcat" it was developed in an open, participatory environment and released in 1998 for the very first time. It is one of the most widely used java-server due to several capabilities such as good extensibility, proven core engine, and well-test and durable.

2.1.2 Installation Process of Apache Tomcat Server

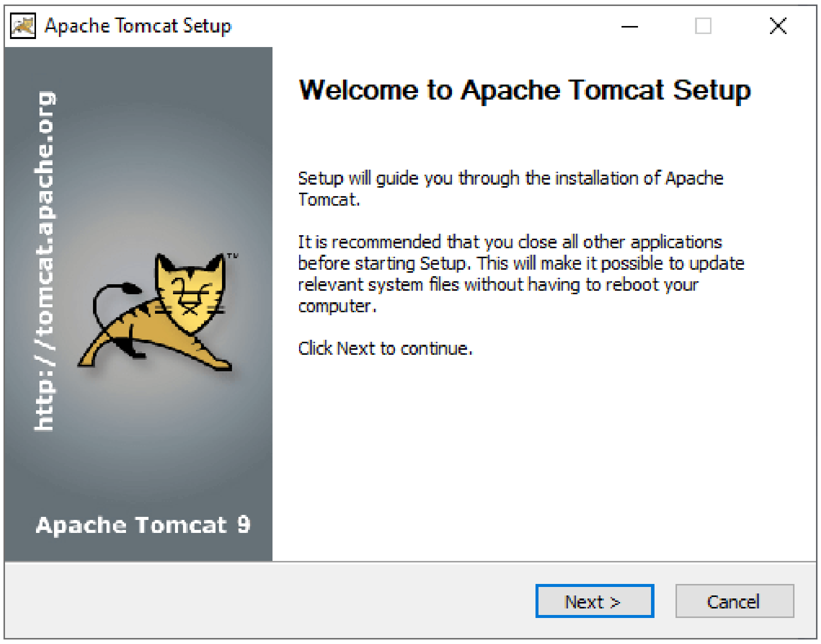
Before installing tomcat, make sure you have JDK (java development kit) you can use the following given steps for downloading and installing the Tomcat on your system:

Step 1. Open the Google Chrome or any of your web browser and type "download Tomcat for windows" in the search box. You can also go directly on Tomcat's website by clicking on this https://tomcat.apache.org/download-90.cgi#9.0.38

Now download any version of Tomcat you want:

Step 2. Go to Download and click on the downloaded file and wait for little until the installation process starts.

Step 3. Once the installation process gets started, click on the **"Next"** button, as shown below:



Apache Tomcat Setup

Welcome to Apache Tomcat Setup

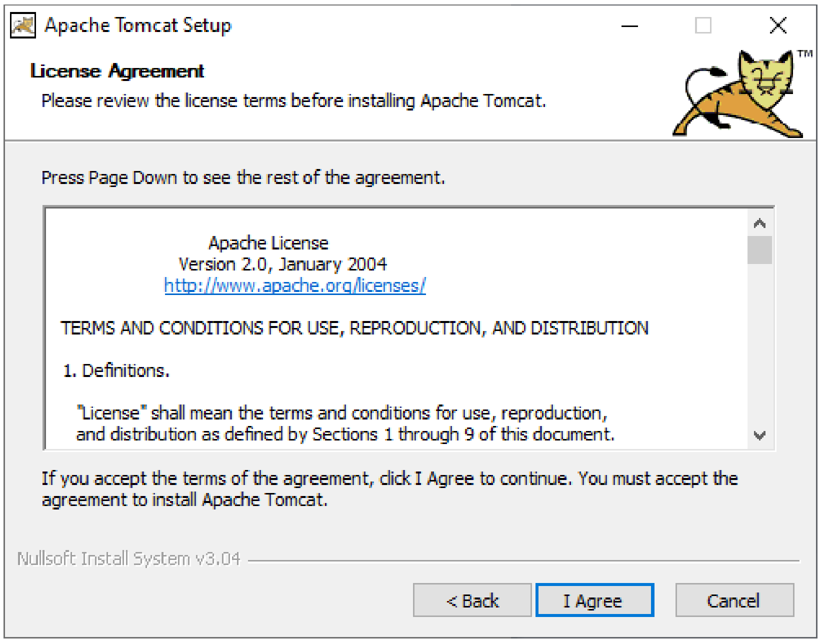
Setup will guide you through the installation of Apache Tomcat.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 1: Installation Steps

Step 4. Click on the button labeled as **"I Agree."**



Apache Tomcat Setup

License Agreement

Please review the license terms before installing Apache Tomcat.

Press Page Down to see the rest of the agreement.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

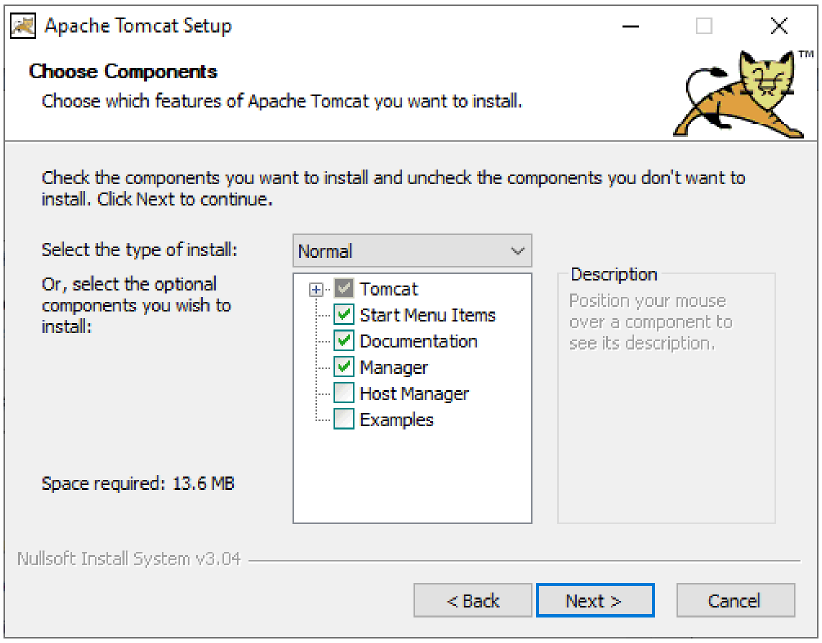
1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Apache Tomcat.

Figure 2: Installation Steps Continued...

Step 5. Click on the **"Next"** button.


Apache Tomcat Setup

Choose Components

Choose which features of Apache Tomcat you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select the type of install: Normal

Or, select the optional components you wish to install:

- [✓] Tomcat
- [✓] Start Menu Items
- [✓] Documentation
- [✓] Manager
- [ ] Host Manager
- [ ] Examples

Description

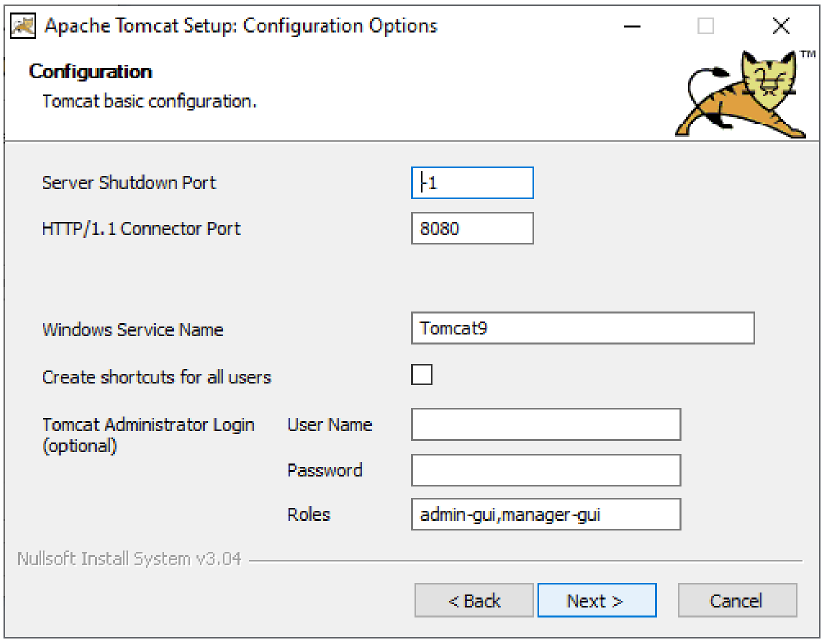
Position your mouse over a component to see its description.

Space required: 13.6 MB

Nullsoft Install System v3.04

Figure 3: Installation Steps continued...

Step 6. Enter the user name and password and click on the **"Next"** button, as shown below:


Apache Tomcat Setup: Configuration Options

Configuration

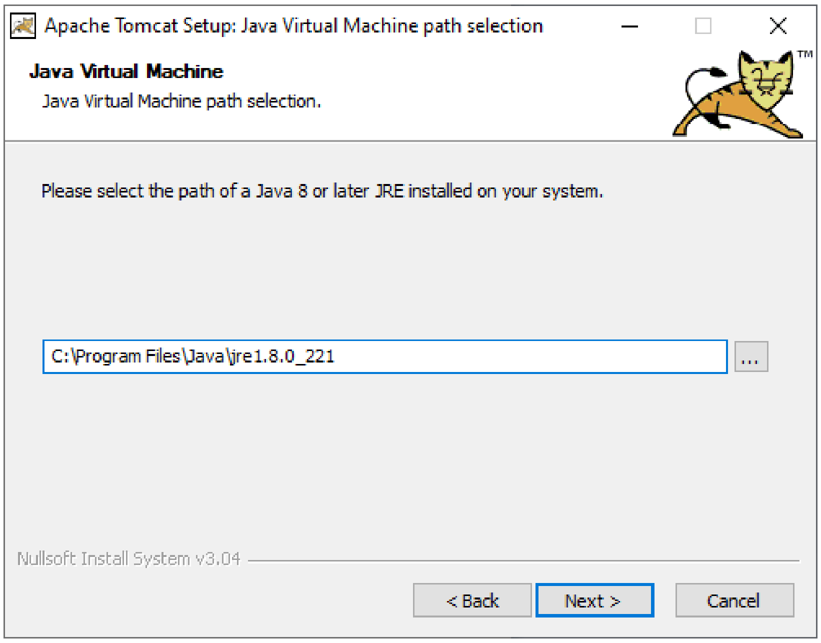
Tomcat basic configuration.

Server Shutdown Port		-1
HTTP/1.1 Connector Port		8080
Windows Service Name		Tomcat9
Create shortcuts for all users		[ ]
Tomcat Administrator Login (optional)	User Name	
	Password	
	Roles	admin-gui,manager-gui

Nullsoft Install System v3.04

Figure 4: Installation Steps continued...

Step 7. Then give path for Java jdk/ jre, and click on the **"Next"** button again


Apache Tomcat Setup: Java Virtual Machine path selection

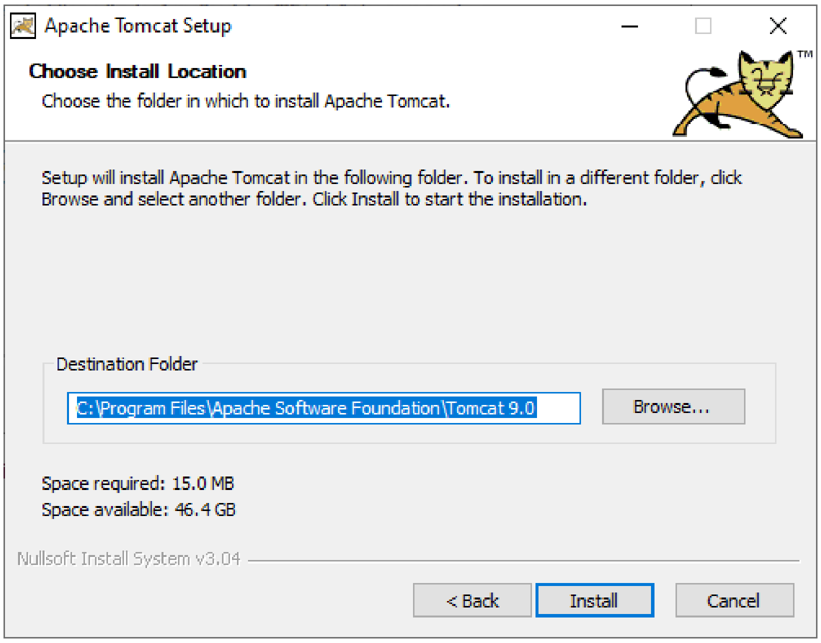
Java Virtual Machine
Java Virtual Machine path selection.

Please select the path of a Java 8 or later JRE installed on your system.

C:\Program Files\Java\jre1.8.0_221

Figure 5: Installation Steps continued...

Step 8. Now click on the **"Install"** button.


Apache Tomcat Setup

Choose Install Location
Choose the folder in which to install Apache Tomcat.

Setup will install Apache Tomcat in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder
C:\Program Files\Apache Software Foundation\Tomcat 9.0

Space required: 15.0 MB
Space available: 46.4 GB

Figure 6: Start Installing Apache Tomcat

Wait for some time until the Tomcat gets installed.

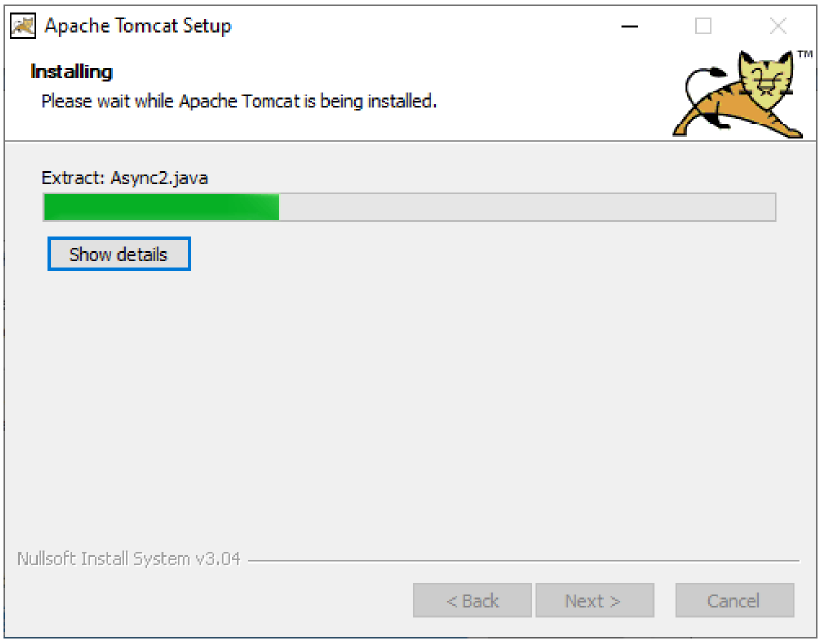
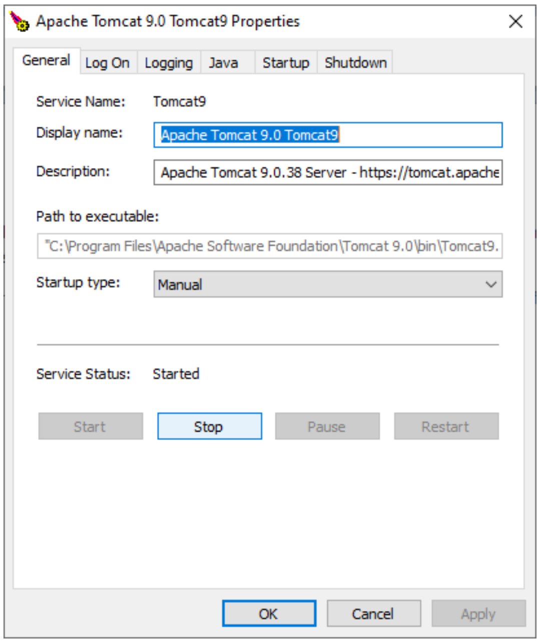

Apache Tomcat Setup
Installing
Please wait while Apache Tomcat is being installed.
Extract: Async2.java
Show details
Nullsoft Install System v3.04
< Back
Next >
Cancel

Figure 7: Extracting Files

Step 9. Now click on the **"Finish"** button, here the installation of Tomcat is completed. It may ask you to restart your system, so restart your system.

There will be no specific icon for tomcat on your windows because it is only services so you have to start these servicesyou can see this service in Bin folder in c:\Apache Software Foundation\Tomcat 9.0. Open tomcat9w.exe. Apache Tomcat 9.0 Tomcat 9.0 Properties will be open you can start or stop the service with the help of start/ stop button. After starting service you can open localhost:8080 in your web



Service Name: Tomcat9
Display name: Apache Tomcat 9.0 Tomcat9
Description: Apache Tomcat 9.0.38 Server - https://tomcat.apache
Path to executable:
"C:\Program Files\Apache Software Foundation\Tomcat 9.0\bin\Tomcat9.
Startup type: Manual
Service Status: Started

Figure 8: Starting Tomcat 9.0 Service

browser.

2.1.3 Advantages of Tomcat

Some significant advantages of Tomcat are as follows:

- **It is open-source:** It means anyone from anywhere can download, install, and use it free of cost.
- **Incredibly Lightweight:** It is actually a very light application. However, itprovides all necessary and standard functionalities required to operate a server.
- **Highly flexible:** It offers high flexibility, a user can run it in any fashion he wants, and it will still work as fine without any issues.
- **Stability:** It is one of the most stable platforms available today to build on and using it to run our applications.
- **It provides us an extra level of security:** Tomcat's installation is behind the protection of an extra firewall which can be accessible only from the Apache installation.
- **It is well documented:** It has excellent documentation available, including a vast range of freely available online tutorials that can be downloaded or viewed directly online by the user.

2.1.4 Disadvantages of Tomcat

- It is not as fast as the Apache if we are working with the static pages
- It has some issues like a memory leak
- It's way to handle the logs.
- Issues in the SSL installations
- Its user interface is inferior and basic.

2.2 INSTALATION OF IDE (ECLIPSE/ NETBEANS)

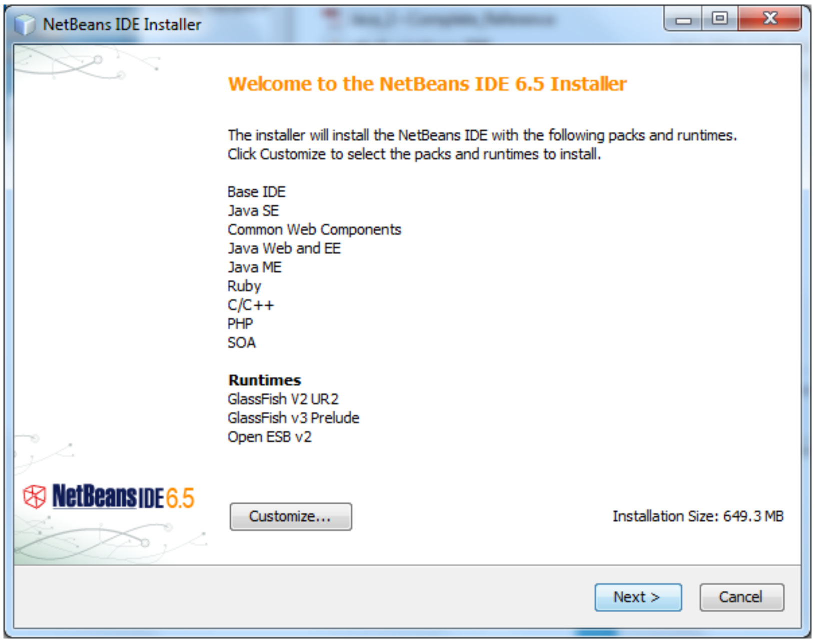
In the Java development community, there are two main Integrated Development Environments (IDEs): Eclipse, which is associated with IBM and NetBeans which is a Sun/ Oracle environment.

NETBEANS: is an integrated development environment for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Microsoft Windows, macOS, Linux and Solaris.

ECLIPSE: is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including C, C++, COBOL, FORTRAN, and PHP.

2.1.1 Installation Process of NetBeans:

1. You need to have a setup file of the NetBean. If you don't have the setup you can download from the following link: https://netbeans.org/images_www/v6/download/community/8.2
2. You can download any type of setup as per your requirements from the above mention web page.
3. Click on the next option


NetBeans IDE Installer

Welcome to the NetBeans IDE 6.5 Installer

The installer will install the NetBeans IDE with the following packs and runtimes.
Click Customize to select the packs and runtimes to install.

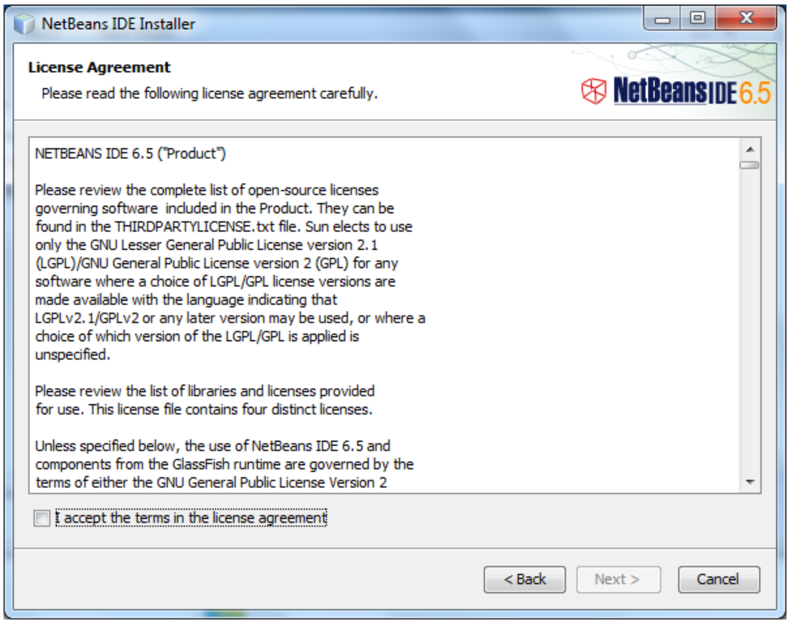
Base IDE
Java SE
Common Web Components
Java Web and EE
Java ME
Ruby
C/C++
PHP
SOA

Runtimes
GlassFish V2 UR2
GlassFish v3 Prelude
Open ESB v2

Installation Size: 649.3 MB

Figure 09: Installation steps for NetBeans

4. Check on the **"I accept"** option and click on the "**Next"** button.



NetBeans IDE Installer

License Agreement

Please read the following license agreement carefully.

NETBEANS IDE 6.5 ("Product")

Please review the complete list of open-source licenses governing software included in the Product. They can be found in the THIRDPARTYLICENSE.txt file. Sun elects to use only the GNU Lesser General Public License version 2.1 (LGPL)/GNU General Public License version 2 (GPL) for any software where a choice of LGPL/GPL license versions are made available with the language indicating that LGPLv2.1/GPLv2 or any later version may be used, or where a choice of which version of the LGPL/GPL is applied is unspecified.

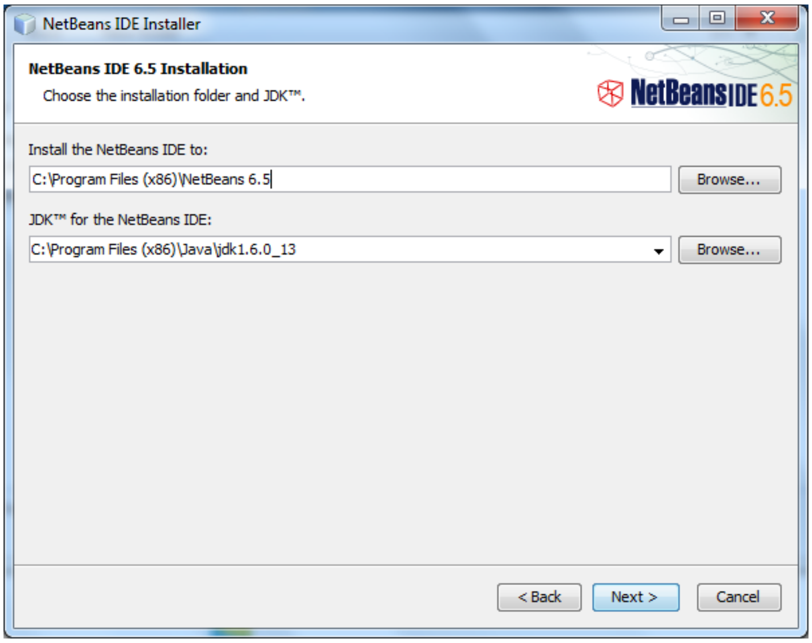
Please review the list of libraries and licenses provided for use. This license file contains four distinct licenses.

Unless specified below, the use of NetBeans IDE 6.5 and components from the GlassFish runtime are governed by the terms of either the GNU General Public License Version 2

☐ I accept the terms in the license agreement

Figure 10: Accept the License Agreement

5. Select the path where you want to install the software and press the "**Next"** button.



NetBeans IDE Installer

NetBeans IDE 6.5 Installation

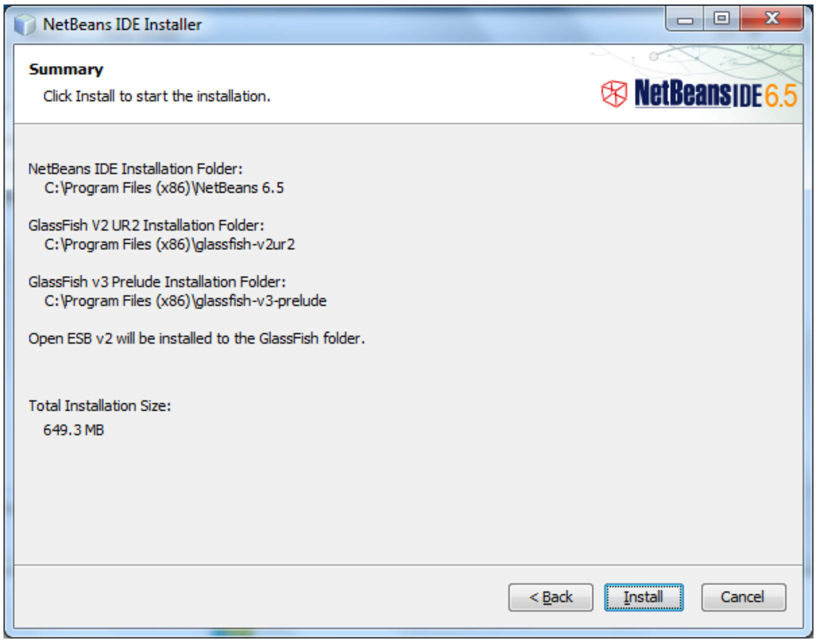
Choose the installation folder and JDK™.

Install the NetBeans IDE to: C:\Program Files (x86)\NetBeans 6.5

JDK™ for the NetBeans IDE: C:\Program Files (x86)\Java\jdk1.6.0_13

Figure 11: Selecting Path for JDK

6. Click on the "**Install**" button.


NetBeans IDE Installer

Summary

Click Install to start the installation.

NetBeans IDE Installation Folder:
C:\Program Files (x86)\NetBeans 6.5

GlassFish V2 UR2 Installation Folder:
C:\Program Files (x86)\glassfish-v2ur2

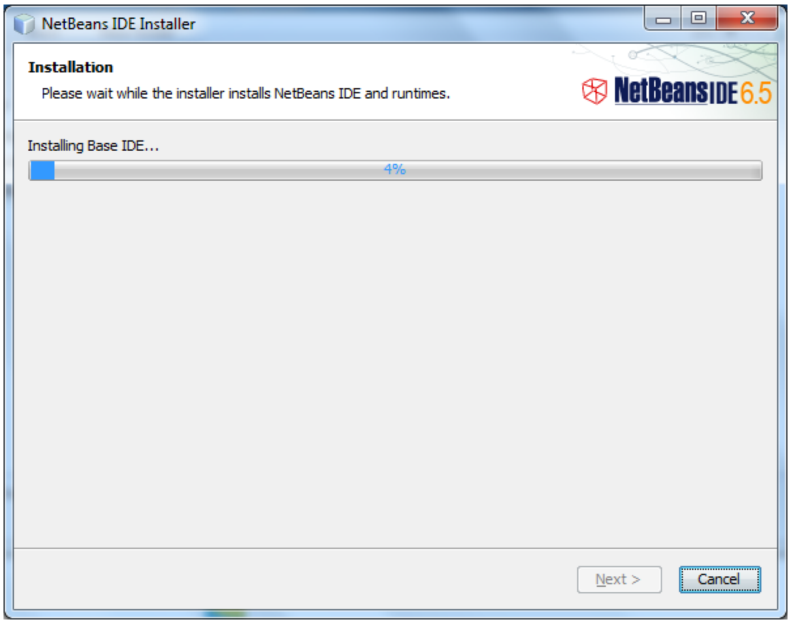
GlassFish v3 Prelude Installation Folder:
C:\Program Files (x86)\glassfish-v3-prelude

Open ESB v2 will be installed to the GlassFish folder.

Total Installation Size:
649.3 MB

Figure 12: Installation process started

7. Wait for the while till the time the setup is properly installed into the Computer


NetBeans IDE Installer

Installation

Please wait while the installer installs NetBeans IDE and runtimes.

Installing Base IDE...

4%

Figure 13: Wait for Complete Installation

8. After complication of the setup you can click on the "**Finish"** button. Now you can start the NetBeans for further use.

2.1.2 Installation Process of Eclipse:

1. In the first step, Open your browser and navigate to the URL https://www.eclipse.org
2. Then, click on the "Download" button to download Eclipse IDE.
3. After downloading click on the "*eclipse-inst-jre-win64.exe*" file to install Eclipse IDE.
4. Then, click on "Eclipse IDE for Java Developers".
5. Then, click on the "Install" button.
6. Now click on "Create a new Java project".

Now, you are ready to make new Java projects using eclipse IDE.

2.2 STANDARD DIRECTORY STRUCTURE OF JAVA WEB APPLICATION

A web application typically consists of a structured hierarchy of directories. Within the web application directory hierarchy, a special directory named WEB-INF must be created. However, the resources in the WEB-INF directory are visible to servlets and Java classes that reside within the web application.

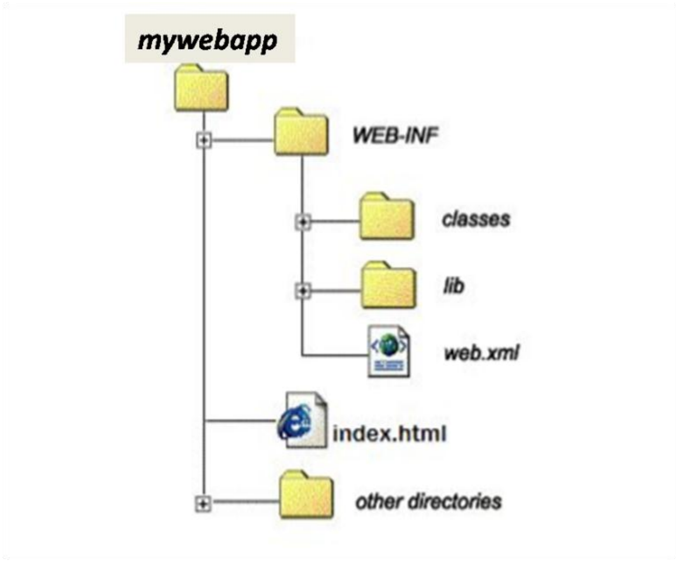


Figure 14: Web Application Directory Structure

In order to run your java web application (servlet, JSP, etc.) we need webserver. Before running the application you need to package the resources inside it (servlets, SP's, xml files, etc.) in a standardized way as shown below.

The Root Directory: The root directory of your web application can have any name. In the above example the root directory name is mywebapp. Under the root directory, you can put all files that should be accessible in your web application.

This is the main or Root folder of web application. Usually name of this folder becomes your web application context. For example, if our web application name is mywebapp, then folder name mywebapp and web application will be accessible via *http://localhost:8080/mywebapp*

The WEB-INF Directory: The WEB-INF directory is located just below the web app root directory. This directory is a Meta information directory. Files stored here are not supposed to be accessible from a browser (although your web app can access them internally, in your code). Inside the WEB-INF directory there are two important directories (classes and lib, and one important file(web.xml). These are described below.

1. **web.xml (Deployment Descriptor):** The web.xml file contains information about the web application, which is used by the Java webserver / servlet container in order to properly deploy and execute the web application. For instance, the web.xml contains information about which servlets a web application should deploy, and what URL's they should be mapped to.

2. **Classes Directory**: The classes directory contains all compiled Java classes that are part of your web application. The classes should be located in a directory structure matching their package structure.

3.**lib folder:** The lib directory contains all JAR files used by your web application. This directory most often contains any third party libraries that your application is using. You could, however, also put your own classes into a JAR file, and locate it here, rather than putting those classes in the classes directory.

WAR (Web Application Archive) File:

Once the web application is developed, it needs to be deployed on server. In WAR (web archive) the web application has to be packaged. Once the packaged war file is deployed to the server, it gets exploded and deploys to the server.

The WAR file is a standard format for web applications that has specific directories and specific files. This includes a WEB-INF directory, a WEB-INF/web.xml file used to describe the application, a WEB-INF/lib directory for JAR files used by the application.

2.3 DEVELOPMENT OF WEB APPLICATION

Web app development is the process of building an application program; to be specific, an application program that is stored on a server, delivered to users through an active internet connection, via a web browser. Since web apps are accessed online, users do not need to download, or store, web apps on their device - be it a desktop workstation, laptop, or mobile device - in order to run them.

Web development makes sure the web app offers compatibility with both Android and iOS, Identifies life cycle and optimization metrics, builds an intelligent, iterative user interface.

2.3.1 Web Application Development Stages

There are many steps involved in the web application development process. Most web applications are built using a web application framework that simplifies code and helps reduce errors. Popular web application frameworks include:

Front-end: Front-end development means building the front-end portion of websites and web applications. Front-end frame works are: Svelte, Vue.js, React, etc.

Back-end: Back-end development comprises a site's structure, system, data, and logic. Laravel, Ruby on Rails, Django are some back-end frame works. Some web application frameworks, such as ASP.net, handle both front- and back-end duties.

A web application development process will include the following steps:

1. Requirements review & proposal
2. Planning & blueprints
3. Web application design
4. Copywriting & labeling
5. Web application programming
6. Testing & launch
7. Application maintenance

1. Requirements Review & Proposal

Entrepreneurs and businesses mostly start with a set of ideas for their web application. The detailed document of these ideas includes application goals, features, technology, budget, vision, and future plans. Through this document the development team gets a clear understanding of your app objectives, key goals, target audience, focus industry, and other critical elements. The discussions and questionnaires on these objectives help web developers get further clarity into project goals. Once the application development team has 100% clarity the proposal is prepared.

2. Planning & Blueprints

In this stage the roadmap is created that will be followed to build it. With the help of first stage of the web app development model, developers create a blueprint including flowcharts and sketches that helps determine the overall structure of the web application.

Flowcharts\ Sitemaps - show the relationship between different web pages and help understand how the inner structure of your website will work &look like.

Top web app development teams keep clients in loop during this stage to make sure that the core of the application comes out correctly. The time spent on web app varies depending on the complexity of the web application.

3. Web Application Design

Web application users don’t know what happens behind the frontend of the app. They interact only with the design part of the application. In this stage Designers work with color schemes, graphics, icons, templates, user experience, style guides, transitions, buttons, and much more to finalize the design aspect of the web application.

After finalizing the initial mockups, they are shared with clients for review & feedback. The design iterations and mockup changes go on until the client is satisfied.

4. Web Application Programming

In this stage, frameworks and APIS are developed, app features are built, security layers are added, payment gateways are integrated, and lots of other capabilities are crafted. Stage 4 forms the biggest chunk of the web application model.

5. Copywrite& Labeling

Copywrite and labeling are less than 5% of the application development work but without it, it is hard to make sense of everything you have built. Stage 5 is about finalizing the headlines, captions, labeling, copy, and everything else that's in the text form. The collaboration of designer, developer, copywriter, and IA (Information Architecture) is critical to executing all the copy in the right place.

6. Testing & Launch

Testing the application is the most important aspect of the web app development model. That's because there are hundreds of things that can go wrong even after the application has been executed correctly. Even after double testing everything, it is better to launch your web application initially in the beta version.

7. Application Maintenance

A web application needs routine checkups and enhancements. With the passage of time, you will want have new features, and launch the next App Version. This is why your app development agreement should talk about application maintenance, after-delivery support, and future upgrades.

EXERCISE

PART I: SHORT QUESTIONS

1. What is Tomcat?
2. Write some advantages of tomcat?
3. What is NetBeans?
4. What is WEB-INF?
5. What is web.xml or Deployment Descriptor?
6. Write names of Web Application Development Stages.

PART II: LONG QUESTIONS

1. What is tomcat? Write installation steps for tomcat.
2. Describe about NetBeans? Also write its Installation steps.
3. Write Web Application Standard Directory Structure?
4. Write stages of Web Application Development.

MULTIPLE CHOICE QUESTIONS

1. Tomcat is a ______

a. User application b. Software application

c. Web Server d. Graphics application

2. Which of the following is not a JAVA Development IDE

a. NetBeans b. Eclipse c. Turbo C d. Visual Studio

3. The root directory of your web application can have any

a. Name b. Type c. format d. none

4. Which of the following is not accessible from a browser or user.

a. root directory b. WEB-INF c. index.html d. none

5. WAR stands for

a. Web Application Art b. Web Application Archive

c. Web Address Architecture d. None

6. the development team gets a clear understanding of your app objectives in the :-

a. Planning and Blueprint stage

b. Requirements Review & Proposal stage

c. Testing & Launch stage

d. Web Application Programming stage

7. ________ shows the relationship between different web pages and help understand inner structure of your website.

a. Sitemap b. plan c. Code d. Table

8. Working with color schemes, graphics, icons, templates, user experience, style guides, transitions, buttons, and much more is done in:

a. Testing & Launch stage b. Web Application Design stage

c. Application Maintenance stage d. Copywrite and labeling stage

MCQ's ANSWERS:

1	2	3	4	5	6	7	8
C	C	A	B	B	B	A	B

References

how-to-download-and-install-eclipse-on-windows. (n.d.). Retrieved from www.geeksforgeeks.org: https://www.geeksforgeeks.org/how-to-download-and-install-eclipse-on-windows/

how-to-install-netbeans-java-ide-on-windows. (n.d.). Retrieved from www.geeksforgeeks.org: https://www.geeksforgeeks.org/how-to-install-netbeans-java-ide-on-windows/

Web Application Directory Structure. (n.d.). Retrieved from www.oreilly.com: https://www.oreilly.com/library/view/programming-jakarta-struts/0596003285/ch04s03.html

web-app-development-process-model. (n.d.). Retrieved from www.techosquare.com: https://www.techosquare.com/blog/web-app-development-process-model

web-app-structure. (n.d.). Retrieved from www.wideskills.com: https://www.wideskills.com/servlets/web-app-structure

what-is-tomcat. (n.d.). Retrieved from /www.javatpoint.com: https://www.javatpoint.com/what-is-tomcat

Chapter No. 03

HTML and CSS

At the end of this chapter students will be able to:

3.1- Describe Html Document Structure and Tags
3.2- Describe CSS Styles
3.3- Designing Tables
3.4- Designing Forms
3.5- Advance Page Layout

3.1 HTML DOCUMENT STRUCTURE AND TAGS

3.1.1 Html Document Structure:

HTML stands for Hypertext Markup Language, and it is the most widely used language to write Web Pages. HTML is the standard markup language for creating Web pages. It describes the structure of a Web page. It consists of a series of elements. These elements tell the browser how to display the content. Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext. HTML is a Markup Language which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.
Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, etc.
An HTML document has two main parts: the head and the body. But firstly every HTML document should start by declaring that it is an HTML document. All HTML documents must start with a document type declaration: <!DOCTYPEhtml>.
The HTML document itself begins with <html> and ends with </html>.
The visible part of the HTML document is between <body> and </body>.
Head of an HTML Document:
The head element contains Meta information about the HTML page. The HEAD of an HTML document is where information (which might be ignored

by some Web browsers), such as the document's title, can be placed. The HEAD of a document should be declared using the following HTML tags:
<head> Should appear after the <html> definition. </head> should appear at the end of head.
A typical entry in the HEAD section of a document would be:
<title> Title of the document </title>. The title appears on the title bar of your web browser.

Body of an HTML Document

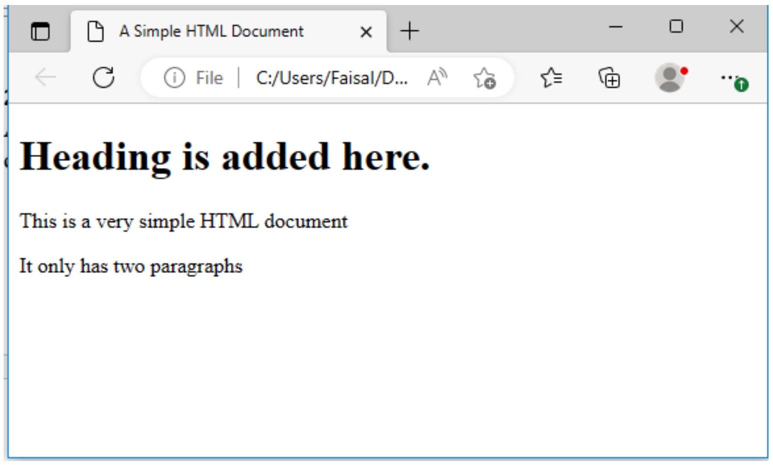
The BODY of an HTML document is where all the information you wish to view in your web browser. The text must be carefully marked-up, paragraphs must begin with the <p> tag and the end of each paragraph must be clearly marked using the HTML tag </p>. If the body text doesn't contain paragraph breaks then the text will be viewed as one long paragraph! The BODY of a document should be declared using the following HTML tags:
<body> tag should appear after the </head> tag. </body> tag should appear after the document's text but before the </html> tag.

A Simple HTML Document (Example):

Write the following text in your html editor (notepad, NetBeans or any other editor). You can use both lowercase and uppercase, as HTML is not case sensitive.

```
<!DOCTYPE html>
<html>
<head>
<title> A Simple HTML Document </title>
</head>
<body>
<h1>Heading is added here</h1>
<p>This is a very simple HTML document</p>
<p>It only has two paragraphs</p>
</body>
</html>
```

Save this file as "**first.html**" here first is the name of your html document and html is the extension that tells that the file you saved is an html file. The page will look like as given in the picture below:



Heading is added here.

This is a very simple HTML document

It only has two paragraphs

Figure 1: My First HTML Document

3.1.2 HTML TAGS

A tag is an instruction to tell the browser how it should display certain content on the screen. Sometimes developers might refer to tags as elements. Each tag element is defined by a starting tag, some content, and an ending tag. For example:

<h1>My First Heading</h1>

Here <h1> is the starting tag used for heading, My First Heading is the content for this tag element and </h1> is the ending tag which tells the tag is ended here.

Each tag is enclosed in angle brackets **(<>)** and the ending tag is same just like starting tag but has a forward slash **(/)** before the tag name.

Some tags in html do not need any closing or ending tag, like <hr> tag which is used to add a horizontal line to separate the sections in your document. Some important tags are discussed in this topic.

Basic HTML Tags: Some basic html tags are:-

Tag	Description	Example
<!DOCTYPE>	Defines the document type	<!DOCTYPE html>
<html>	Defines an HTML document	<html>….</html> All the contents for a specific html document comes inside these two tags.
<head>	Contains title, metadata/information for the document	<head> <title>My First Page</title> </head>
<title>	Defines a title for the document	<title>My First Page</title> It comes inside the head section.
<body>	Defines the document's body	<body>…</body> Each and every thing visible in a browser is added inside body section. e.g. <body> <h1>First Heading</h1> <p>First Paragraph</p> </body>
<h1> to <h6>	Defines HTML headings. There are six level of headings available h1 to h6. While h1 is largest and h6 is smallest. We use headings to clearly label different sections.	<h1>First heading</h1> <h2>Second heading</h2> . . . <h6>Sixth heading</h6>
<p>	Defines a paragraph in an html document	<p>This is a paragraph in an html document</p>
<br>	Inserts a single line break. This tag do not requires an ending tag.	<p>This is a paragraph<br> in an html document</p> The text after <br> will appear in next line

<u><hr></u>	Defines a thematic change in the content. It shows a horizontal line separating sections.	<p>This is a very simple HTML document</p><hr> <p>It only has tree paragraphs</p> There will be a horizontal line between both paragraphs.
<u><!--...--></u>	Defines a comment. As usual comments are not shown in the browser window. As they provide extra information to the person who creates or manages the page.	<!-- Here is where the project description begins -->

Forms and Input Tags: Some important form tags are:-

Tag	Description	Example
<u><form></u>	Defines an HTML form for user input	<form>…</form> All the elements of a form comes inside these two tags
<u><input></u>	Defines an input control. It is used to add a text, button or password field for inputting	<form> <label>Name: <input type="text"></label></input> </form> This will add a textbox to input text.
<u><textarea></u>	Defines a multiline input control (text area)	<label>Address: </label><textareacolspan="4" rowspan="4"></textarea>
<u><button></u>	Defines a clickable button. You can put tags like <i>,<b>,
,<img>etc in <button> tag which was not possible while creating a button using <input> tag	<button type=”button”>submit</button>

<select>	Defines a drop-down list	<label>Religion</label> <select><option>Islam</option> <option>Cristian</option> <option>Other</option> </select>
<option>	Defines an option in a drop-down list	<label>Religion</label> <select><option>Islam</option> <option>Cristian</option> <option>Other</option> </select>
<label>	Defines a label for an <input> element	<label>Name: <input type="text"></label></input> </form> This will give a label to your input text.

Images: img tag is used for image in an html page

Tag	**Description**	**Example**
<img>	Defines an image	<imgsrc="E:\CIT\CIT-303 TextBook\pic.jpg" alt="Web Development" widht="200" height="200">

Link Tags: Some important tags are:-

Tag	**Description**	**Example**
<a>	Defines a hyperlink. That is to link your document to an external file, web page or a content on the same page (Bookmark).	<a href="http://www.tevta.gop.pk" target="_blank">TEVTA</a>
<link>	Defines the relationship between a document and an external resource (mostly used to link to an external style sheets)	<link rel="stylesheet" href="E:\CIT\Web Development with Java\Project\mystyle.css">

List Tags: To add list of values both ordered and unordered lists we use list tags: Few tags are:-

Tag	Description	Example
<ul>	Defines an unordered list	<ul style="list-style-type:circle; padding-left:0px"><li>Islamabad</li><li>Karachi</li><li>Lahore</li><li>Peshawar</li><li>Quetta</li></ul>
<ol>	Defines an ordered list	<ol style="list-style-type:lower-roman"><li>Islamabad</li><li>Karachi</li><li>Lahore</li><li>Peshawar</li><li>Quetta</li></ol>
<li>	Defines a list item	"><li>Islamabad</li>

Table Tags: Some Tags used in table creation are:-

Tag	Description	Example
<table>	Defines a table	<table>…</table> All the tags relevant to table comes inside these two tags
<th>	Defines a header cell in a table	<tr> <th>Sr#</th> <th>Degree/Certificate</th> <th>Marks obtained</th> <th>University/Board</th> </tr>
<tr>	Defines a row in a table	For example we may add a new row for above header row as: <tr><td>1</td><td>Matric</td><td>672/850</td><td>BISE BWP</td></tr>

<td>	Defines a cell in a table	For example each data cell is defined as: <tr><td>1</td><td>Matric</td><td>672/850</td><td>BISE BWP</td></tr>

Styles Tags: To apply styles to a web page we use the tags:

Tag	Description	Example
<style>	Defines style information for a document	<h1 style="color:blue">Heading is added here</h1>
<div>	Defines a section in a document. <div> is a block-level element	<div> <h1 style="color:blue">Heading is added here</h1> <p>This is a paragraph in an html document</p> <p><I>This is a very simple HTML document</i></p> </div>
<header>	Defines a header for a document or section	<header> <p>Web Development with Java</p> <p>CIT Third Year</p><hr> </header>
<footer>	Defines a footer for a document or section	<footer> <hr><p>Developed by: TEVTA</p> <p><a href="https://www.tevta.gop.pk">www.tevta.gop.pk</a></p> </footer>

Meta Info:

Tag	Description	Example
<head>	Defines information about the document. The contents inside head	<head> <link rel="stylesheet" href="E:\CIT\Web Development with

	section are not visible in the document. It may content title and other relevant information about your web document.	Java\Project\mystyle.css"> <title> A Simple HTML Document </title> </head>
<meta>	Defines metadata about an HTML document. Metadata is data about data. <meta> tags comes inside head section	<meta name="description" content= "Web Development Basics"> <meta name="keywords" content=" HTML, CSS, JavaScript"> <meta name="Developed by" content="TEVTA">

Attributes:

An attribute indicates the characteristics of an object. For example, size and color of a line or text are the attributes. Attributes are specified with opening tag to provide additional information about an element or object of the document.

Example:

<h1 align=”center”>Web Programming </h1>

<hr size=50 color=red>

Some important attributes are: background-color, Background, Text, Link, Vlink, Alink, href, src, width, alt, style, etc.

3.2 DESCRIBE CSS STYLES

CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once.

CSS can be added to HTML elements in 3 ways:

Inline - by using the style attribute in HTML elements

Internal - by using a <style> element in the <head> section

External - by using an external CSS file

Benefits of using CSS:

A style sheet gives the user several benefits:

- better control over layout
- better control over text display

- separate form from structure
- better site maintainability
- smaller Web pages which means faster downloads.

Example:

```
body {
background-color: lightblue;
}
h1 {
	color: white;
	text-align: center;
}
p {
	font-family: verdana;
	font-size: 20px;
}
```

3.2.1 Inline CSS

An inline CSS is used to apply a unique style to a single HTML element. An inline CSS uses the style attribute of an HTML element. This example sets the text color of the <h1> element to blue:

Example:

```
<h1 style="color:blue;">This is a Blue Heading</h1>
```

3.2.2 Internal CSS

An internal CSS is used to define a style for a single HTML page. An internal CSS is defined in the <head> section of an HTML page, within a <style> element:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
background-color: powderblue;
}
h1 {
```

```
color: blue;
}
p{
color: red;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

3.2.3 External CSS

An external style sheet is used to define the style for many HTML pages. With an external style sheet, you can change the look of an entire web site, by changing one file. To use an external style sheet, add it as a link to in the <head> section of the HTML page:

Example:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="styles.css">
</head><body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

An external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a **.css** extension.

Here is how the "**styles.css**" looks:

```
body {
background-color: powderblue;
}
h1 {
color: blue;
}
```

```
p {
color: red;
}
```

CSS Fonts: The CSS color property defines the text color to be used. The CSS font-family property defines the font to be used. The CSS font-size property defines the text size to be used.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
color: blue;
font-family: verdana;
font-size: 300%;
}
p{
color: red;
font-family: courier;
font-size: 160%;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

CSS Border: The CSS border property defines a border around an HTML element:

Example

```
p {
border: 1px solid powderblue;
}
```

CSS Padding: The CSS padding property defines a padding (space) between the text and the border:

Example

```
p {
border: 1px solid powderblue;
padding: 30px;
}
```

External References :

External style sheets can be referenced with a full URL or with a path relative to the current web page. This example uses a full URL to link to a style sheet:

Example

```
<link rel="stylesheet" href="https://www.w3schools.com/html/styles. css">
```

Why we use CSS?

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes. The style definitions are normally saved in external .css files. With an external stylesheet file, you can change the look of an entire website by changing just one file.

3.3 DESCRIBE TABLES

Defining an HTML Table: The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells. An HTML table is defined with the <table> tag.

Each table row is defined with the <tr> tag. A table header is defined with the <th> tag. By default, table headings are bold and centered. A table data/cell is defined with the <td> tag.

Cellpadding and Cellspacing Attributes: There are two attributes called cellpadding and cellspacing which you will use to adjust the white space in your table cells. The cellspacing attribute defines space between table cells, while cellpadding represents the distance between cell borders and the content within a cell. <table border=”1”, cellpadding=”5” cellspacing=”5”>.

Colspan and Rowspan Attributes: You will use colspan attribute if you want to merge two or more columns into a single column. Similar way you will use rowspan if you want to merge two or more rows.

Some important Table Tags are:

<u><table></u>:Defines a table,e.g. <table>…</table> All the tags relevant to table comes inside these two tags

<u><caption></u>:Defines a table caption,e.g.

```
<table><caption>Student Information</caption>
 </table>
```

<u><th></u>: Defines a header cell in a table

```
<tr>
<th>Sr#</th>
<th>Degree/Certificate</th>
<th>Marks obtained</th>
<th>University/Board</th>
 </tr>
```

<u><tr></u>: Defines a row in a table. For example we may add a new row for above header row as:

```
 <tr><td>1</td><td>Matric</td><td>672/850</td><td>BISE BWP</td></tr>
```

<u><td></u>: Defines a cell in a table For example each data cell is defined as:

```
 <tr><td>1</td><td>Matric</td><td>672/850</td><td>BISE BWP</td></tr>
```

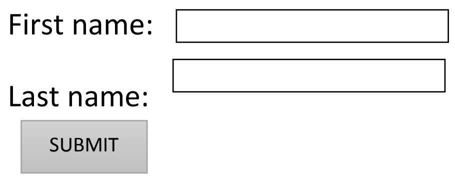
Simple Tabel Example:

```
<table boarder=2 cellspacing="5” cellpadding="5">
        <tr>
                <th>    Sr#      </th>
                <th>Degree/ Certificate</th>
                <th>Marks obtained</th>
                <th>University/Board</th>
        </tr>
        <tr>
                <td>101</td>
                <td>Matriculation</td>
                <td>950</td>
                <td>BISE Lahore</td>
        </tr>
        <tr>
                <td>102</td>
                <td>DAE</td>
                <td>990</td>
                <td>PBTE Lahore</td>
        </tr>
</table>
```

3.4 Designing Forms

HTML Forms are required, when you want to collect some data from the site visitor. For example, during user registration you would like to collect information such as name, email address, credit card, etc.

There are various form elements available like text fields, textarea fields, dropdown menus, radio buttons, checkboxes, etc. Form Example:



The <form> Element: The HTML <form> element defines a form that is used to collect user input:
<form> … </form>
An HTML form contains form elements.
Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.
The <input> Element:
The <input> element is the most important form element.
The <input> element can be displayed in several ways, depending on the type attribute.
Here are some examples:

Type	**Description**
<input type="text">	Defines a one-line text input field

<input type="radio">	Defines a radio button (for selecting one of many choices)
<input type="submit">	Defines a submit button (for submitting the form)

Text Input: <input type="text"> defines a one-line input field for text input:
Example

```
<form>
First name:<input type="text" name="firstname"><br>
Last name:<input type="text" name="lastname"><br>
</form>
```

Radio Button Input: <input type="radio"> defines a radio button. Radio buttons let a user select ONE of a limited number of choices:
Example

```
<form>
<input type="radio" name="gender" value="male" checked> Male<br><input
type="radio" name="gender" value="female"> Female<br>
<input type="radio" name="gender" value="other">Other<br>
</form>
```

The Submit Button:
<input type="submit"> defines a button for submitting the form data to a form-handler. The form-handler is typically a server page with a script for processing input data. **Example**

```
<form action="/action_page.php">
First name:<input type="text" name="firstname" value="ahmad"><br>
Last name:<input type="text" name="lastname" value="ali"><br>
<input type="submit" value="Submit">
</form>
```

The Action Attribute: The action attribute defines the action to be performed when the form is submitted. E.g. <form action="/action_page.php">
The Target Attribute: The target attribute specifies if the submitted result will open in a new browser tab, a frame, or in the current window.
e.g. <form action="/action_page.php" target="_blank">
The Method Attribute: The method attribute specifies the HTTP method (GET or POST) to be used when submitting the form data: e.g.
<form action="/action_page.php" method="get">

3.5 Advance Page Layout

Page layout is the part of graphic design that deals with the arrangement of visual elements on a page. Page layout is used to make the web pages look better. It establishes the overall appearance, relative importance, and relationships between the graphic elements to achieve a smooth flow of information and eye movement for maximum effectiveness or impact.

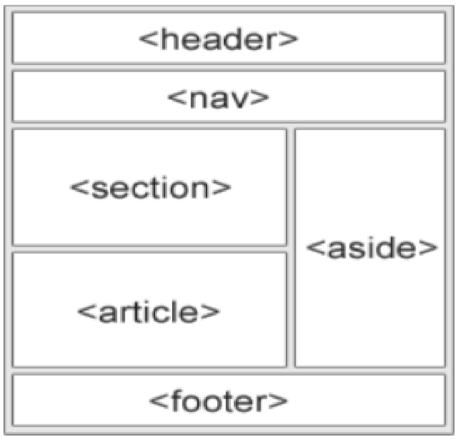


Figure 4: HTML PAGE LAYOUT

Page Layout Information:

Header: The part of a front end which is used at the top of the page. <header> tag is used to add header section in web pages.

Navigation bar: The navigation bar is same as menu list. It is used to display the content information using hyperlink.

Index / Sidebar: It holds additional information or advertisements and is not always necessary to be added into the page.

Content Section: The content section is the main part where content is displayed.

Footer: The footer section contains the contact information and other query related to web pages. The footer section always put on the bottom of the web pages. The <footer> tag is used to set the footer in web pages.

Example:

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Page Layout</title>
<style>
  .head1 {
font-size:40px;
color:#009900;
font-weight:bold;
  }
  .head2 {
font-size:17px;
margin-left:10px;
margin-bottom:15px;
  }
body {
margin: 0 auto;
background-position:center;
background-size: contain;
  }
  .menu {
position: sticky;
top: 0;
background-color: #009900;
padding:10px 0px 10px 0px;
color:white;
margin: 0 auto;
overflow: hidden;
  }
  .menu a {
float: left;
color: white;
text-align: center;
padding: 14px 16px;
text-decoration: none;
font-size: 20px;
  }
  .menu-log {
right: auto;
float: right;
  }
footer {
width: 100%;
bottom: 0px;
background-color: #000;
color: #fff;
position: absolute;
padding-top:20px;
padding-bottom:50px;
text-align:center;
font-size:30px;
font-weight:bold;
  }
  .body_sec {
margin-left:20px;
  }
</style>
</head>
<body>
<!-- Header Section -->
<header>
<div class="head1">Computer Science</div>
<div class="head2">A computer science web Page</div>
</header>
<!-- Menu Navigation Bar -->
<nav>
<div class="menu">
<a href="#home">HOME</a>
<a href="#news">NEWS</a>
<a href="#notification">NOTIFICATIONS</a>
<div class="menu-log">
<a href="#login">LOGIN</a>
</div>
</div>
</nav>
```

```
<!-- Body section -->
<div class = "body_sec">
<section id="Content">
<h3>Content section</h3>
</section>
</div>
<!-- Footer Section -->
<footer>Footer Section</footer>
</body>
</html>
```

EXERCISE

PART I: SHORT QUESTIONS

13. What is html tag and give one example?
14. Write any five html tags?
15. What is CSS?
16. What is an Internal CSS?
17. Explain external CSS
18. Why we use CSS?
19. Describe cellpadding and cellspacing attributes.
20. How can you add a header in an html page?

PART II: LONG QUESTIONS

6. Briefly describe HTML Document Structure.
7. Brieflydescribe how to design a form in an html page.
8. Brieflydescribe CSS styles.

MULTIPLE CHOICE QUESTIONS

1. HTML Document has two main parts:

a. Header and Footer b. Head and Body
c. Title and Content d. Head and Content

2. Each tag is enclosed in

a. {} curly Brackets b. [] Square Bracket
c. () Parenthesis d. <> Angle Brackets

3. In the ending tag a ______ comes along with the tag name.

a. Back Slash (\ ) b. Forward Slash (/)
c. (-) Hyphen d. none

4. Which of the following is a comment tag.

a. /*…*/ b. <!--…--> c. <…> d. //

5. A/An ______ style is used to apply a unique style to a single HTML element.

a. Inline b. Internal c. External d. None

6. An external stylesheet is saved in a separate file with extension.

a. **.html** b. **.css** c. **.exe** d. **.java**

7. We use ____ tag for Hyperlink.

a. <a> b. <link> c. <div> d. <nav>

8. To insert a header row in a table we use:

a. <th> tag b. <tr> tag c. <header> tag d. None

9. The tag____ has no ending Tag:

a. <p> tag b. <hr> tag c. <nav> tag d. none

10. The distance between cell boarder and cell content is st by ______ attribute.

a. cellspacing b. rowspan c. cellpadding d. colspan

1	2	3	4	5	6	7	8	9	10
B	D	B	B	A	B	A	A	B	C

Reference:

HTML Tutorial. (n.d.). Retrieved from https://www.w3schools.com: https://www.w3schools.com/html/html_intro.asp

Chapter. No. 04

JAVASCRIPT

4.1 Introduction to Javascript
4.2 Hiding and Showing Elements
4.3 Styling Elements
4.4 Using Jquery
4.5 Jquery Selectors
4.6 Validating Forms

4.1 What is JavScript?

JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform.

- JavaScript is the Programming Language for the Web.
- JavaScript can update and change both HTML and CSS.
- JavaScript can calculate, manipulate and validate data.

What can JavaScript Do?

1. JavaScript Can Change HTML Content
2. JavaScript Can Change HTML Attribute Values
3. JavaScript Can Change HTML Styles (CSS)
4. JavaScript Can Hide HTML Elements
5. JavaScript Can Show HTML Elements

Hello World using JavaScript:

Just to give you a little excitement about JavaScript programming, I'm going to give you a small conventional JavaScript Hello World program, You can try it using Demo link

```
<html>
<body>
<script language = "javascript" type = "text/javascript">
```

```
<!--
document.write("Hello World!")
//-->
</script>
</body>
</html>
```

4.1.1 JavaScript Can Change HTML Content

One of many JavaScript HTML methods is getElementById().

This example uses the method to "find" an HTML element (with id="demo") and changes the element content (innerHTML) to "Hello JavaScript":

Example

```
document.getElementById("demo").innerHTML = "Hello JavaScript";
```

4.1.2 JavaScript Can Hide HTML Elements

Hiding HTML elements can be done by changing the display style:

Example

```
document.getElementById("demo").style.display="none";
or
document.getElementById('demo').style.display='none';
```

4.1.3 JavaScript Can Show HTML Elements

Showing hidden HTML elements can also be done by changing the display style

Example:

```
document.getElementById("demo").style.display="block";
or
document.getElementById('demo').style.display='block';
```

4.1.4 JavaScript Can Change HTML Styles (CSS)

Changing the style of an HTML element, is a variant of changing an HTML attribute:

Example:

```
document.getElementById("demo").style.fontSize="35px";
or
document.getElementById('demo').style.fontSize='35px';
```

JavaScript Functions?

A JavaScript function is a block of code designed to perform a particular task. A JavaScript function is executed when "something" invokes it (calls it)

Example:

```
Function myFunction(p1,p2){
   return p1 * p2;             // The function returns the product of p1andp2
}
```

4.3 Styling Elements

The HTML DOM allows JavaScript to change the style of HTML elements.

To change the style of an HTML element, use this syntax:

document.getElementById(*id*).style.*property* = *new style*

The following example changes the style of a <p> element:

```
<html>
<body>

<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color = "blue";
</script>

</body>
</html>
```

Using Events

The HTML DOM allows you to execute code when an event occurs. Events are generated by the browser when "things happen" to HTML elements:

- An element is clicked on
- The page has loaded
- Input fields are changed

This example changes the style of the HTML element with id="id1", when the user clicks a button

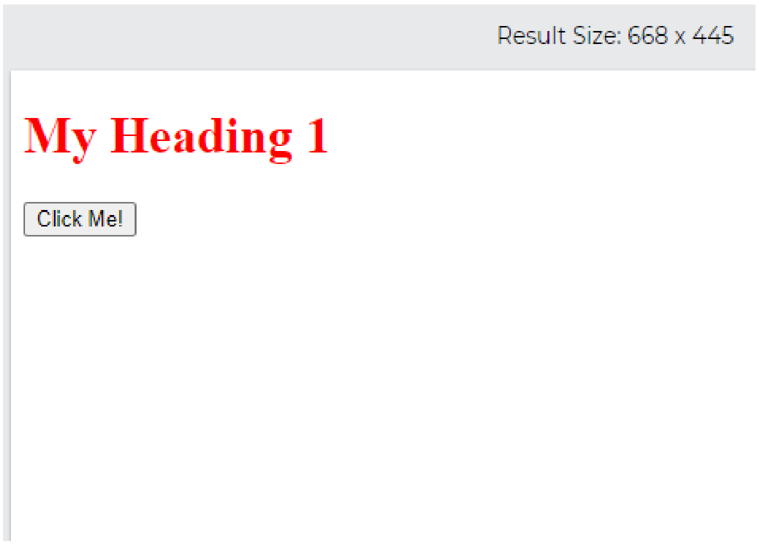
Example:

```
<!DOCTYPE html>
<html>
<body>

<h1 id="id1">My Heading 1</h1>

<button type="button"
onclick="document.getElementById('id1').style.color = 'red'">
Click Me!</button>

</body>
</html>
```


Result Size: 668 x 445

My Heading 1

Figure 5: Styling Elements

4.4 Using Jquery

jQuery is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.
The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

4.4.1 jQuery Syntax

The jQuery syntax is tailor-made for selecting HTML elements and performing some action on the element(s).
Basic syntax is**: $(selector).action()**

- A $ sign to define/access jQuery
- A (selector) to "query (or find)" HTML elements
- A jQuery action() to be performed on the element(s)

Examples:

$(this).hide() - hides the current element.
$("p").hide() - hides all <p> elements.
$(".test").hide() - hides all elements with class="test".
$("#test").hide() - hides the element with id="test".

4.4.2 jQuery Hide and Show

With jQuery, you can hide and show HTML elements with the hide() and show() methods:

```
$("#hide").click(function(){
$("p").hide();
});
```

```
$("#show").click(function(){
$("p").show();
});
```

4.5 jQuery Selectors

jQuery selectors are one of the most important parts of the jQuery library. jQuery selectors allow you to select and manipulate HTML element(s).jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors. All selectors in jQuery start with the dollar sign and parentheses: $().

4.5.1 The element Selector

The jQuery element selector selects elements based on the element name.

You can select all <p> elements on a page like this:
$("p")

Example

When a user clicks on a button, all <p> elements will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
}):
```

4.5.2 The #id Selector

The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.

An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element. To find an element with a specific id, write a hash character, followed by the id of the HTML element:

$("#test")

Example

When a user clicks on a button, the element with id="test" will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
```

4.5.3 The .class Selector

The jQuery *.class* selector finds elements with a specific class.
To find elements with a specific class, write a period character, followed by the name of the class:
$(".test")

Example
When a user clicks on a button, the elements with class="test" will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
    $(".test").hide();
  });
});
```

4.6 Apply validation on HTML form

Forms are used in webpages for the user to enter their required details that are further send it to the server for processing. A form is also known as web form or HTML form

```
<html>

<body>
```

```
<h2>JavaScript Validation</h2>

<p>Please input a number between 1 and 10: </p>

<input id="numb">

<button type="button" onclick="myFunction()">Submit</button>

<p id="demo"></p>

<script>

functionmyFunction() {

  // Get the value of the input field with id="numb"

let x = document.getElementById("numb").value;

  // If x is Not a Number or less than one or greater than 10
let text;

if (isNaN(x) || x < 1 || x > 10)
{
text = "Input not valid";
  }
else {
text = "Input OK";
  }
document.getElementById("demo").innerHTML = text;
}
</script>

</body>
</html>
```

EXERCISE

PART I: SHORT QUESTIONS

1. What is JavaScript?
2. What can JavaScript Do?
3. Write aexample of Hide HTML Element through javaScript?
4. Explain JavaScript Functions?
5. Write the syntax of JavaScript Functions.
6. Define jQuery Selector?
7. Define the #id Selector?
8. Define the .class Selector?
9. Write any two Examples of jQuery Selectors with Description?
10. Explain jQuery Hide and Show?
11. How to create sliding effects on HTML elements with   jQuery?

PART II: LONG QUESTIONS

1- What is JavaScript? Give the example of simple JavaScript Hello World program .

2- Give detail description of Styling Elements in javaScript. Also Add the Examples

3- Write a note on jqQuery selectors.

4- Write the code to apply validation on HTML form using javaScript

MULTIPLE CHOICE QUESTIONS

1. JavaScript is designed for creating
   a) network-centric applications
   b) client-centric applications
   c) server-centric applications
   d) none

2. __________can **calculate**, **manipulate** and **validate** data
   a) JavaScript
   b) JSp
   c) server
   d) None

3. HTML Elements Can be Hide and show through_______
   a) JavaScript
   b) JSp
   c) server
   d) None

4. Which of the following method syntax is correct __________
   a) getElementsById()
   b) getElementById()
   c) gotElementByid()
   d) All of Above

5. jQuery is a _____________library
   a) JavaScript
   b) HTML
   c) CSS
   d) None

6. jQuerywrapsmany lines of JavaScript code called ____________
   a) veriable
   b) Constructer
   c) Method
   d) All of Above

7. Which of the following correct jQuery syntax ___________
   a) _(selector).action
   b) ()$(selector).action()
   c)@(selector).action()
   d) None

8. $(".test").hide() will hide _______

a) the current element.

b) all elements with class="test”

c) Both

d)None

9. Selectors in ______start with the dollar sign and parentheses: $().

a) JavaScript

b) HTML

c) CSS

d) jQuery

10. Selector uses to find the specific element.

a) .class

b) $ id p

c)#*id* selector

d)None

11.Selector uses to find the specific elements

a) *.class* selector

b) $ id p

c) #*id* selector

d)None

12.Selects the current HTML element

a) $(.this)

b) $(current)

c) $(this)

d) All of Above

Answers

1	2	3	4	5	6	7	8	9	10
A	A	A	B	A	C	B	B	D	C
11	12								
A	C								

REFRENCES

1. https://www.tutorialspoint.com/javascript/index.html
2. https://www.w3schools.com/jquery/jquery_syntax.asp

Chepter No. 5

JAVA SERVLETS

5.1 What is a Servlet?
5.2 Advantages of Servlet
5.3 Servlet Types
5.4 Writing Basic Hello World Servlet
5.5 Servlet Life Cycle
5.6 Servlet and Forms
5.7 Servlet and Input Validation

5.1 What is a Servlet?

1. Servlet can be described in many ways, depending on the context.
2. Servlet is a technology which is used to create a web application.
3. Servlet is an API that provides many interfaces and classes including documentation.
4. Servlet is an interface that must be implemented for creating any Servlet.
5. Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
6. Servlet is a web component that is deployed on the server to create a dynamic web page.

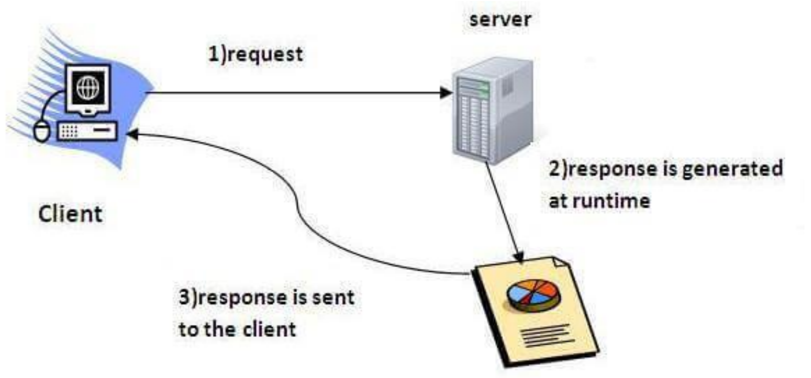

server
1)request
2)response is generated at runtime
Client
3)response is sent to the client

Figure.1: Servlet Request Response

5.2 Advantages of Servlet

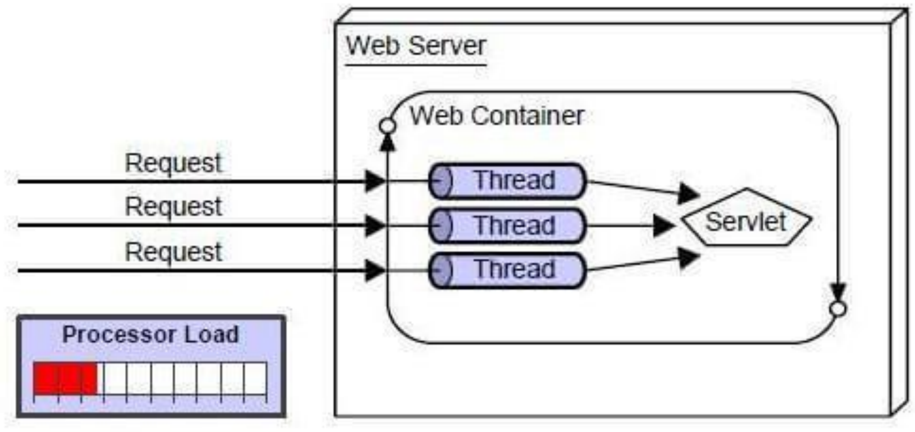


Figure 2: Servlet Request

There are many advantages of Servlet over CGI. The web container creates threads for handling the multiple requests to the Servlet. Threads have many benefits over the processes such as they share a common memory area, lightweight, cost of communication between the threads are low. The advantages of Servlet are as follows:

1. **Better performance:** because it creates a thread for each request, not process.
2. **Portability:** because it uses Java language.
3. **Robust:** JVM manages Servlets, so we don't need to worry about the memory leak, garbage collection, etc.
4. **Secure:** because it uses java language.

5.3 Servlet Types

There are two main servlet types, generic and HTTP:

5.3.1 Generic servlets

A generic servlet is a protocol independent Servlet that should always override the service() method to handle the client request. The service() method accepts two arguments ServletRequest object and ServletResponse object. The request object tells the servlet about the request made by client while the response object is used to return a response back to the client.

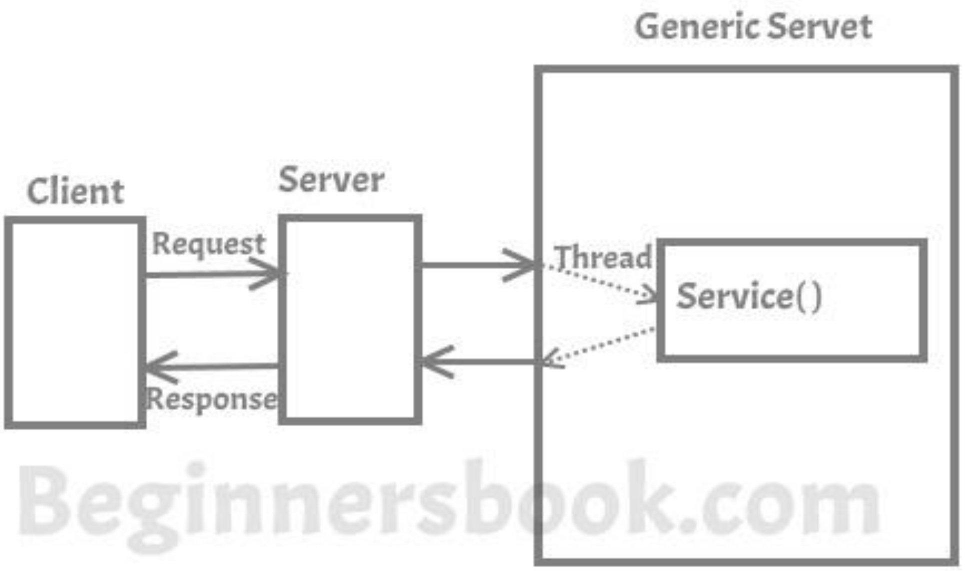


Figure 3: Generic Servlet

5.3.2 HTTP servlets:

Unlike Generic Servlet, the HTTP Servlet doesn’t override the service() method. Instead it overrides the doGet() method or doPost() method or both. The doGet() method is used for getting the information from server while the doPost() method is used for sending information to the server.

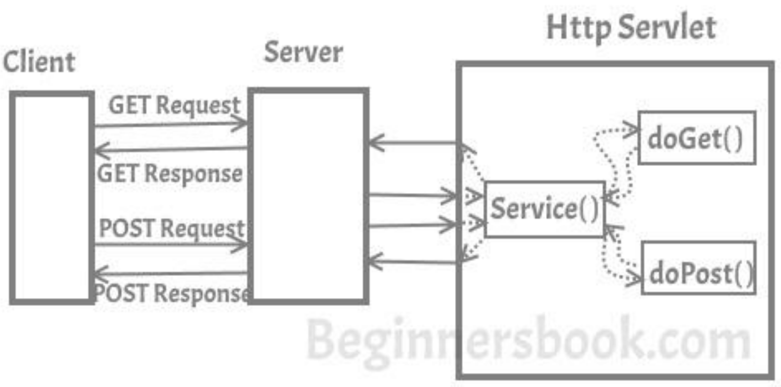


Figure 4: Http servlet

5.4 Writing Basic Hello World Servlet

Write a Servlet to display “Hello World” on browser.
WEB-INF/web.xml

```
<web-app>
<servlet>
<servlet-name>s10</servlet-name>
<servlet-class>Hello</servlet-class>
</servlet>
<!-- Servlet Mapping -->
<servlet-mapping>
<servlet-name>s10</servlet-name>
<url-pattern>/hello</url-pattern>
</servlet-mapping>
</web-app>
```

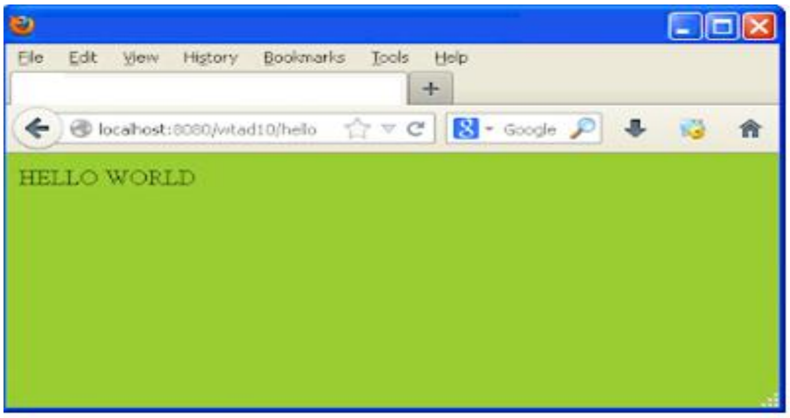
WEB-INF/classes/Hello.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Hello extends HttpServlet
{
 public void service( HttpServletRequest request, HttpServletResponse
response ) throws ServletException, IOException
 {
response.setContentType("text/html");
PrintWriter out = response.getWriter();

out.println("<HTML>" +
    "<HEAD><TITLE>WTAD.practical-10 @iAmLearningHere</TITLE></HEAD>"
+
    "<body bgcolor=YellowGreen>" +
    "HELLO WORLD" +
    "</body></HTML>");
 }
}
```

Preview



HELLO WORLD

How to Run?
> Start Apache Server
> Start localhost and open program folder
> type url '/hello'

5.5 Servlet Life Cycle

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

- The servlet is initialized by calling the **init()** method.
- The servlet calls **service()** method to process a client's request.
- The servlet is terminated by calling the **destroy()** method.

Now let us discuss the life cycle methods in detail.

1 The init() Method

The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations, just as with the init method of applets

```
publicvoidinit()throwsServletException{
// Initialization code...
}
```

2 The service() Method

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.

```
publicvoid service(ServletRequestrequest,ServletResponse response)
throwsServletException,IOException{
}
```

3 The doGet() Method

A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

```
publicvoiddoGet(HttpServletRequestrequest,HttpServletResponse response)
throwsServletException,IOException{
// Servlet code
}
```

4 The doPost() Method

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

```
publicvoiddoPost(HttpServletRequestrequest,HttpServletResponse response)
throwsServletException,IOException{
// Servlet code
}
```

5 The destroy() Method

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections

```
publicvoid destroy(){
// Finalization code...
}
```

Architecture Diagram

The following figure depicts a typical servlet life-cycle scenario.

- First the HTTP requests coming to the server are delegated to the servlet container.

- The servlet container loads the servlet before invoking the service() method.
- Then the servlet container handles multiple requests by spawning multiple threads, each thread executing the service() method of a single instance of the servlet.

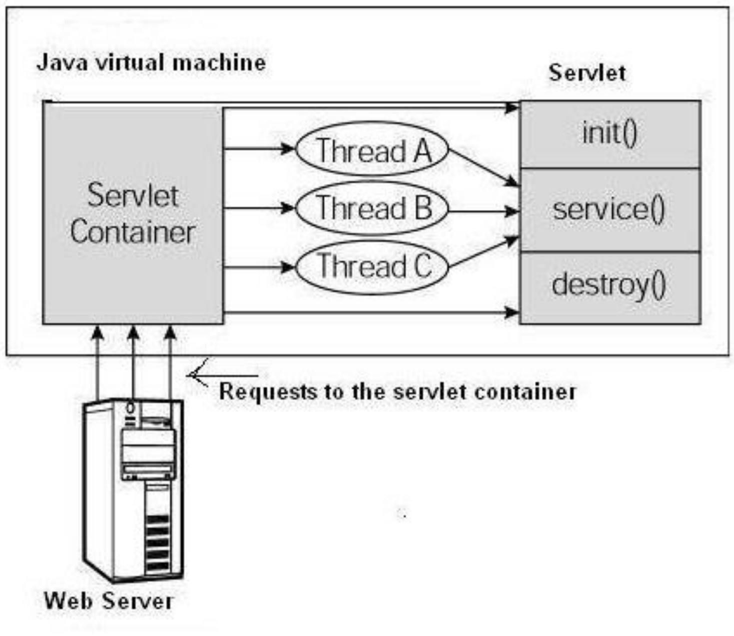


Figure 5: Architectural diagram

5.6 Servlet and Forms

A web page is the combination of many input elements such as label, text box, checkbox, options, images, etc., It can be prepared by enclosing all the input elements inside an “HTML FORM” on the server-side with java servlet. Usually, an HTML form collects data from the user via these input elements and lets us see how they will be sent to the server-side by using HTTP GET/POST methods.

Flow:

1. **Request** will be sent from HTML form
2. It will be sent as either **GET/POST** method in the Java servlet side
3. **Response** will be provided from servlet as an HTML form

Example of Registration form in servlet:

you will learn that how to create simple registration form in servlet.
create a table first as given below:

```
CREATE TABLE  "REGISTERUSER"
  (    "NAME" VARCHAR2(4000),
    "PASS" VARCHAR2(4000),
    "EMAIL" VARCHAR2(4000),
    "COUNTRY" VARCHAR2(4000)
  )
/
```

In this example, we have created the three pages.

- register.html
- Register.java
- web.xml

register.html

In this page, we have getting input from the user using text fields and combobox. The information entered by the user is forwarded to Register servlet, which is responsible to store the data into the database.

```
<html>
<body>
<form action="servlet/Register" method="post">

Name:<input type="text" name="userName"/><br/><br/>
Password:<input type="password" name="userPass"/><br/><br/>
Email Id:<input type="text" name="userEmail"/><br/><br/>
Country:
<select name="userCountry">
<option>UK</option>
<option>Pakistan</option>
<option>other</option>
</select>

<br/><br/>
<input type="submit" value="register"/>
```

```
</form>
</body>
</html>
```

Register.java

This servlet class receives all the data entered by user and stores it into the database. Here, we are performing the database logic. But you may separate it, which will be better for the web application.

```
import java.io.*;
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;

public class Register extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse respon
se)
        throws ServletException, IOException {

response.setContentType("text/html");
PrintWriter out = response.getWriter();

String n=request.getParameter("userName");
String p=request.getParameter("userPass");
String e=request.getParameter("userEmail");
String c=request.getParameter("userCountry");

try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

PreparedStatement ps=con.prepareStatement(
"insert into registeruser values(?,?,?,?)");

ps.setString(1,n);
ps.setString(2,p);
ps.setString(3,e);
ps.setString(4,c);

int i=ps.executeUpdate();
if(i>0)
out.print("You are successfully registered...");


}catch (Exception e2) {System.out.println(e2);}

out.close();
}

}
```

web.xml file

The is the configuration file, providing information about the servlet.

```
<web-app>

<servlet>
<servlet-name>Register</servlet-name>
<servlet-class>Register</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>Register</servlet-name>
<url-pattern>/servlet/Register</url-pattern>
</servlet-mapping>

<welcome-file-list>
   <welcome-file>register.html</welcome-file>
</welcome-file-list>
</web-app>
```

5.6 Validating Forms

Form Validation in Java Servlet | Verifying the pattern and format of form data before it is getting used in the business logic as inputs are called form validations, otherwise business logic may give invalid results or exceptions by taking the inputs. Example:- Checking required fields of the form are filled or not, checking whether age types as a numeric value and e.t.c.

Form Validation Examples	**Business Logic Examples**
Credit card number is having 16 digits or not? (format verification)	Credit card number is existing or a valid number?
The given date value is in MM/DD/YYYY pattern or not? (Pattern verification)	Getting sales report for given data value.
Whether age is entered as a number or not, and it is in the range of 1 to 150?	Checking whether the person is eligible to vote or not based on the given age.

EXERCISE

PART I: SHORT QUESTIONS

1. What is a Servlet?
2. Write any two Advantages of Servlet?
3. Define a Generic servlets?
4. What is a HTTP servlets explain with diagram?
5. Enlist steps of Writing Basic Hello World Servlet?
6. What are the processes of Servlet Life Cycle?
7. Write any two methods of servlet life cycle?
8. Explain the the service() Method of servlet life cycle
9. Draw the Architecture Diagram servlet life-cycle?
10. Explain Servlet and Forms?
11. What is a Register.java?
12. What is Form Validating?

PART II: LONG QUESTIONS

1. Explain servlet with its architectural diagram. Also discuss the advantages of Servlet
2. Discuss the Types of Servlet
3. Discuss the Servlet Life Cycle in detail with its architectural diagram

MULTIPLE CHOICE QUESTIONS

1. Servlet is used to create ________________
   A) Static web page C) Dynamic web pages
   B) Both D) None
2. The web container creates_________ for handling the multiple requests to the Servlet.
   A) threads C) Process
   B) Both D) None
3. ____________Servlet that should always override the service() method to handle the client request
   A) Generic servlet C) Dynamic servlet
   B) HTTP Servlet D) None
4. The service() method accepts arguments
   A) two C) Three
   B) One D) None
5. The request Once application created next step would bewhich will handle
   A) None C)todestroyServlet
   B) to initiate Servlet D) to create Servlet
6. The servlet calls ___________ method to process a client's request.
   A) init() C) service()
   B) Destroy() D) None
7. The_________ method is the main method to perform the actual task in servlet life cycle.
   A) service() C) init()
   B) Destroy() D) None
8. The_________method is called only once at the end of the life cycle of a servlet.
   A) destroy() C) Close()
   B)both D) None
9. The servlet container loads the servlet before invoking the ______ method.

A) service() C) init()
B) Destroy() D) None

1	2	3	4	5	6	7	8	9
C	A	A	A	D	C	A	A	A

References

https://www.javatpoint.com/example-of-registration-form-in-servlet

https://www.knowprogram.com/servlet/form-validation-in-java-servlet/

Chapter No. 6

JAVA SERVER PAGES

6.1 Introduction
6.2 Life Cycle of JSP
6.3 JSP Elements
6.4 JSP Standard Actions

6.1 Introduction to JSP:

Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications. JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

Advantages of JSP pages

- They are easy to maintain.
- No recompilation or redeployment is required.
- JSP has access to entire API of JAVA.
- JSP are extended version of Servlet.

Features of JSP

- **Coding in JSP is easy**: - As it is just adding JAVA code to HTML/XML.
- **Reduction in the length of Code**: - In JSP we use action tags, custom tags etc.
- **Connection to Database is easier**:-It is easier to connect website to database and allows reading or writing data easily to the database.
- **Make Interactive websites**: - In this we can create dynamic web pages which helps user to interact in real time environment.
- **Portable, Powerful, flexible and easy to maintain**: - as these are browser and server independent.
- **No Redeployment and No Re-Compilation**:- It is dynamic, secure and platform independent so no need to re-compilation.

- **Extension to Servlet** :- as it has all features of servlets, implicit objects and custom tags

6.2 Life Cycle of JSP

Life Cycle is defined as translation of JSP Page into servlet as a JSP Page needs to be converted into servlet first in order to process the service requests. The Life Cycle starts with the creation of JSP and ends with the disintegration of that.

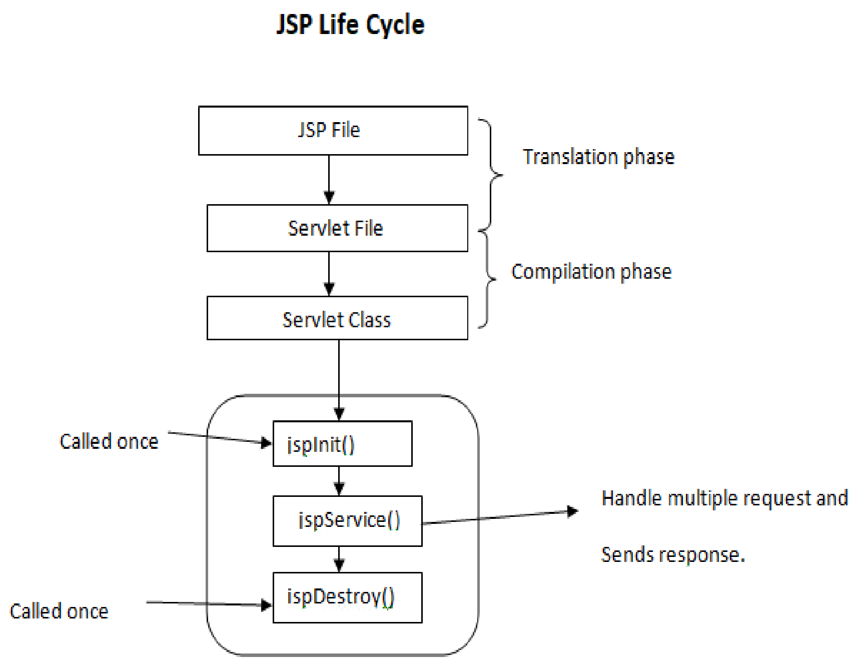


Figure 1: JSF life cycle

Following steps explain the JSP life cycle:

1. Translation of JSP page
2. Compilation of JSP page(Compilation of JSP page into _jsp.java)
3. Classloading (_jsp.java is converted to class file _jsp.class)
4. Instantiation(Object of generated servlet is created)
5. Initialisation(_jspinit() method is invoked by container)

6. Request Processing(_jspservice() method is invoked by the container)
7. Destroy (_jspDestroy() method invoked by the container)

Let us have more detailed summary on the above points:

1. **Translation of the JSP Page:** This is the first step of the JSP life cycle. This translation phase deals with the Syntactic correctness of JSP. Here test.jsp file is translated to test.java.
2. **Compilation of JSP page:** Here the generated java servlet file (test.java) is compiled to a class file (test.class).
3. **Classloading:** Servlet class which has been loaded from the JSP source is now loaded into the container.
4. **Instantiation:** Here an instance of the class is generated. The container manages one or more instances by providing responses to requests.
5. **Initialization:** jspInit() method is called only once during the life cycle immediately after the generation of Servlet instance from JSP.
6. **Request processing:** _jspService() method is used to serve the raised requests by JSP. It takes request and response objects as parameters. This method cannot be overridden.
7. **JSP Cleanup:** In order to remove the JSP from use by the container or to destroy the method for servlets jspDestroy()method is used. This method is called once, if you need to perform any cleanup task like closing open files, releasing database connections jspDestroy() can be overridden.

6.3 JSP Elements

JSP elements are called the JSP tags on JSP page. On the JSP page mainly three groups of JSP elements are used :

- **JSP Scripting Elements**
- **JSP Directive Elements**
- **JSP Standard Action Elements**

6.3.1 JSP Scripting Elements

JSP Scripting Elements are used for writing the Java Code inside the JSP page. There are different types of scripting elements these elements are used for various purposes. Following are the scripting elements :

- **JSP Scriptlet element/tag :** A scriptlet tag is denoted by the special characters <% %>. The special characters '<%' specifies the starting of the scriptlet tag and '%>' specifies end of scriptlet. On the JSP page Java Code is written inside it.
- **JSP Declaration element/tag :** The declaration tag is denoted by the special characters <%! %>. The special character '<%!' specifies the start of the declaration and '%>' specifies the end of the declaration. Declaration element is used to declare the fields, methods to use inside the JSP page. These declaration element helps developer to declare fields, methods like in Servlet how they can declare.
- **JSP Expression element/tag :** Then expression tag is denoted by the special characters <%= %>. The special character '<%= specifies the starting of the expression and '%>' specifies the end of the expression. The expression tag is used to set the output. The expression is used like the 'out' implicit object in JSP.

6.3.2 JSP Directive Elements

Directive elements in JSP provides the special information to the JSP engine. It gives the information about the JSP page. Each JSP page goes through the two phases i.e. translation phase and request time phase. At the translation phase JSP page is translated into a Servlet. The JSP engine translates the JSP page into a Servlet, directive elements are handled at the translation time. These directives are translated to the Servlet only for once, until there is no changes made to the directive elements in the Servlet.
JSP Directive Elements can be declared within the following special characters '<%@ %>'. Syntax for using directive in the JSP page is as follows :
<%@ directiveName attr1="value1" attr2="value2" %>

6.3.3 JSP Standard Action Elements

JSP Action Elements are used to take action on the basis of some information. These action elements can create, modify or use the other objects. Action elements are coded with the strict syntax they are used to apply the built-in functionality. Action elements are represented as <jsp:tagName> </jsp:tagName>. Code is written inside these tags, the <jsp:tagName> specifies the starting of the action tag where as the action tag is ends with </jsp:tagName>. When you are not required to defined the body of the tag you may end up the action tag by <jsp:tagName/>

6.4 JSP Standard Actions

The JSP specification provides a standard tag called Action tag used within JSP code and is used to remove or eliminate Scriptlet code from your JSP code as JSP Scriptlets are obsolete and are not considered nowadays. There are many JSP action tags or elements, and each of them has its own uses and characteristics. Each JSP action tag is implemented to perform some precise tasks.

The action tag is also implemented to streamline flow between pages and to employ a Java Bean. As it coincides with the XML standard, the syntax for the action element is:

Syntax:

<jsp:action_name attribute = "attribute_value" />
Here is the list of JSP Actions:

1. **jsp:forward:**is used for forwarding the request and response to other resources.
2. **jsp:include:**is used for including another resource.
3. **jsp:body:**is used for defining dynamically-defined body of XML element.
4. **jsp:useBean:**is used for creating or locating bean objects.
5. **jsp:setProperty:**is used for setting the value of property in the bean object.

6. **jsp:getProperty:**is used for printing the value of the property of the bean.
7. **jsp:element:**is used for defining XML elements dynamically.
8. **jsp:plugin:**is used for embedding other components (applets).
9. **jsp:param:** is used for setting the parameter for (forward or include) value.
10. **jsp:text:**is used for writing template text in JSP pages and documents.
11. **jsp:fallback:**is used for printing the message if plugins are working.
12. **jsp:attribute:**is used for defining attributes of dynamically-defined XML element.

Basic Attributes of Action Tags

Two basic attributes are commonly used for all action tags. These are:

1. **Id:** The id attribute defines unique action elements and allows actions to be referenced within the JSP page. When the action creates an object's instance, the id attribute is used to refer to it.
2. **Scope:** The scope attribute is used for identifying an action's life cycle. It correlates with the id attribute because the scope attribute is used to establish that particular object's lifespan associated with the ID.

EXERCISE

PART I: SHORT QUESTIONS

1. Define JSP?
2. Enlist Features of JSP
3. Define Life Cycle of JSP?
4. Enlist the steps of JSP life cycle.
5. What is Instantiation JSP Life cycle?
6. What is JSP expression tag?
7. What is JSP Declaration Tag?
8. Difference between JSP Scriptlet tag and Declaration tag
9. What is a JSP Standard Actions?
10. Write any two JSP Action Tags with Description?

PART II: LONG QUESTIONS

1. Write a note on JSP Life Cycle
2. Write a note on JSP Elements
3. Discuss JSP Standard Actions

MULTIPLE CHOICE QUESTIONS

1. Java Server Pages (JSP) is a ______________
   a) server-side programming b) client-side programming
   c) desktop app programming d) AI programming
2. The Life Cycle starts with the_____________ of JSP
   a) creation b) initialization
   c) declaration d)none of these
3. Fourth steps of life cycle of JSP
   a)Instantiation b)Classloading
   c) Request Processing d) Translation
4. method will initiate the servlet instance
   a) _jspinit() b) _jspdestroy()
   c) both d) none
5. The scripting elements provide the ability to insert_______inside the jsp
   a) java code b) html code
   b) css code d) none
6. Syntax of JSP scriptlet tag is
   a)<% java source code %> b) <! java source code %>
   c) <! java source code !> d) <!@java source code %>
7. The jspscriptlet tag can only declare
   a) variables b) Methods
   c)Instance d) none
8. The declaration of jsp declaration tag is placed outside the ______ ______method
   a) _jspService() b)_jspServiceINIT()
   c) _jspdestroy() d)All of above
9. Jsp action tag _______includes another resource.
   a) jsp:include b) jsp: contain
   c) jsp: exclude d) NONE

10. Tag is used for bean development
    a) jsp:useBean b)jsp:setProperty
    c) jsp:getProperty d)All of above

1	2	3	4	5	6	7	8	9	10
A	A	C	A	A	A	B	A	A	A

References

https://www.javatpoint.com/jsp-tutorial

https://www.w3schools.in/jsp/jsp-page-life-cycle
https://www.geeksforgeeks.org/life-cycle-of-jsp/
https://www.roseindia.net/jsp/jsp-elements

Chapter No. 07

SESSION MANAGEMENT

At the end of this chapter students will be able to understand:

7.1 Session
7.2 Cookie
7.3 URL Rewriting
7.4 Hidden Field
7.5 Working with HTTP Session

7.1 What is a Session?

Session management refers to the process of securely handling multiple requests to a web-based application or service from a single user or entity. Websites and browsers use HTTP to communicate, and a session is a series of HTTP requests and transactions initiated by the same user. Typically, a session is started when a user authenticates their identity using a password or another authentication protocol.
Session is used to store everything that we can get from the client from all the requests the client makes.

7.1.1 How Session Works

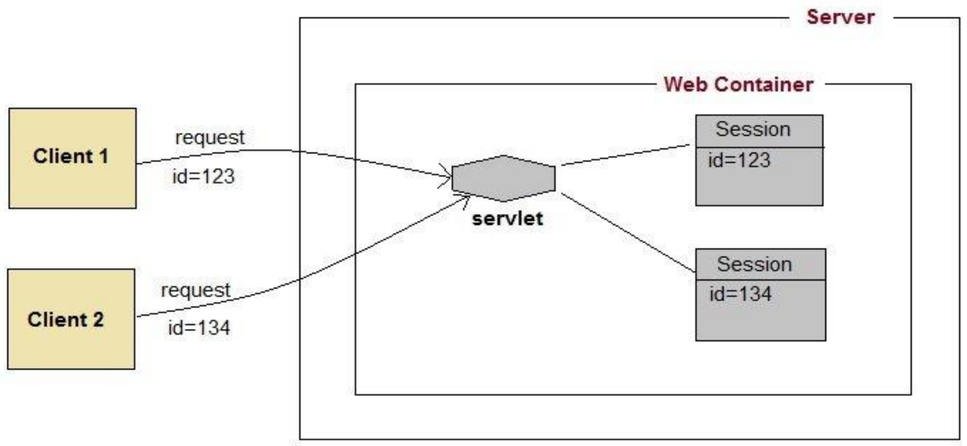


Figure 1: How session works

- On client's first request, the Web Container generates a unique session ID and gives it back to the client with response. This is a temporary session created by web container.
- The client sends back the session ID with each request. Making it easier for the web container to identify where the request is coming from.
- The Web Container uses this ID, finds the matching session with the ID and associates the session with the request.

7.2 Cookie

Cookies are small text files that a website stores on your device (smartphones, computers etc.) when you browse the internet. They are created when your browser loads a particular website, and the site sends information to your browser which then creates a text file. Cookies can store a range of information, including personal data (such as name, home address, email address) and information about your preferred language or location etc. that allows the site to present you with information customized to fit your needs.

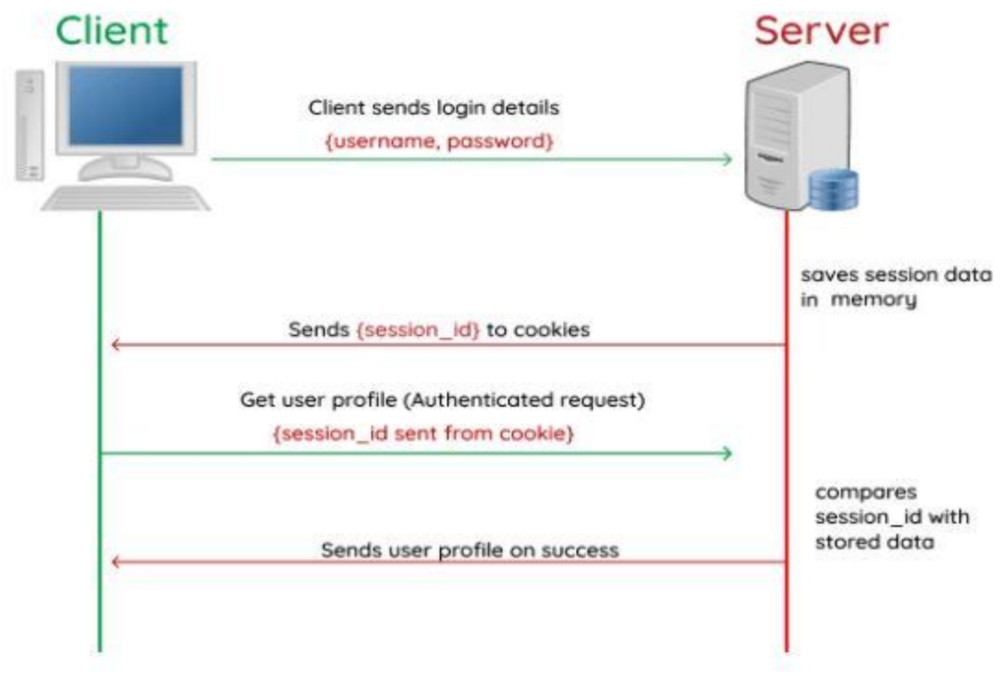

Client
Server
Client sends login details
{username, password}
saves session data in memory
Sends {session_id} to cookies
Get user profile (Authenticated request)
{session_id sent from cookie}
compares session_id with stored data
Sends user profile on success

Figure 2: how session creation process

7.2.1 Types of Cookies

There are two types of cookies. They are as following:

- Session
- Persistent

7.2.1.1 Session cookies:

Session cookies are cookies that last for a session. A session starts when you launch a website or web app and ends when you leave the website or close your browser window. Session cookies contain information that is stored in a temporary memory location which is deleted after the session ends. Unlike other cookies, session cookies are never stored on your device. Therefore, they are also known as transient cookies, non-persistent cookies, or temporary cookies.

- **How do session cookies work?**

The session cookie is a server-specific cookie that cannot be passed to any machine other than the one that generated the cookie. The server creates a “session ID” which is a randomly generated number that temporarily stores the session cookie. This cookie stores information such as the user’s input and tracks the movements of the user within the website. There is no other information stored in the session cookie.

The most common example of a session cookie in action is the shopping cart on e-commerce websites. When you visit an online shop and add items to your shopping cart, the session cookie remembers your selection so your shopping cart will have the items you selected when you are ready to checkout. Without session cookies, the checkout page will not remember your selection and your shopping cart will be empty.
Session cookies also help users to browse and add items to the shopping cart without logging in on an e-commerce site. Only when users checkout, do they have to add their name, address, and payment information.

7.2.1.2 Persistent Cookies:

Persistent cookies are cookies that are stored on a user’s device for a considerably longer time. Therefore, they are also known as permanent cookies. Persistent cookies recognize users and remember their browser settings or preferences on their subsequent visits and help websites provide better user experiences.

After logging into their email accounts, users often see a prompt asking them whether they want to save their login information to access their email account easily in the future. Once they agree to save their information, users can open their email account without keying in their login information again, and they'll find their account as it was when they last logged in.

7.2.2 What is the difference between session cookies and persistent cookies?

Session cookies do not retain any information on your device or send information from your device. These cookies are deleted when the session expires or terminated when the browser window is closed.

Persistent cookies remain on the device until you erase them or they expire. They are ideal for storing information, for instance, persistent cookies help you stay logged in on a website even if you close your browser window.

7.3 URL rewriting

In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource. We can send parameter name/value pairs using the following format:

url?name1=value1&name2=value2&??

A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&). When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can use getParameter() method to obtain a parameter value.

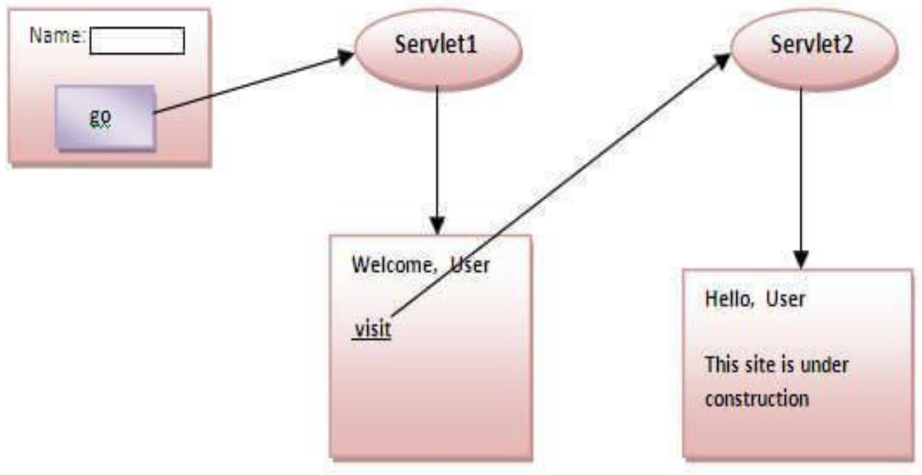


Figure 3: how to obtain parameter value

Advantage of URL Rewriting

- It will always work whether cookie is disabled or not (browser independent).
- Extra form submission is not required on each pages.

Disadvantage of URL Rewriting

- It will work only with links.
- It can send Only textual information.

7.4 Hidden Field

Hidden Form Field (invisible) text field is used for maintaining the state of a user.

In such case, we store the information in the hidden field and get it from another servlet. This approach is better if we have to submit form in all the pages and we don't want to depend on the browser.

Let's see the code to store value in hidden field.

```
<input type="hidden" name="uname" value="gct">
```

Here, uname is the hidden field name and gct is the hidden field value.

It is widely used in comment form of a website. In such case, we store page id or page name in the hidden field so that each page can be uniquely identified.

Advantage of Hidden Form Field

It will always work whether cookie is disabled or not.

Disadvantage of Hidden Form Field:

It is maintained at server side.

Extra form submission is required on each pages.

Only textual information can be used.

7.5 Working with HTTP Session

HttpSession object is used to store entire session with a specific client. We can store, retrieve and remove attribute from HttpSession object. Any servlet can have access to HttpSession object throughout the getSession() method of the HttpServletRequest object.

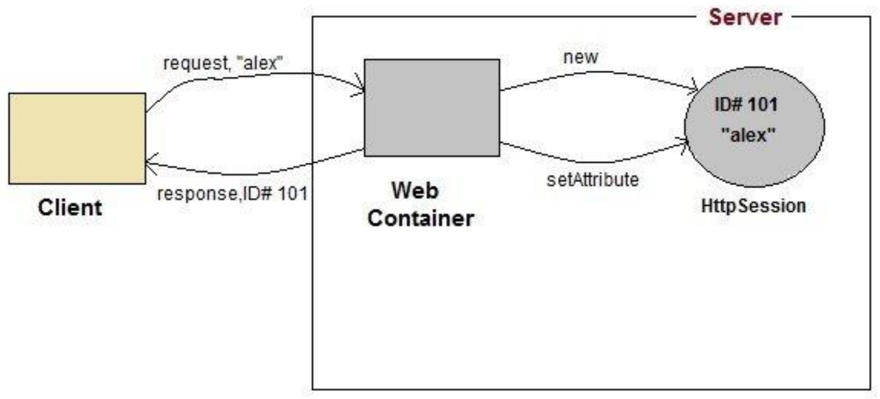


Fig ure 4: how HTTP session works

1. On client's first request, the Web Container generates a unique session ID and gives it back to the client with response. This is a temporary session created by web container.
2. The client sends back the session ID with each request. Making it easier for the web container to identify where the request is coming from.
3. The Web Container uses this ID, finds the matching session with the ID and associates the session with the request.

EXERCISE

PART I: SHORT QUESTIONS

1. What is session
2. What is cookies
3. What is session cookies
4. What is persistent cookies
5. Explain URL rewriting
6. Explain hidden field
7. Explain working with http session

PART II: LONG QUESTIONS

1. What is a session? How it works.
2. Write a note on Cookies.

MULTIPLE CHOICE QUESTIONS

1. Which of the below is not a session tracking method?

a) URL rewriting

b) History

c) Cookies

d) SSL sessions

2.Which of the following is stored at client side?

a) URL rewriting

b) Hidden form fields

c) SSL sessions

d) Cookies

3. Which of the following leads to high network traffic?

a) URL rewriting

b) Hidden form fields

c) SSL sessions

d) Cookies

4. What is the maximum size of cookie?

a) 4 KB

b) 4 MB

c) 4 bytes

d) 40 KB

5. Which method creates unique fields in the HTML which are not shown to the user?

a) User authentication

b) URL writing

c) HTML Hidden field

d) HTML invisible field

6. Which object is used by spring for authentication?

a) ContextHolder

b) SecurityHolder

c) AnonymousHolder

d) SecurityContextHolder

7. Which object of HttpSession can be used to view and manipulate information about a session?

a. session identifier

b. creation time

c. last accessed time

d. All mentioned above

8. Which class provides stream to read binary data such as image etc. from the request object?

a. ServltInputStream

b. ServletOutputStream

c. Both A & B

d. None of the above

9. Which of these ways used to communicate from an applet to servlet?

a. RMI Communication

b. HTTP Communication

c. Socket Communication

d. All mentioned above

10. Which method is used to send the same request and response objects to another servlet in RequestDispacher ?

a. forward()

b. sendRedirect()

c. Both A & B

d. None of the above

MCQ's Answes

1	2	3	4	5	6	7	8	9	10
B	D	A	A	C	D	D	A	D	A

References

- Session Retrieved from https://www.studytonight.com/servlet/url-rewriting-for-session-management.php
- Cookie Retrieved from https://www.cookieyes.com/blog/session-cookies
- URL Rewriting Retrieved from https://www.javatpoint.com/url-rewriting-in-session-tracking#
- Hidden Field Retrieved from https://www.javatpoint.com/hidden-form-field-in-session-tracking#session2ex
- Working with HTTP Session Retrieved from https://www.studytonight.com/servlet/httpsession.php

Chapter No. 08

JDBC PROGRAMMING

Objectives

After completion of this chapter students will be able to:

8.1- Introduction
8.2- Configure MySQL
8.3- Connecting to the Database
8.4- Accessing Data
8.5- The execute Query Method
8.6 The execute Update Method

8.1 Introduction JDBC Programming

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver
- Native Driver
- Network Protocol Driver
- Thin Driver

We have discussed the above four drivers in the next chapter.

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.

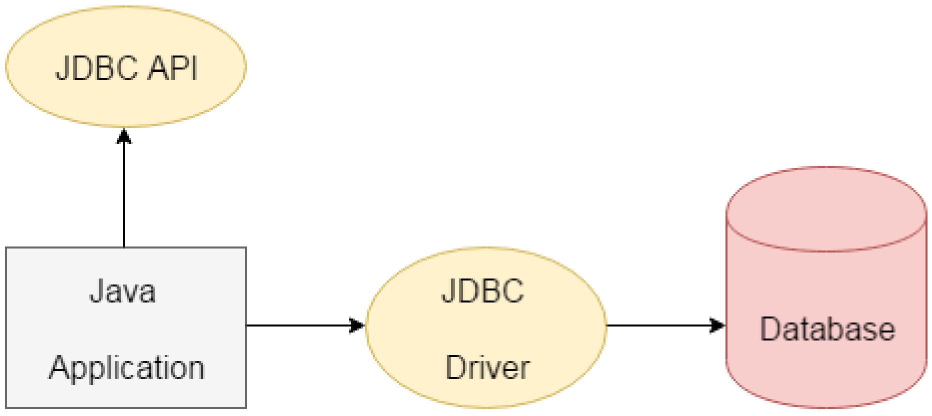


Figure 1: JDBC

The current version of JDBC is 4.3. It is the stable release since 21st September, 2017. It is based on the X/Open SQL Call Level Interface. The java.sql package contains classes and interfaces for JDBC API.

8.1.1 Why Should We Use JDBC

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

We can use JDBC API to handle database using Java program and can perform the following activities:

- Connect to the database
- Execute queries and update statements to the database
- Retrieve the result received from the database.

8.1.2 What is API

API (Application programming interface) is a document that contains a description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other. An API can be created for applications, libraries, operating systems, etc.

8.2 Configure MySQL

8.2.1 Introduction to My SQL

MySQL pronounced either "My S-Q-L" or "My Sequel," is an open source relational database management system. It is depends on the structure query language (SQL), which is utilized for adding, modifying, and removing information in the database. Standard SQL commands, such like DROP, ADD, INSERT, and UPDATE can be utilized with My SQL.

MySQL runs on virtually all of the platforms, by including UNIX, Linux and Windows. It is wholly multi-threaded by using kernel threads, and provides application program to interfaces (APIs) for various programming languages, including C, C++, Java, Eiffel, Perl, PHP, Python, and Tcl.

My SQL is utilized in a wide range of applications, by including data warehousing, e-commerce, logging applications, Web databases, and distributed applications. This is also increasingly embedded in third-party software and other technologies.

MySQL is most commonly found on Web servers. A website that uses My SQL might include Web pages that access information through a database. These pages are frequently referred to as "dynamic," meaning the content of each of page is produced from a database as the page loads. Websites that utilized dynamic Web pages are frequently referred to as database-driven websites.

Various database-driven websites that use My SQL also utilize a Web scripting language like PHP to access information from the database. MySQL commands can be incorporated to the PHP code, permitting part or all of a Web page to be produced from database information. Since both MySQL and PHP are both open source (meaning they are free to download and use), the PHP/MySQL combination has become a well-liked choice for database-driven websites.

8.2.2 Features of MY-SQL

Due to its unique storage engine architecture MySQL performance is extremely high.
Supports large number of embedded applications that makes MySql extremely flexible.
Use of Triggers, Stored process and views which permits the developer to give a higher productivity.
Allows transactions to be rolled back, commit and crash recovery.
Triggers & cursor.

8.2.3 Benefits of MySQL:

Whether you are a Web developer or a dedicated network administrator with an interest in creating database applications, MySQL is simple to use, yet very powerful, secure, and scalable. And due to its small size and speed, this is the ideal database solution for Web sites.
Some of its advantages include the given:
1) It's easy to use: Whereas a basic knowledge of SQL is needed and most relational databases require the same knowledge – My SQL is extremely easy to use. With only a few simple SQL statements, you can developed and interact with MySQL.
2) Its secure:
It includes solid data security layers that protect sensitive data from intruders. Rights can be set to permit all or some privileges to individuals. Passwords are encrypted.
3) Its inexpensive: It is included for free with NetWare 6.5 and available by free download from MySQL Web site.
4) Its fast: In the interest of speed, My SQL designers take the decision to offer fewer features than other major database competitors, such like Sybase and Oracle. Though, in spite of having fewer features than the other commercial database products, MySQL still offers all features required by most database developers.
5) Its scalable: It can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is approximate 4 GB. Though, you can increase this number to a theoretical limit of 8 TB of data.

6) **It manages memory very well:** MySQL server has been thoroughly tested to stop memory leaks.
7) **It runs on many operating systems:** MySQL runs on various operating systems, by including Novell NetWare, Linux, Windows many varieties of UNIX (such like Solaris, Sun, AIX, and DEC UNIX), OS/2, FreeBSD, and others.
8) **It supports several development interfaces:** Development interfaces include ODBC, JDBC and scripting (PHP and Perl), letting you developing database solutions that run not only in your NetWare 6.5 environment, but across all of the major platforms, by including UNIX, Linux and Windows.

8.2.4 MySQL Installation on Windows

Here we will show you step by step how to install MySQL on the Windows platform using the MySQL Installer.
The simplest and recommended method is to download MySQL Installer for Windows from https://dev.mysql.com/downloads/installer/ and execute it.

After downloading, unzip it, and double click the MSI installer .exe file.
Then follow the steps below:
1. "Choosing a Setup Type" screen: Choose "Full" setup type. This installs all MySQL products and features. Then click the "Next" button to continue.
2. "Check Requirements" screen: The installer checks if your pc has the requirements needed. If there is some failing requirements, click on each item to try to resolve them by clicking on the Execute button that will install all requirements automatically. Click "Next".
3. "Installation" screen: See what products that will be installed. Click "Execute" to download and install the Products. After finishing the installation, click "Next".
4. "Product Configuration" screen: See what products that will be configured. Click the "MySQL Server 8.0.23" option to configure the MySQL Server. Click the "Next" button. Choose the "Standalone MySQL Server/Classic MySQL Replication" option and click on the "Next" button. In page "Type and Networking" set Config Type to "Development Computer" and "Connectivity" to "TCP/IP" and "Port" to "3006". Then, click the "Next" button.

5. "Authentication Method" screen: Choose "Use Strong Password Encryption for Authentication". Click "Next".
6. "Accounts and Roles" screen: Set a password for the root account. Click "Next".
7. "Windows Service" screen: Here, you configure the Windows Service to start the server. Keep the default setup, then click "Next".
8. "Apply Configuration" screen: Click the "Execute" button to apply the Server configuration. After finishing, click the "Finish" button.
9. "Product Configuration" screen: See that the Product Configuration is completed. Keep the default setting and click on the "Next" and "Finish" button to complete the MySQL package installation.
10. In the next screen, you can choose to configure the Router. Click on "Next", "Finish" and then click the "Next" button.
11. "Connect To Server" screen: Type in the root password (from step 6). Click the "Check" button to check if the connection is successful or not. Click on the "Next" button.
12. "Apply Configuration" screen: Select the options and click the "Execute" button. After finishing, click the "Finish" button.
13. "Installation Complete" screen: The installation is complete. Click the "Finish" button.

Open the MySQL Command Line Client from cmd.
You should see a mysql> prompt. If you have set any password, write your password here.
Now, you are connected to the MySQL server, and you can execute all the SQL command.

8.3 Connecting to the Database

The programming involved to establish a JDBC connection is fairly simple. Here are these simple four steps −

- **Import JDBC Packages** − Add import statements to your Java program to import required classes in your Java code.

- **Register JDBC Driver** − This step causes the JVM to load the desired driver implementation into memory so it can fulfill your JDBC requests.
- **Database URL Formulation** − This is to create a properly formatted address that points to the database to which you wish to connect.
- **Create Connection Object** − Finally, code a call to the DriverManager object's getConnection( ) method to establish actual database connection.

8.3.1 Import JDBC Packages

The Import statements tell the Java compiler where to find the classes you reference in your code and are placed at the very beginning of your source code.

To use the standard JDBC package, which allows you to select, insert, update, and delete data in SQL tables, add the following imports to your source code −

```
import java.sql.* ;  // for standard JDBC programs
import java.math.* ; // for BigDecimal and BigInteger support
```

8.3.2 Register JDBC Driver

You must register the driver in your program before you use it. Registering the driver is the process by which the Oracle driver's class file is loaded into the memory, so it can be utilized as an implementation of the JDBC interfaces.
You need to do this registration only once in your program.
Class.forName()
The most common approach to register a driver is to use Java's Class.forName() method, to dynamically load the driver's class file into memory, which automatically registers it. This method is preferable because it allows you to make the driver registration configurable and portable.
The following example uses Class.forName( ) to register the Oracle driver –

```
try {
Class.forName("oracle.jdbc.driver.OracleDriver");
}
catch(ClassNotFoundException ex) {
System.out.println("Error: unable to load driver class!");
System.exit(1);
}
```

8.3.3 Database URL Formulation

After you've loaded the driver, you can establish a connection using the DriverManager.getConnection() method. For easy reference, let me list the three overloaded DriverManager.getConnection() methods −

- getConnection(String url)
- getConnection(String url, Properties prop)
- getConnection(String url, String user, String password)

Here each form requires a database URL. A database URL is an address that points to your database.

Formulating a database URL is where most of the problems associated with establishing a connection occurs.

Following table lists down the popular JDBC driver names and database URL.

RDBMS	JDBC driver name	URL format
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://hostname/ databaseName
ORACLE	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@hostname:portNumber:databaseName
DB2	COM.ibm.db2.jdbc.net.DB2Driver	jdbc:db2:hostname:port Number/databaseName
Sybase	com.sybase.jdbc.SybDriver	jdbc:sybase:Tds:hostname: port Number/databaseName

All the highlighted part in URL format is static and you need to change only the remaining part as per your database setup.

8.3.4 Create Connection Object

We have listed down three forms of DriverManager.getConnection() method to create a connection object.

- **Using a Database URL with a username and password**

The most commonly used form of getConnection() requires you to pass a database URL, a username, and a password −

Assuming you are using Oracle's thin driver, you'll specify a host:port:databaseName value for the database portion of the URL.

If you have a host at TCP/IP address 192.0.0.1 with a host name of amrood, and your Oracle listener is configured to listen on port 1521, and your database name is EMP, then complete database URL would be −

```
jdbc:oracle:thin:@amrood:1521:EMP
```

Now you have to call getConnection() method with appropriate username and password to get a Connection object as follows −

```
String URL = "jdbc:oracle:thin:@amrood:1521:EMP";
String USER = "username";
String PASS = "password"
Connection conn = DriverManager.getConnection(URL, USER, PASS);
```

8.3.5 Closing JDBC Connections

At the end of your JDBC program, it is required explicitly to close all the connections to the database to end each database session. However, if you forget, Java's garbage collector will close the connection when it cleans up stale objects.

To close the above opened connection, you should call close() method as follows −

conn.close();

8.4 Accessing Data

Data access refers to a user’s ability to access or retrieve data stored within a database or other repository. Users who have data access can store, retrieve, move or manipulate stored data, which can be stored on a wide range of hard drives and external devices.

There are two ways to access stored data: sequential access and random access.

8.4.1 Sequential access

The sequential method requires information to be moved within the disk using a seek operation until the data is located. Each segment of data has to be read one after another until the requested data is found.

8.4.2 Random access

Reading data randomly allows users to store or retrieve data anywhere on the disk, and the data is accessed in constant time.

Oftentimes when using random access, the data is split into multiple parts or pieces and located anywhere randomly on a disk. Sequential files are usually faster to load and retrieve because they require fewer seek operations.

8.5 The execute Query Method

This method is used to retrieve data from database using SELECT query. This method returns the ResultSet object that returns the data according to the query.

8.5.1 The SQL SELECT Statement

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

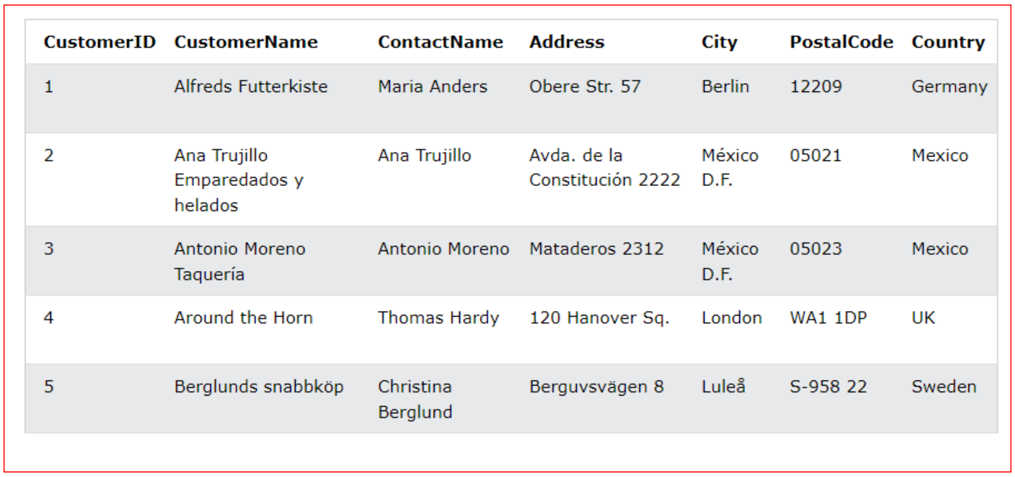
SELECT Syntax

```
SELECT column1, column2, ...
FROM table_name;
```

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

```
SELECT * FROM table_name;
```

Below is a selection from the "Customers" table in the Northwind sample database.



CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Fig 8.2 show table data

The following SQL statement selects the "CustomerName" and "City" columns from the "Customers" table.

```
SELECT CustomerName, City FROM Customers;
```

The following SQL statement selects all the customers from the country "Mexico", in the "Customers" table:

SELECT * FROM Customers
WHERE Country='Mexico';

8.6 The execute Update Method

This statement is used to execute SQL statements which update or modify database.
This method returns an int value which represent number of rows affected by the query. This will be 0 for statement which are returning nothing.
This method is use to execute non select query.
DML :INSERT , UPDATE and DELETE
DDL : CREATE, ALTER

8.6.1 DML:

It stands for Data Manipulation Language. The DML commands deal with the manipulation of existing records of a database. It is responsible for all changes that occur in the database. The changes made in the database using this command can't save permanently because its commands are not auto-committed. Therefore, changes can be rollback. The following commands come under DML language:
INSERT: It is a SQL query that allows us to add data into a table's row.
The following SQL statement inserts a new record in the "Customers" table.
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

- **UPDATE**: This command is used to alter or modify the contents of a table.

  The following SQL statement updates the first customer (CustomerID = 1) with a new contact person *and* a new city.

  UPDATE Customers
  SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
  WHERE CustomerID = 1;

- **DELETE**: This command is used to delete records from a database table, either individually or in groups.

  The following SQL statement deletes the customer "AlfredsFutterkiste" from the "Customers" table.

  ```
  DELETE FROM Customers WHERE CustomerName='AlfredsFutterkiste'
  ;
  ```

8.6.1.1 Why we use DML commands?

- The following are the reasons to use the DML commands:
- It helps users to change the data in a database table.
- It helps users to specify what data is needed.
- It facilitates human interaction with the system.

8.6.2 DDL

DDL stands for Data Definition Language. As the name suggests, the DDL commands help to define the structure of the databases or schema. When we execute DDL statements, it takes effect immediately. The changes made in the database using this command are saved permanently because its commands are auto-committed. The following commands come under DDL language:

- **CREATE**: It is used to create a new database and its objects such as table, views, function, stored procedure, triggers, etc.

  The following example creates a table called "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City:

  ```
  CREATE TABLE Persons (
     PersonID int,
     LastName varchar(255),
     FirstName varchar(255),
     Address  varchar(255),
  ```

```
   City  varchar(255)
);
```

- **DROP**: It is used to delete the database and its objects, including structures, from the server permanently.

  The following SQL statement drops the existing table "Shippers":

  ```
  DROP TABLE Shippers;
  ```

- **ALTER**: It's used to update the database structure by modifying the characteristics of an existing attribute or adding new attributes.

  The following SQL adds an "Email" column to the "Customers" table:

  ```
  ALTER TABLE Customers
  ADD Email varchar(255);
  ```

- **TRUNCATE**: It is used to completely remove all data from a table, including their structure and space allocates on the server.

  The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.
  TRUNCATE TABLE *table_name*;

- **RENAME**: This command renames the content in the database.
  To rename a column in a table, use the following syntax:
  ALTER TABLE *table_name*
  RENAME COLUMN *old_name* to *new_name*;

8.6.2.1 Why we use DDL commands?

The following are the reasons to use DDL commands:

- It allows us to store shared data in a database.
- It improved integrity due to the data independence feature.
- It will enable multiple users to work on the same databases.
- It improved security efficient data access.

EXERCISE

PART I: SHORT QUESTIONS

1- Define JDBC
2- Why should we use JDBC
3- What is API
4- What is MY SQL
5- Explain advantages of MY SQL
6- Explain connecting to the database
7- What is sequential access
8- What is random access
9- Why we import JDBC packet
10- Explain register JDBC driver
11- Explain connection object

PART II: LONG QUESTIONS

1- Write a note on MY SQL
2- Write a note on execute query method
3- Write a note on execute update method
4- Write the note on DML and DDL.

MULTIPLE CHOICE QUESTIONS

1) What are the types of ResultSet in JDBC?

a. Forward ResultSet

b. Scrollable ResultSet

c. Only a

d. Both a and b

2) Select the packages in which JDBC classes are defined?

a. jdbc and javax.jdbc

b. rdb and javax.rdb

c. jdbc and java.jdbc.sql

d. sql and javax.sql

3) Which of the following method is used to perform DML statements in JDBC?

a. executeResult()

b. executeQuery()

c. executeUpdate()

d. execute()

4) How many transaction isolation levels provide the JDBC through the Connection interface?

a. 3

b. 4

c. 7

d. 2

5) Which of the following method is static and synchronized in JDBC API?

a. getConnection()

b. prepareCall()

c. executeUpdate()

d. executeQuery()

6) Which methods are required to load a database driver in JDBC?

a. getConnection()

b. registerDriver()

c. forName()

d. Both b and c

7) Parameterized queries can be executed by?

a. ParameterizedStatement

b. PreparedStatement
c. CallableStatement and Parameterized Statement
d. All kinds of Statements

8) Which of the following is not a valid statement in JDBC?
a. Statement
b. PreparedStatement
c. QueryStatement
d. CallableStatement

9) Stored procedure can be called by using the ....
a. CallableStatement
b. Statement
c. CalledStatement
d. PreparedStatement

10) A good way to debug JDBC-related problems is to enable???..?
a. JDBC tracing
b. Exception handling
c. Both a and b
d. Only b

11) Which JDBC driver can be used in servlet and applet both?
a. Type 3
b. Type 4
c. Type 3 and Type 2
d. Type 3 and Type 4

12) JDBC-ODBC driver is also known as?
a. Type 4
b. Type 3
c. Type 1
d. Type 2

13) How many stages are used by Java programmers while using JDBC in their programs?
a. 3
b. 2
c. 5
d. 6

14) How many ways to register a driver?
a. 2

b. 3
c. 4
d. 5

MCQ's Answers

1	2	3	4	5	6	7	8	9	10
D	D	C	B	A	D	B	C	B	A
11	12	13	14						
D	C	D	C						

References

Introduction JDBC Programming Retrieved from https://www.javatpoint.com/java-jdbc

ConfigureMySQL Retrieved from https://www.tutorsglobe.com/homework-help/computer-programming/my-sql-73688.aspx
https://www.w3schools.com/mysql/mysql_install_windows.asp

Connectingto theDatabase Retrieved from https://www.tutorialspoint.com/jdbc/jdbc-db-connections.htm

Accessing Data Retrieved from https://www.techopedia.com/definition/26929/data-access

The execute QueryMethod Retrieved from

https://www.w3schools.com/sql/sql_select.asp

The execute UpdateMethod Retrieved from

https://www.javatpoint.com/ddl-vs-dml#:~:text=Key%20Differences%20between%20DDL%20and%20DML%20Commands&text=Data%20Definition%20Language%20(DDL)%20statements,already%20exists%20in%20the%20database.

https://www.w3schools.com/sql/sql_delete.asp

Chapter No. 09

CUSTOM TAGS

Objectives

After completion of this chapter students will be able to:

9.1- Custom Tags Overview

9.2- Custom Tag Handlers

9.3- Customizing Tag behavior with attributes

9.1 Custom Tags

A custom tag is a user-defined JSP language element. When a JSP page containing a custom tag is translated into a servlet, the tag is converted to operations on an object called a tag handler. The Web container then invokes those operations when the JSP page's servlet is executed.

JSP tag extensions lets you create new tags that you can insert directly into a JavaServer Page. The JSP 2.0 specification introduced the Simple Tag Handlers for writing these custom tags.

To write a custom tag, you can simply extend SimpleTagSupport class and override the doTag() method, where you can place your code to generate content for the tag.

Advantages of Custom Tags in JSP

Portable – Once declared in the Tag Library, these tags can be used anywhere.

Simple – They are simple and convenient to use.

Expressive – It provides a wide range of functionalities, scripting elements to the page authors.

Usable from different scripting languages – This functionality can extend to other scripting languages.

No need for scriptlet tag – Once we create a custom tag, we don’t need scriptlet tag. Though it is a bad approach in JSP.

Reusable – We can use the similar business logic over and over again.

Separation of business logic from JSP – For easy maintenance, Custom tags separate business logic and JSP.

9.1.1 Create "Hello" Tag

Consider you want to define a custom tag named <ex:Hello> and you want to use it in the following fashion without a body −
<ex:Hello />
To create a custom JSP tag, you must first create a Java class that acts as a tag handler. Let us now create the HelloTag class as follows −

```
packagecom.tutorials;

importjavax.servlet.jsp.tagext.*;
importjavax.servlet.jsp.*;
import java.io.*;

public class HelloTag extends SimpleTagSupport {
public void doTag() throws JspException, IOException {
JspWriter out = getJspContext().getOut();
out.println("Hello Custom Tag!");
  }
}
```

The above code has simple coding where the doTag() method takes the current JspContext object using the getJspContext() method and uses it to send "Hello Custom Tag!" to the current JspWriter object.

9.2 Tag handler

The object that implements a custom tag is called a tag handler. JSP technology defines two types of tag handlers: simple and classic. Simple tag handlers can be used only for tags that do not use scripting elements in attribute values or the tag body. Classic tag handlers must be used if scripting elements are required.

9.2.1 Simple Tag Handlers

The classes and interfaces used to implement simple tag handlers are contained in the javax.servlet.jsp.tagext package. Simple tag handlers implement the SimpleTag interface. Interfaces can be used to take an existing Java object and make it a tag handler. For most newly created handlers, you would use the SimpleTagSupport classes as a base class.

The heart of a simple tag handler is a single method, doTag, which is invoked when the end element of the tag is encountered. Note that the default implementation of the doTag method of SimpleTagSupport does nothing.

A tag handler has access to an API that allows it to communicate with the JSP page. The entry point to the API is the JSP context object (javax.servlet.jsp.JspContext). The JspContext object provides access to implicit objects. PageContext extends JspContext with servlet-specific behavior. Through these objects, a tag handler can retrieve all the other implicit objects (request, session, and application) that are accessible from a JSP page. If the tag is nested, a tag handler also has access to the handler (called the parent) that is associated with the enclosing tag.

9.2.2 Classic Tag Handler

The classes and interfaces used to implement classic tag handlers are contained in the javax.servlet.jsp.tagext package. Classic tag handlers implement either the Tag, the IterationTag, or the BodyTag interface. Interfaces can be used to take an existing Java object and make it a tag handler. For newly created classic tag handlers, you can use the TagSupport and BodyTagSupport classes as base classes. These classes and interfaces are contained in the javax.servlet.jsp.tagext package.

9.3 Customized tags behavior with attributes

Custom tag attributes are used to provide the requested information to the tag handler class. A custom tag can have either no attributes or many attributes. These attributes are used after the tag name in the start tag and they contains a name and its respective value, these tag attributes are customizing the custom tag behavior as the parameters customized the behavior of a method, more than one attributes can be separated by giving a white space.

9.3.1 Tags with Attributes

A simple tag can have attributes. Attributes customize the behavior of a custom tag just as parameters customize the behavior of a method. There are three types of attributes:

- Simple attributes
- Fragment attributes
- Dynamic attributes

9.3.1.1 Simple Attributes

Simple attributes are evaluated by the container before being passed to the tag handler. Simple attributes are listed in the start tag

9.3.1.2 Fragment Attributes

A JSP fragment is a portion of JSP code passed to a tag handler that can be invoked as many times as needed. You can think of a fragment as a template that is used by a tag handler to produce customized content. Thus, unlike a simple attribute which is evaluated by the container, a fragment attribute is evaluated by a tag handler during tag invocation.

9.3.1.3 Dynamic Attributes

A dynamic attribute is an attribute that is not specified in the definition of the tag. Dynamic attributes are used primarily by tags whose attributes are treated in a uniform manner but whose names are not necessarily known at development time.

EXERCISE

PART I: SHORT QUESTIONS

1. Define custom tag.
2. Write the advantages of Custom Tags
3. Define custom Tag Handler
4. Define simple tag handler
5. Define classic tag handler
6. Simple attributes
7. Fragment attributes
8. Dynamic attributes

PART III: LONG QUESTIONS

1. Define custom tag in detail.
2. How to Customize Tag behavior with attributes

MULTIPLE CHOICE QUESTIONS

1. Which technology do we mix our business logic with the presentation logic?
A. Servlet
B. JSP
C. Both A and B
D. None of the above
2. Which of the following is an advantage of the statement – Separation of business logic from JSP ?
A. Custom Tags in JSP
B. JSP Standard Tag Library
C. All the above
D. None of the above
3. Which tag is used to execute java source code in JSP?
A. Declaration Tag
B. Scriptlet tag
C. Expression tag
D. None of the above
4. A JSP page consists of which tags?
A. HTML tags
B. JSP tags
C. Both A & B
D. None of the above
5. Which packages does a JSP API consist of?
A. javax.servlet.jsp
B. java.servlet
C. javax.servlet.jsp.tagext
D. Both A & C
6. Which of the scripting of JSP not putting content into service method of the converted servlet?
A. Declarations
B. Expressions
C. Scriptlets
D. None of the above
7. The difference between Servlets and JSP is the …………….
A. translation

B. compilation
C. syntax
D. Both A and B
8. JSP includes a mechanism for defining ………………………… or custom tags.

A. static attributes
B. local attributes
C. global attributes
D. dynamic attributes
9. Which is not a directive?
A. include
B. page
C. export
D. taglib
10. Which http method send by browser that asks the server to get the page only? A. get
B. option
C. put
D. post

MCQ's Answers

1	2	3	4	5	6	7	8	9	10
A	A	B	C	D	B	C	D	C	A

References

Custom Tags Overview. (n.d.). Retrieved from https://www.tutorialspoint.com/jsp/jsp_custom_tags.htm

Customizing Tag behavior with attributes. (n.d.). Retrieved from https://docs.oracle.com/cd/E19575-01/819-3669/bnaoy/index.html

Custom Tag Handlers (n.d.). Retrieved from rapidapi.com: https://rapidapi.com/blog/api-glossary/http-request-methods/

Chapter No. 10

JAVA SERVER FACES

At the end of this chapter students will be able to:

10.1- Introduction
10.2- Using JSF with JSP
10.3- JSF Life Cycle
10.4- Events Handling
10.5- JSF Tag Library

10.1 INTRODUCTION

It is a server side component based user interface framework. It is used to develop web applications. It provides a well-defined programming model and consists of rich API and tag libraries. Java Server Faces is a standardized display technology, which was formalized in a specification through the Java Community Process.

The JSF API provides components (inputText, commandButtonetc) and helps to manage their states. It also provides server-side validation, data conversion, defining page navigation, provides extensibility, supports for internationalization, accessibility etc.

The JSF Tag libraries are used to add components on the web pages and connect components with objects on the server. It also contains tag handlers that implements the component tag.

10.1.1 Benefits of Java Server Faces

- It provides clean and clear separation between behavior and presentation of web application. You can write business logic and user interface separately.
- Java Server Faces API's are layered directly on top of the Servlet API. Which enables using different presentation technologies, creating your own custom components directly from the component classes.
- Inclusion of Facelets technology in Java Server Faces 2.0, provides massive advantages to it. Facelets is now the preferred presentation technology for building Java Server Faces based web applications.

- JSF reduces the effort in creating and maintaining applications, which will run on a Java application server and will render application UI on to a target client.
- JSF facilitates Web application development by –
  a. Providing reusable UI components
  b. Making easy data transfer between UI components
  c. Managing UI state across multiple server requests
  d. Enabling implementation of custom components
  e. Wiring client-side event to server-side application code

10.2 USING JSF WITH JSP

JSF plays nicely with JSP. In fact, all JSF implementations must support JSP and provide tag libraries with custom actions for representing the standard JSF UI components in JSP pages. If you're familiar with JSP, adding JSF to the mix is fairly simple. While you can use JSP to develop complete, simple applications without writing a single line of Java code, be aware that most JSF applications require event handlers and backend logic implemented as Java classes; hence, you must have a Java programmer handy when you use JSF.

Using JSF Components in a JSP Page

The JSP page for the initial version of the filtering criteria form is shown in Example below:

Example: Initial filtering criteria form JSP page (expense/stage1/filterArea.jsp)

```
<%@ page contentType="text/html" %>
<%@ tagliburi="http://java.sun.com/jsf/html" prefix="h" %>
<%@ tagliburi="http://java.sun.com/jsf/core" prefix="f" %>
<f:view>
<h:form>
   From: <h:inputText size="8" />
<br>
   To: <h:inputText size="8" />
<br>
   Status:
<h:selectManyCheckbox>
<f:selectItemitemValue="1" itemLabel="Open" />
```

```
<f:selectItemitemValue="2" itemLabel="Submitted" />
<f:selectItemitemValue="3" itemLabel="Accepted" />
<f:selectItemitemVvalue="4" itemLabel="Rejected" />
</h:selectManyCheckbox>
<p>
<h:commandButton value="Filter" />
</h:form>
</f:view>
```

The page starts with a JSP page directive, saying that the page generates HTML, and two taglib directives that declare the tag libraries used in the page. Each taglib directive contains two attributes: the uri attribute specifies the unique identifier for the library and the prefix attribute defines the namespace prefix used for elements from the library in this page. Note that even though the uri attribute value looks like a URL, it doesn't mean that the tag library is accessed over the Internet when you run the page. It's just a string that uniquely identifies a specific library, and a URL is typically used for public libraries because it's a pretty good guarantee that it won't clash with the identifier.

10.3 JSF LIFE CYCLE

JSF application life cycle consists of six phases which are as follows −

- Restore view phase
- Apply request values phase; process events
- Process validations phase; process events
- Update model values phase; process events
- Invoke application phase; process events
- Render response phase

The following diagram shows the typical order where the six phases perform.

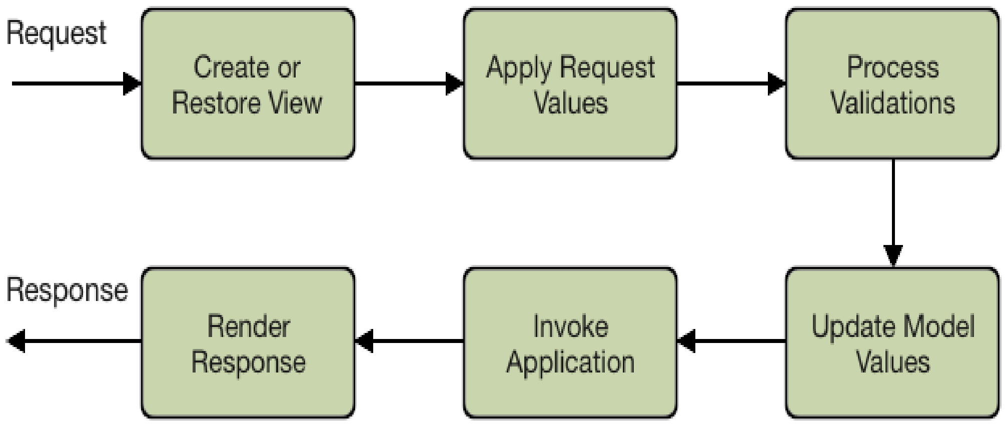


Figure 6: JSF Life Cycle

1. Restore View Phase: This phase begins each time a consumer requests a JSF page by simply clicking on a link, button, and so on. In this page view generation, the binding of components to the event handlers and validators is performed as well as the view is preserved within the FacesContext object.

2. Apply request value: The objective of this phase is made for each component to retrieve the current state

3. Process Validations Phase: In this phase, local values stored to get the component in the tree will be in comparison to the validation rules authorized to get the components.

4. Update Model Values Phase: Following verifying that data is valid in the last phase local values of elements could be started related to server-based object properties, for example, backing beans.

5. Invoke Application Phase: Ahead of this phase the component values had been transformed, validated, as well as, put on the bean objects so that you can utilize these to perform the application's business logic.

6. Render Response Phase: This phase fetches a new view if needed using values from server-side beans then we output the view using the values that are in the tree and then for beans that are not on the request in other words on the session scope or application scope we will then save those the current state.

10.4 EVENTS HANDLING

When a user clicks a JSF button or link or changes any value in the text field, JSF UI component fires an event, which will be handled by the application code. To handle such an event, an event handler is to be registered in the application code or managed bean.

When a UI component checks that a user event has occured, it creates an instance of the corresponding event class and adds it to an event list. Then, Component fires the event, i.e., checks the list of listeners for that event and calls the event notification method on each listener or handler.

JSF also provide system level event handlers, which can be used to perform some tasks when the application starts or is stopping.

Following are some important *Event Handler* in JSF 2.0 −

S.No	Event Handlers & Description
1	valueChangeListener Value change events get fired when the user make changes in input components.
2	actionListener Action events get fired when the user clicks a button or link component.
3	Application Events Events firing during JSF lifecycle: PostConstructApplicationEvent, PreDestroyApplicationEvent ,PreRenderViewEvent.

- **ValueChangeListener:**

When user interacts with input components, such as h:inputText or h:selectOneMenu, the JSF fires a valueChangeEvent which can be handled in two ways.

Technique	Description
Method Binding	Pass the name of the managed bean method in valueChangeListenerattribute of UI Component.
ValueChangeListener	Implement ValueChangeListener interface and pass the implementation class name to

	valueChangeListener attribute of UI Component.

- **ActionListener:**

When user interacts with components, such as h:commandButton or h:link, the JSF fires an action events which can be handled in two ways.

Technique	**Description**
Method Binding	Pass the name of the managed bean method in actionListener attribute of UI Component.
ActionListener	Implement ActionListener interface and pass the implementation class name to actionListener attribute of UI Component.

10.5 JSF TAG LIBRARY

To develop a JSF application in a convenient way Sun Microsystems, Inc provided some tag libraries that fulfils the basic requirement or functionality to the discrete JSF applications. To develop JSF applications not only its own tag libraries core and html but, also JSTL core1.2 and JSTL functions1.1 tag libraries are supported. Tag libraries of JSF core and html expresses the JavaServer Faces interface on the inside view template, these can be used with JSP or Facelets or both.

Each JSF Tag Libraries have their own namespace, or prefix these are as follows :

SF Core : JSF core library.
JSF Html : JSF html library.
JSF Composite : JSF composite library.
JSF ui : JSF facelets library.
JSTL Core : JSTL core library.
JSTL Functions : JSTL functions library.
To include above these tag libraries you will have to use taglib directive (<%@ tagliburi="" prefix="" %>) on the JSP page.

Following table describes the libraries and their corresponding uri and prefix.

Library	URI	Prefix	Number of Tags

JSF core	http://java.sun.com/jsf/core	f	27
JSF html	http://java.sun.com/jsf/html	h	31
JSF composite	http://java.sun.com/jsf/composite	composite	11
JSF ui	http://java.sun.com/jsf/facelets	ui	12
JSTL core	http://java.sun.com/jsp/jstl/core	c	7
JSTL functions	http://java.sun.com/jsp/jstl/functions	fn	16

NOTE : You would be required the jsf-impl.jar, jstl.jar, standard.jar files to use these above libraries. Some of the tag examples of each library are as follows:

JSF core tags:
1. <f:attribute></f:attribute> tag is used to add attribute to the UIComponent.
2. <f:actionListener></f:actionListener> tag is used for registering ActionListener instance on the UIComponent.

JSF html tags:
1. <h:commandButton></h:commandButton> tag is used for rendering the html input element.
2. <h:commandLink></h:commandLink> tag is used for rendering the html anchor element <a>.

JSF composite tags:
1. <composite:interface></composite:interface> tag declares composite component's usage contract.
2. <composite:implementaion></composite:implementation> tag is used for defining the composite component's implementation.

JSF uitags :
1. <ui:component></ui:component> tag is used for creating a component.
2. <ui:define></ui:define> tag is used to define content.

JSTL core :
1. <c:forEach></c:forEach> tag is used for iterating functionality.
2. <c:if></c:if> tag is used for evaluating the conditional statement when if is 'true' etc.

JSTL functions:

1. { fn:toUpperCase() }tag is used to change a string written in lower case to upper case.
2. { fn:contains() } tag is used to test whether the string that is inputted is contained a substring or not. etc

EXERCISE

PART I: SHORT QUESTIONS

21. What is JFS?
22. Write any two benefits of JFS?
23. How JSF is used with JSP?
24. What is Uri?
25. Write name of phases in JFS life cycle?
26. What is JFS tag library?
27. Give names of JFS libraries?
28. Describe attribute tag in JFS?

PART II: LONG QUESTIONS

9. Briefly escribe using JFS with JSP.
10. Write a note on JFS Life Cycle.

MULTIPLE CHOICE QUESTIONS

1. JSF stands for
a. Java Standard Faces
b. Java Server Faces
c. Java Server Facelet
d. None
2. ___ is just a string that uniquely identifies a specific library.
a. url
b. uri
c. ulr
d. uir
3. There are _____ phases in JFS Lifecycle
a. 5
b. 4
c. 6
d. 8
4. The objective of ______________ phase is made for each component to retrieve the current state.
a. restore view phase
b. apply request phase
c. Process validation phase
d. render response phase
5. When a user clicks a JSF button, JSF UI component fires a/an _____, which will be handled by the application code.
a. input
b. event
c. bullet
d. None
6. Which one of the following is not a JFS tag library
a. SF Core
b. JSF html
c. JSF Composite
d. taglib

7. prefix for jsf core library is______

a. ui
b. c
c. f
d. h
8. __________tag is used to add attribute to the UIComponent.
a. <th> tag
b. <f:actionListener>
c. <f:attribute>
d. None
9. _____________tag is used for rendering the html input element.
a. <f:actionListener>
b. <h:commandButton>
c. <nav> tag
d. none
10. prefix used for JSF html library is___________
a. ui
b. c
c. f
d. h

MCQ's Answers:

1	2	3	4	5	6	7	8	9	10
B	B	C	B	B	D	C	C	B	D

References

eventhandling.php. (n.d.). Retrieved from http://javatechnologycenter.com: http://javatechnologycenter.com/question/jsf_tutorial/eventhandling.php

JSF with JSP. (n.d.). Retrieved from https://www.oreilly.com: https://www.oreilly.com/library/view/javaserver-faces/0596005393/ch01s03s01.html

jsf-example. (n.d.). Retrieved from https://www.javatpoint.com: https://www.javatpoint.com/jsf-example

jsf-life-cycle. (n.d.). Retrieved from https://www.educba.com: https://www.educba.com/jsf-life-cycle/

jsf-overview. (n.d.). Retrieved from https://www.tutorialspoint.com: https://www.tutorialspoint.com/jsf/jsf_overview.htm

Standard JSF Tags. (n.d.). Retrieved from https://www.informit.com: https://www.informit.com/articles/article.aspx?p=1606899

Using JSF Components in a JSP Page. (n.d.). Retrieved from https://www.oreilly.com: https://www.oreilly.com/library/view/javaserver-faces/0596005393/ch06s01s01.html

what-is-jsf. (n.d.). Retrieved from https://www.javatpoint.com: https://www.javatpoint.com/what-is-jsf