Web Component Development Using Java

Session: 1

Introduction to Web Applications

WWW

Objectives

❖ Explain the purpose of different applications

❖ Explain Web applications and their advantages

❖ Describe the architecture and components of Web application

❖ Describe the role of HTTP protocol and its methods used for accessing Web pages

❖ Explain the use of Common Gateway Interface (CGI) language

❖ Explain the different types of components used in developing Web application

❖ Explain the advantages and disadvantages of Servlets

❖ Explain the services provided by a Web container

❖ Describe the life cycle and directory structure of a Web application

❖ Explain how to package Web application

❖ Develop a Web application in NetBeans Integrated Development Environment (IDE)

❖ An application is a collection of programs designed to perform a particular task.

❖ **Desktop Application**

 ❑ Runs only in a local machine.

 ❑ Can neither be viewed nor be controlled by other users.

 ❑ Example: Microsoft Word.

 ❑ Figure depicts the desktop application.
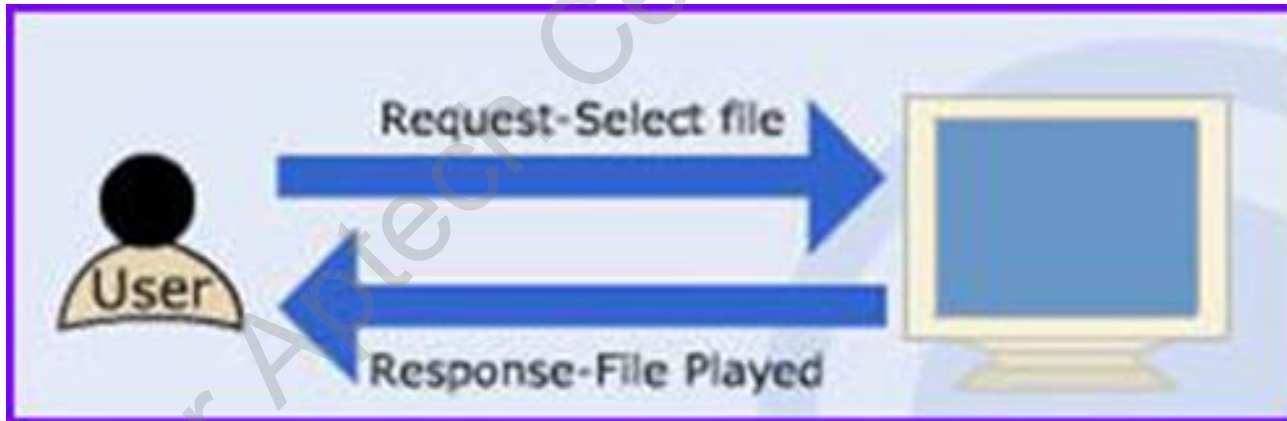
❖ **Networking Application**

❑ Runs in a Local Area Network (LAN), Metropolitan Area Network (MAN), or World Area Network (WAN).

❑ Can be viewed and controlled by users in that particular network only.

❑ Example: Novell Netware.

❑ Figure depicts the networking application.
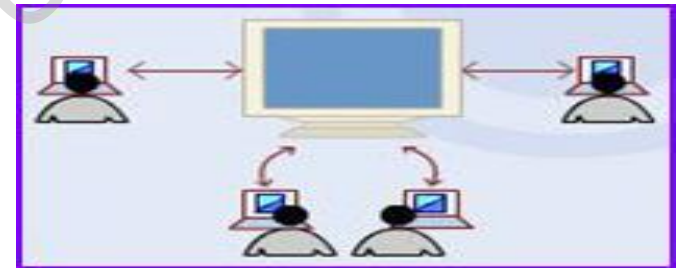
❖ **Web Application**

❑ Runs at a remote location.

❑ Can be viewed and controlled by all the users having administrative or equivalent privilege, throughout the globe, at any instance of time.

❑ Example: Internet mail services, such as `www.yahoo.com`.

❑ Figure depicts the Web application.
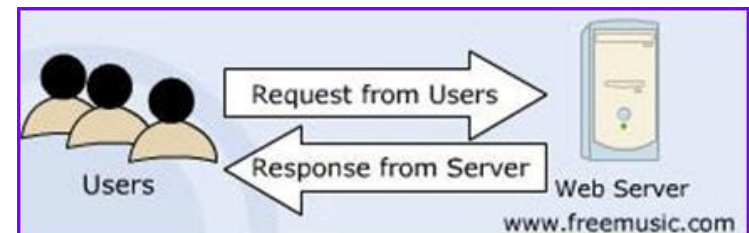
Web Applications 1-2

❖ A Web application:

  ❑ Is a software application that runs on a Web server.

  ❑ Consists of Web pages which can be static and dynamic.

❖ Static Web pages:

  ❑ Are created using Web technologies such as HyperText Markup Language (HTML), Cascading Style Sheet (CSS), and JavaScript.

❖ Dynamic Web pages:

  ❑ Server-side programs are used to achieve dynamic functionalities on Web pages.

  ❑ Provide interactivity with the users.

Users accessing the Web pages on the World Wide Web (WWW) are referred to as Web clients.

Web Applications 2-2

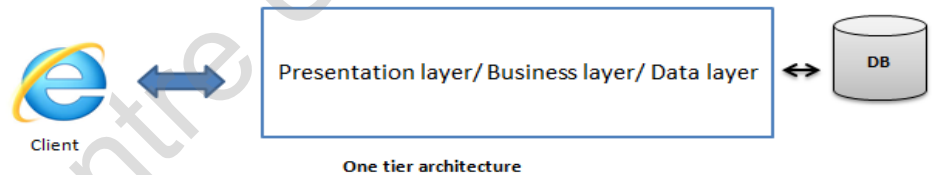❖ The advantages of Web applications over desktop applications are as follows:

  ❑ Easier access to information

  ❑ Lower maintenance and deployment costs

  ❑ Platform independence

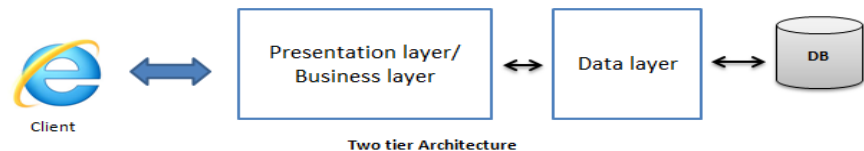 ❑ Wider visibility

Web Application Architecture 1-2

❖ **One-tier architecture**

  ❑ Code related to presentation, business, and data access logic are all clubbed together.

 ❑ Figure shows the one-tier architecture.

❖ **Two-tier architecture**

  ❑ Code related to data access logic is separated from the other two components.

  ❑ Any interaction with data tier will be done through business tier.

  ❑ Figure shows the two-tier architecture.

❖ **Three-tier architecture**

- ❑ Code related to all three components is separated from each other.
- ❑ Business tier acts as an interface between the data tier and the presentation tier.
- ❑ Figure shows the three-tier architecture.
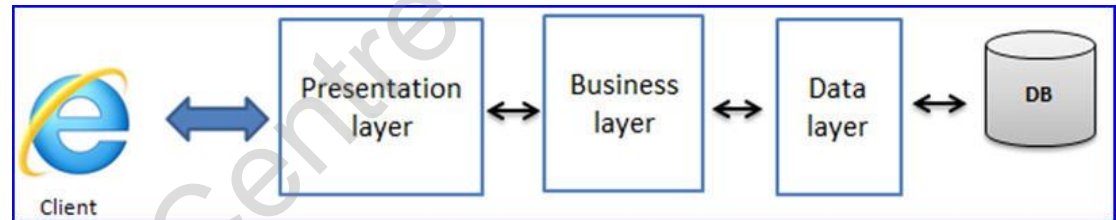
❖ **N-tier architecture**

- ❑ Layers are further subdivided for ease of functioning.
- ❑ Presentation layer is a graphical user interface that displays data to users.
- ❑ Business layer and application layer are separated reducing the number of locations implementing the logic.
- ❑ Figure shows the N-tier architecture.
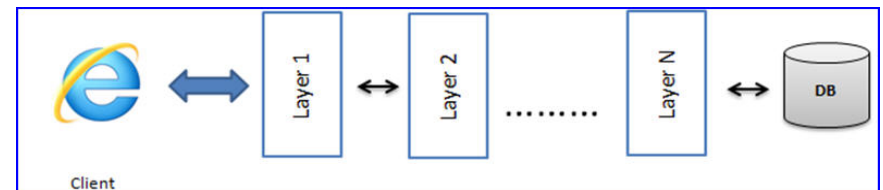
Web Application Technology - HTML

❖ HTML is a presentation language which enables Web designer to create visually attractive Web pages. These pages are stored on Web.

❖ HTML allows Web designers to add images, create forms, and embed media objects on the Web pages. All these are referred as Web resources which are stored on the Web server along with HTML pages.

❖ Figure shows the Web page designed using HTML.

```html
<html>
  <body>

    <h1> Sample HTML Page </h1>

<!-- Displaying an image -->
    An image:
    <img src="HTMLLogo.gif" alt="HTML" width="42" height="42"></p>

    <p>
        This Web page is created in <b> HTML </b> language which has following
        features: </P>

<!-- Displaying a list -->
<ol>
    <li> Stands for Hyper Text Markup Language </li>
    <li> Is a markup language </li>
    <li> Contain HTML tags and plain text </li>
    <li> Also called as web pages </li>

</ol>

  </body>
</html>
```

Web Application Technology - HTTP

❖ The requests and responses sent to a Web application from one computer to another computer are sent using HTTP.

❖ An HTTP client, such as a Web browser opens a connection and sends a request message to an HTTP server asking for a resource.

❖ The server, in turn, returns a response message with the requested resource. Once the resource is delivered, the server closes the connection.

❖ Referred to as a stateless protocol as HTTP does not store any connection information about the page.

❖ Figure depicts HTTP request and response to a Web application.
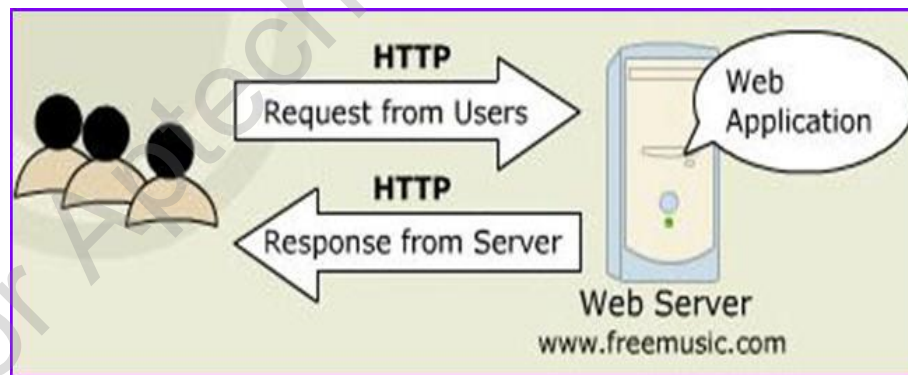
HTTP Request 1-2

❖ The request message sent by the client is a stream of text.

❖ Figure depicts the request message structure.
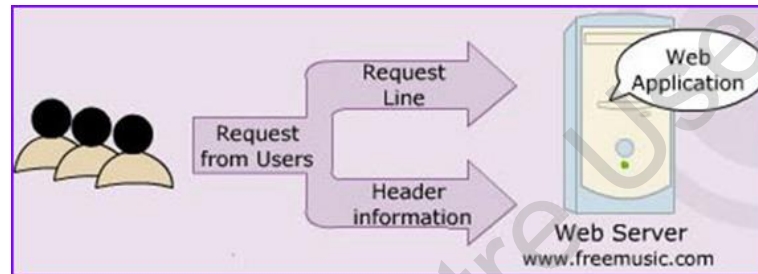
❖ Its elements are namely, Request line and Header information.

❖ **Request line**

 ❑ Contains the HTTP method, the address of the Web page or document referred to as Uniform Resource Locator (URL), and the HTTP version.

 ❑ Figure shows the request line of the request message.
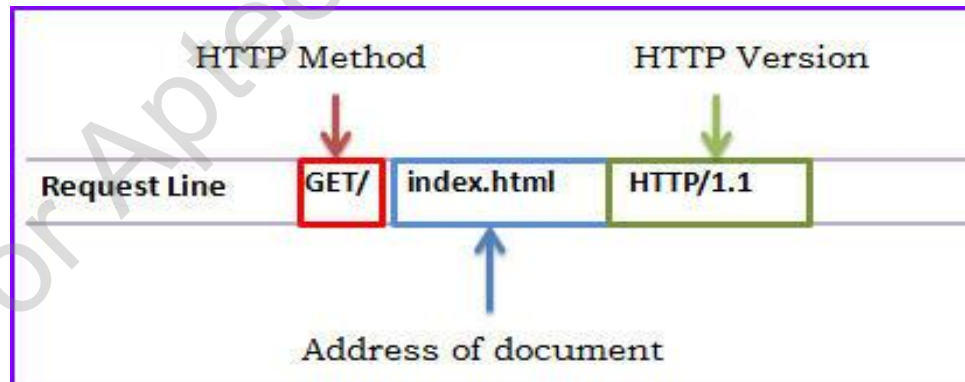
HTTP Request 2-2

❖ **Header information**

- ❑ Specifies the `User-Agent` along with the `Accept` header.
- ❑ The `User-Agent` element header indicates the browser used by the client.
- ❑ The `Accept` header element provides information on what media types the client can accept.
- ❑ After the header, the client sends a blank line indicating the end of request message.
- ❑ Figure shows the sample header information sent in the request message.

```
User-Agent: Mozilla/4.0 (compatible; MSIE 4.0; Windows 95)
Accept: image/gif, image/jpeg, text/*, */*
```

HTTP Response 1-2

❖ The Web application processes the request sent by a client and generates a response message for the requesting client.

❖ Figure depicts the response message structure.
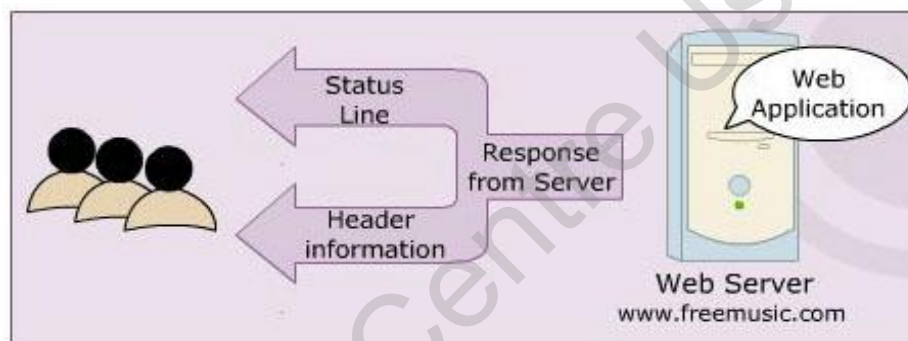
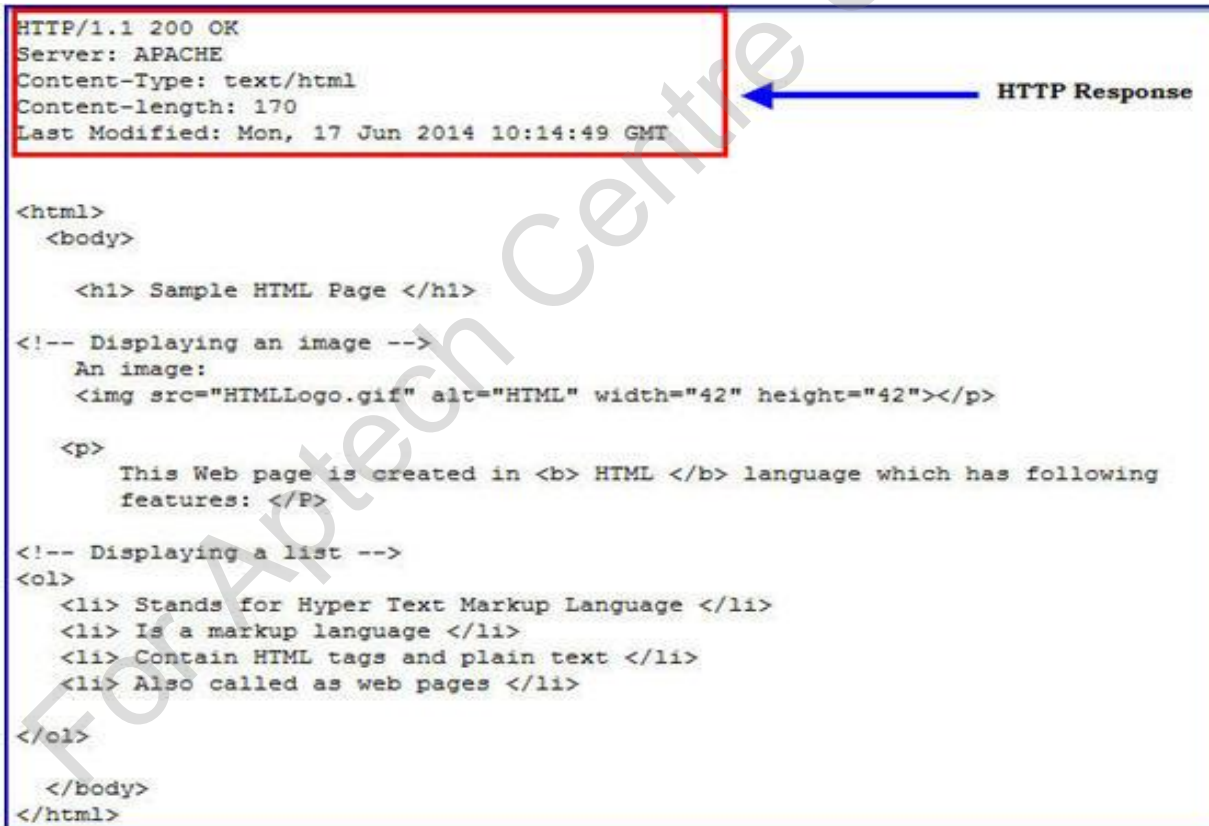Its elements are namely, Status line and Header information.

❖ **Status line**

  ❑ Indicates status of the requested process.

❖ **Header information**

- ❑ Contains information such as server, last modified date, content-length, and content-type.

- ❑ Figure shows the sample message format sent in the response with the HTML content from the Web server.

```
HTTP/1.1 200 OK
Server: APACHE
Content-Type: text/html
Content-length: 170
Last Modified: Mon, 17 Jun 2014 10:14:49 GMT                    ←──────  HTTP Response

<html>
  <body>

    <h1> Sample HTML Page </h1>

<!-- Displaying an image -->
    An image:
    <img src="HTMLLogo.gif" alt="HTML" width="42" height="42"></p>

    <p>
        This Web page is created in <b> HTML </b> language which has following
        features: </P>

<!-- Displaying a list -->
<ol>
    <li> Stands for Hyper Text Markup Language </li>
    <li> Is a markup language </li>
    <li> Contain HTML tags and plain text </li>
    <li> Also called as web pages </li>

</ol>

  </body>
</html>
```

HTTP Methods 1-4

❖ Web applications allow users to enter information using forms and send to the server for processing.

❖ The data entered into the fields in a form is sent to the Web server through URL.

❖ The data is processed on the Web server and the appropriate response is generated which is sent back to the user.

❖ Figure shows an example of request parameters.

The HTTP request messages commonly use the following HTTP methods for transmitting request data over the Web:

❖ **GET**

- ❑ Informs the server to retrieve information from the request URL.

- ❑ Information is passed as a sequence of characters that are appended at the end of the request URL, which forms a query string.

- ❑ The length of query string is restricted to 240 to 255 characters depending on the server.

- ❑ A sample query string constructed while sending user data in the request URL:
`http://www.abcBank.co.uk/search?name=john&pass=j7652`

❖ **HEAD**

- ❑ Functionally same as `GET`, except that the client uses `HEAD` method to receive only the headers of the response and not the message body.

- ❑ Advantageous when the user want to check characteristics of a resource without actually downloading the complete resource.

HTTP Methods 3-4

❖ **POST**

❑ Used when sending information, such as credit card numbers or information to be saved in the database.

❑ Data sent is in encrypted format and not visible to the client, and there is no limit on the amount of data being sent.

❑ The data is passed to the server in the body of the HTTP message, and hence the request cannot be bookmarked or emailed.

❑ Figure shows the request structure with HTTP POST method.
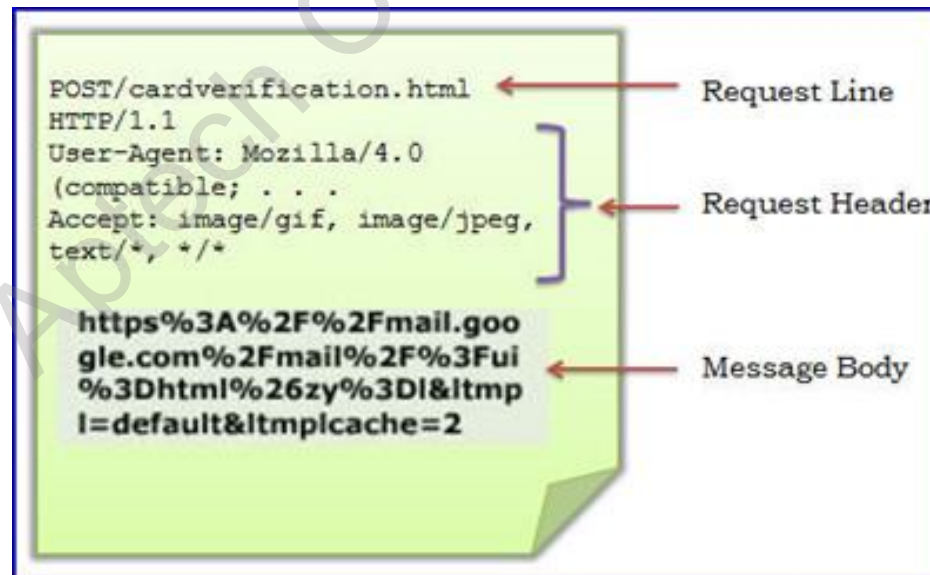
HTTP Methods 4-4

❖ **PUT**

  ❑ Used to request the server to store the data enclosed in the HTTP message body at a location provided in the request URL.

❖ **DELETE**

  ❑ Used to request the server to delete file at a location specified in the request URL.

❖ **OPTIONS**

  ❑ Can be used to query the server on the methods supported for a particular resource available on the server.

❖ **TRACE**

  ❑ Basically used for debugging and testing the request sent to the server.

  ❑ It asks the user to echo back the sent request.

  ❑ Useful when the request sent to the server reaches through the proxies.

Java Web Application

❖ The Web components of Java Web application are as follows:

  ❑ **Servlet**: Java classes that dynamically process HTTP requests and construct responses, a key component in Web development.

  ❑ **JSP**: A text document that creates dynamic Web content, usually to display the content in the Web browser.

❖ Prevalent Server-side Technologies:

  ❑ Common Gateway Interface (CGI)

  ❑ Proprietary Web server API (ISAPI)

  ❑ Server-side JavaScript

  ❑ Active Server Pages

  ❑ PHP

  ❑ Java Servlets

 ❑ Java Server Pages (JSP)

Common Gateway Interface (CGI) 1-2

❖ A set of standards that specify how data is transferred between the Web server and server-side CGI programs.

❖ The data provided by the client in the Web page is sent from HTTP server to the gateway or CGI program.

❖ The CGI program processes the data and returns the result to the Web server, which in turn sends it to the client.

❖ Figure depicts the server process for running CGI.
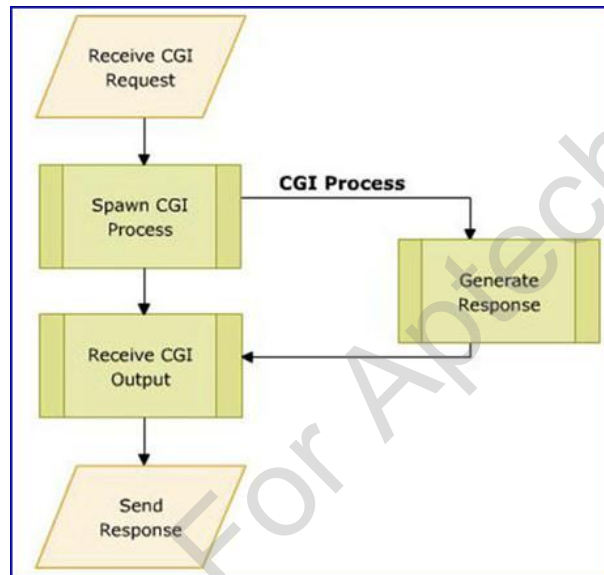
A program that gives dynamic output, thereby executing in real time, is written at the server-side using standards of CGI.

❖ When a Web server receives a request that need to access a CGI program, it creates a new instance or process of the CGI program and then pass the client data to it.

❖ Figure depicts the working of CGI program.
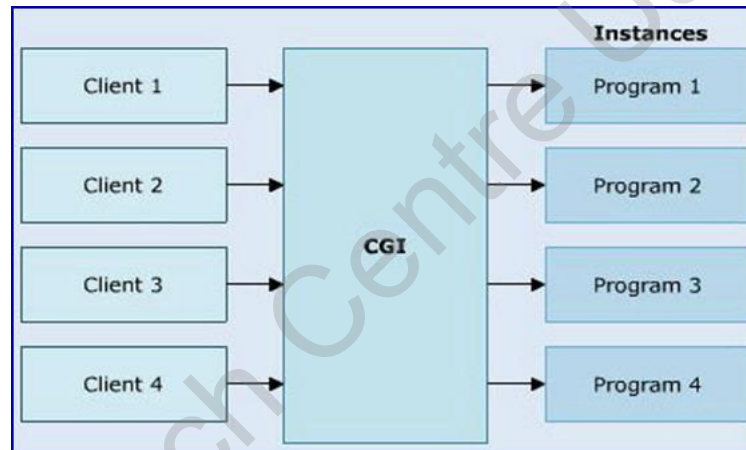
❖ **Disadvantages of CGI:**

- ❑ **Reduced efficiency** - Number of processes that can be executed by a Web server is limited to each server.

- ❑ **Reloading Perl interpreter** - Each time the server receives a request, the Perl interpreter needs to be reloaded. This reduces the efficiency of the server.

Servlets

❖ The processing of Servlet is very similar to CGI program, as it also responds to HTTP request and generates dynamic response.

❖ Servlet:

❑ There is only a single instance of Servlet created on the Web server.

❑ To service multiple clients' request, the Web server creates multiple threads for the same Servlet instance.

❑ Each thread handles request received from the client and generates response that is sent to the Web engine, which in turn sends the response to the client.

❖ Figure shows the working of the Servlet.
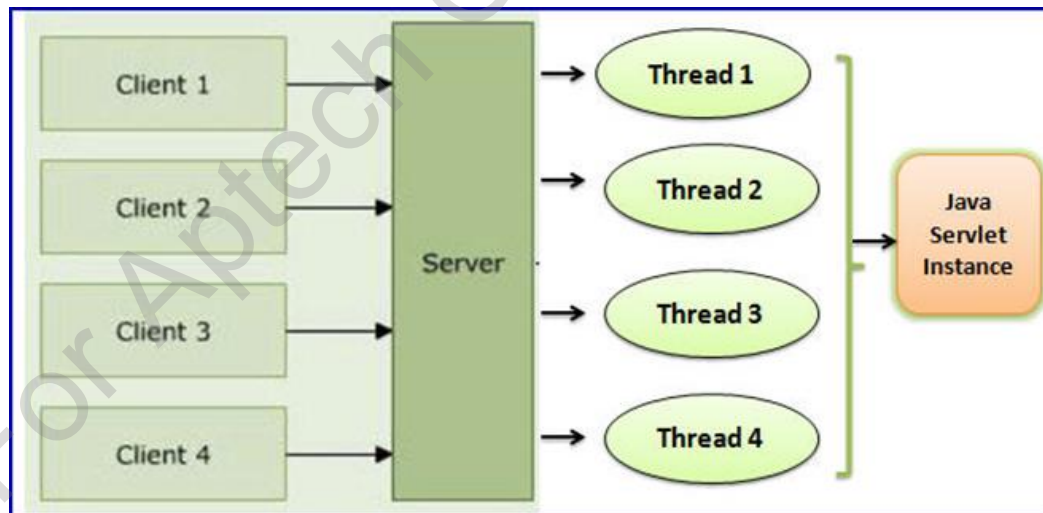
Merits of Servlet

❖ Servlets are written using Java language, which makes extensive use of Java API. The merits of servlets are:

❑ **Enhanced efficiency** - Servlet is required to be executed only once at the beginning with the initialization code. Thereafter, it gets auto refreshed each time a request is sent for execution.

❑ **Ease of use** - Servlets does not have too many complexities. It is just basic Java along with HTML implemented within the Java code, which increases the ease of use.

❑ **Powerful** - The usage of Java APIs makes the Servlets a powerful server-side scripting language.

❑ **Portable** - Java is platform independent and since Servlet uses Java code for writing scripts, it can be executed in any platform.

❑ **Safe and cheap** - Usage of Java codes provides high security of data to be sent and received, thereby maintaining the safety.

❖ Java Web application comprises servlets, JSP pages, images, HTML files, JavaBeans, Applets, and Java classes.

❖ Package all the files associated with Web application into a single Web archive file (war).

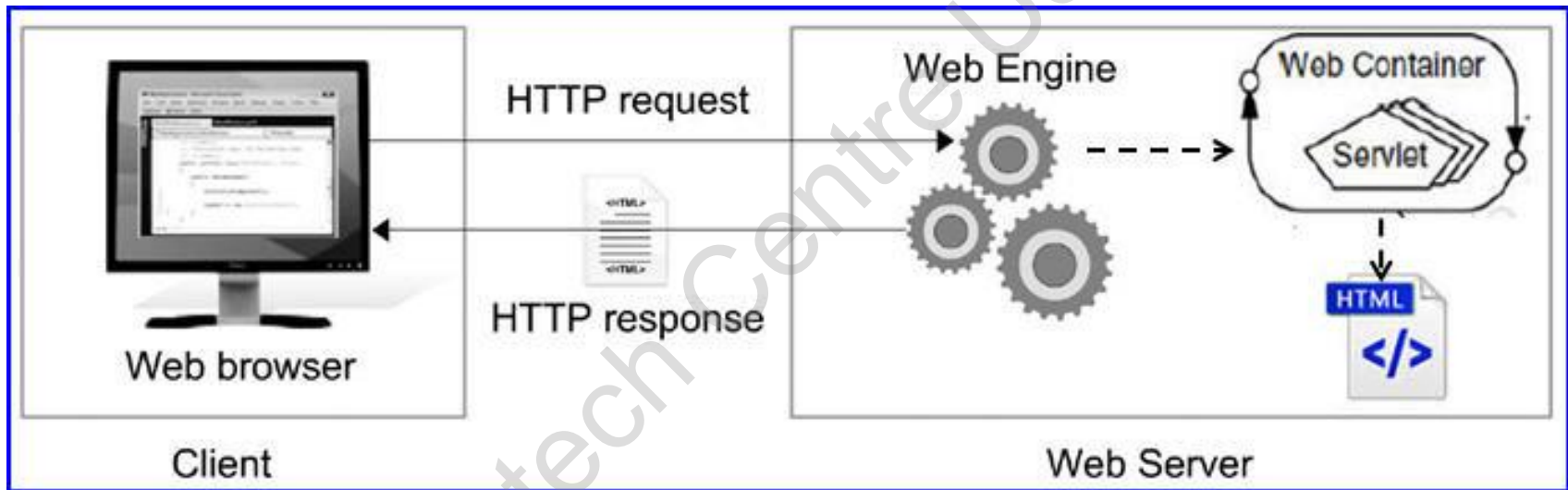❖ The Java Web applications are deployed on a Java-enabled Web server such as Tomcat.

❖ Examples of other Java servers containing a Web container for managing Web components include:

❖ Figure shows a Web server containing a Web container used for processing the HTTP request and HTTP response for the accessed Servlet or JSP.

❖ Every Java Web server contains a Web container which is responsible for managing the Web components on the server.

Web Container

❖ Manages execution of Servlets and JSP pages.

❖ Takes request from a Web server and passes it to a servlet for processing.

❖ Manages the servlet life cycle and other services such as security, transaction, and so on for the Web components.

❖ Referred as Servlet container.

❖ Figure depicts a Web container.
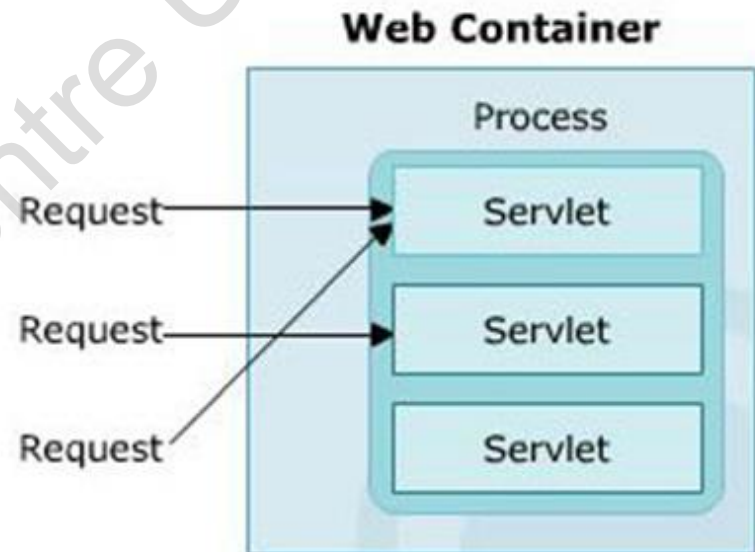
❖ The performance of a servlet depends upon the efficiency of the Web container.

Web Application Life Cycle 1-2

- ❖ Develop the Web component code.

- ❖ Develop the Web application deployment descriptor.

- ❖ Compile the Web application components and helper classes referenced by the components.

- ❖ Optionally, package the application into a deployable unit.

- ❖ Deploy the application into a Web container.

- ❖ Access a URL which references the Web application.

❖ Figure describes the Web application life cycle.
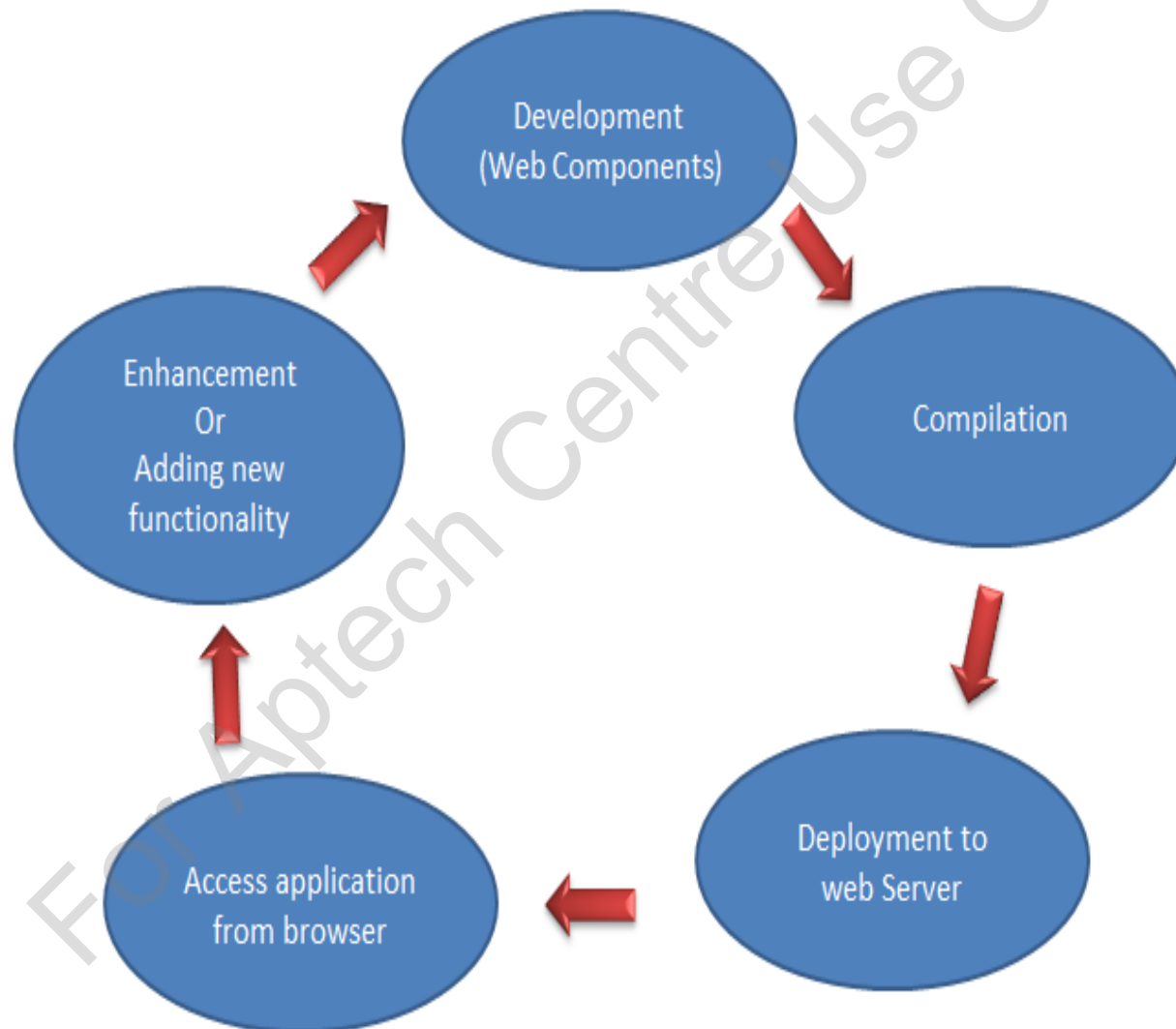
Packaging Web Applications

❖ A Web application is composed of Web pages and Web components which are collectively referred as Web resources.

❖ The Web applications also need to be packaged with Web resources before the application is deployed on the Web server.

❖ The package file used for a Web application has a specific set of directory structure:

 ❑ That is packaged in the `.war` file.

 ❑ Helps the Web container to ensure proper functionality of the application.

Web Application Context Root

❖ Each Web application is identified or assigned with a **context root**.

❖ The context root or context path of the Web application:

  ❑ Is the base URL used for locating Web pages and Web components such as Servlet and JSP within the application.

❖ Example:

  ❑ Suppose the application is deployed on the local Web server.

  ❑ Then, the URL to access the `index.html` page will be:
    `http://localhost:8080/WebApp/page/index.html`.

  ❑ Thus, the context root of the deployed application is the `/WebApp`.

❖ The context root of the Web application contains two main components in the directory structure:

❑ **Static files** - All the HTML pages and images comprise the static files. Images can be collectively stored in an images directory.

❑ **WEB-INF** - Exists within the context root, cannot be referenced.

❖ Figure shows the directory structure of the `.war` file.
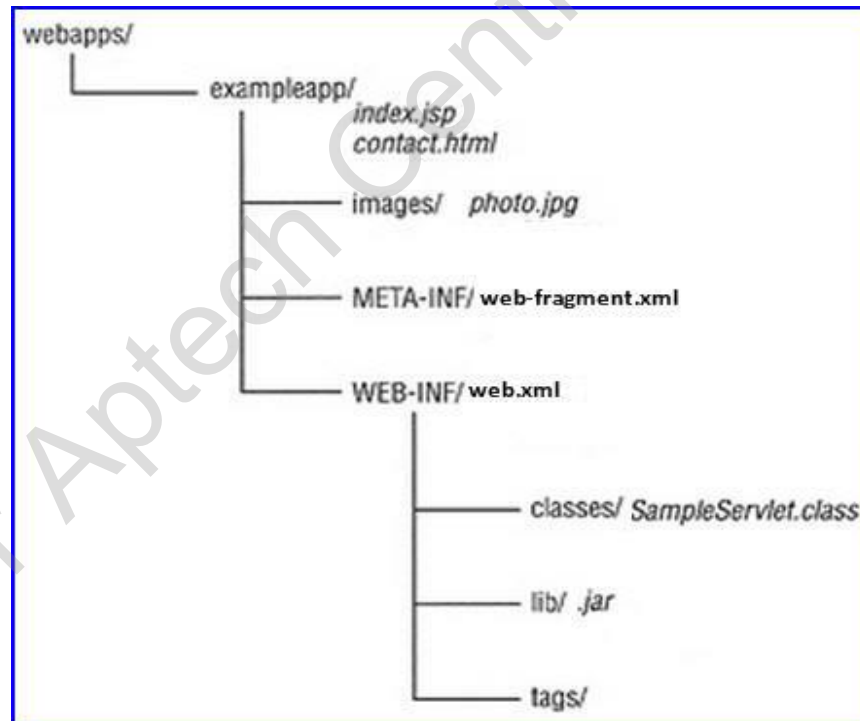
WEB-INF Directory 1-2

The contents of `WEB-INF` directory are as follows:

❖ **classes directory**

  ❑ Servlet classes, JavaBean classes, and any other classes required by the Web applications are stored in this directory.

  ❑ Example: If a servlet or JavaBean class is part of a package such as `www.bookstore.com`, then the corresponding class file will be placed in `classes/com/bookstore` directory.

❖ **web.xml**

  ❑ Basically, it is an XML structured file which provides configuration information for the components of the Web application.

  ❑ Called as the deployment descriptor of the Web application.

  ❑ Includes information such as the default pages to show mapping Servlets with their URLs, and so on.

❖ **lib directory**

- ❑ Library JAR files used by the application are stored within the `WEB-INF/lib` directory.

- ❑ Example: Database driver files.

❖ **tags directory**

- ❑ Contains tag files, which provide implementation for custom tags.

❖ **tag library descriptor (tld) files**

- ❑ Provide information about the attributes of a custom tag and the class to be invoked when the tag comes across in a JSP page.

- ❑ Files have `.tld` extension.

❖ A configuration file which describes how the Web application should be deployed.

❖ Written using XML with name `web.xml` and placed under the **WEB-INF** folder, `WEB-INF/web.xml`.

❖ When the Web server receives a request for the application, it uses the deployment descriptor to map the URL of the request to the Servlet that handles the request.

❖ **Example**: To map a URL request to a Servlet, the following settings must be done in the `web.xml` file:

  ❑ Declare the servlet with the `<servlet>` element.

  ❑ Define a mapping from a URL path to a Servlet declaration with the `<servlet-mapping>` element.

Deployment Descriptor 2-4

❖ The code snippet shows the configuration setting for the `HelloServlet` class created in the Web application.

```xml
<!-- web.xml -->
<?xml version="1.0" encoding="UTF-8"?>
. . .
 <servlet>

   <servlet-name>HelloWorldServlet</servlet-name>
   <servlet-class>mypkg.HelloServlet</servlet-class>

 </servlet>


 <servlet-mapping>

   <servlet-name>HelloWorldServlet</servlet-name>
   <url-pattern>/sayhello</url-pattern>

 </servlet-mapping>
</web-app>
```
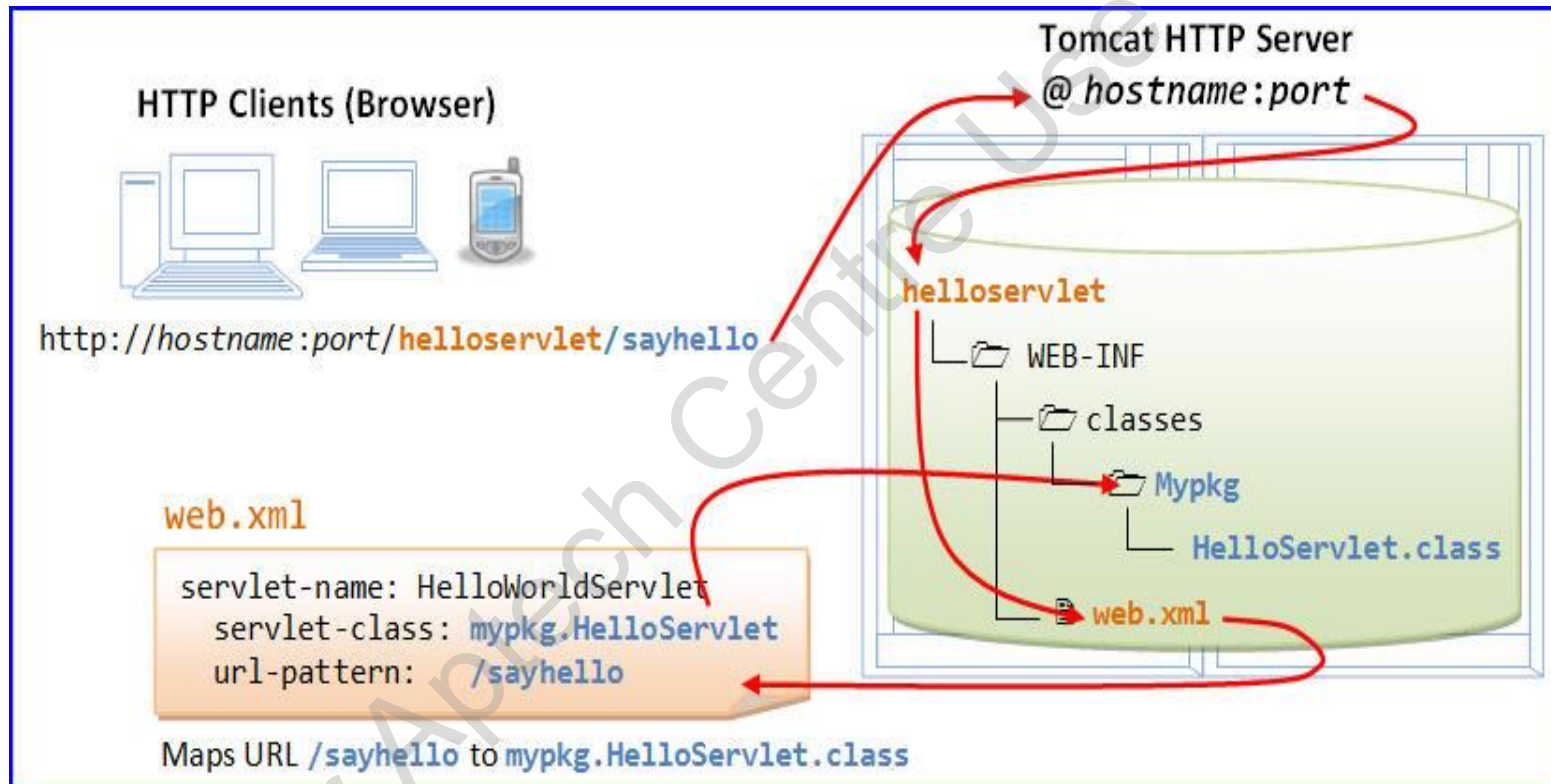
❖ Figure shows the mapping of Servlet with the URL through which it is accessible to the application.

Deployment Descriptor 4-4

❖ Servlet 3.0 has also introduced a new feature called **Web fragments**.

❖ Web fragment is the mechanism that:

❑ Include deployment descriptor information which is specific to a library.

❑ Helps to segregate the unrelated information from the main deployment information.

❑ Is stored in the folder `META-INF\web-fragment.xml` of the Web application directory structure.

Developing Web Applications in NetBeans IDE

❖ NetBeans Integrated Development Environment (IDE) is used to create Java EE based projects with JSP's and Servlets.

❖ Some of the features of the NetBeans IDE are as follows:

   ❑ A source code editor which is similar to an HTML text editor.

   ❑ A compiler compiles the source code into an executable program and an interpreter runs programs and scripts that do not need to be compiled.

   ❑ Build automation tools help automate the processes that need to happen with most software developments such as debugging, compiling, and deployment.

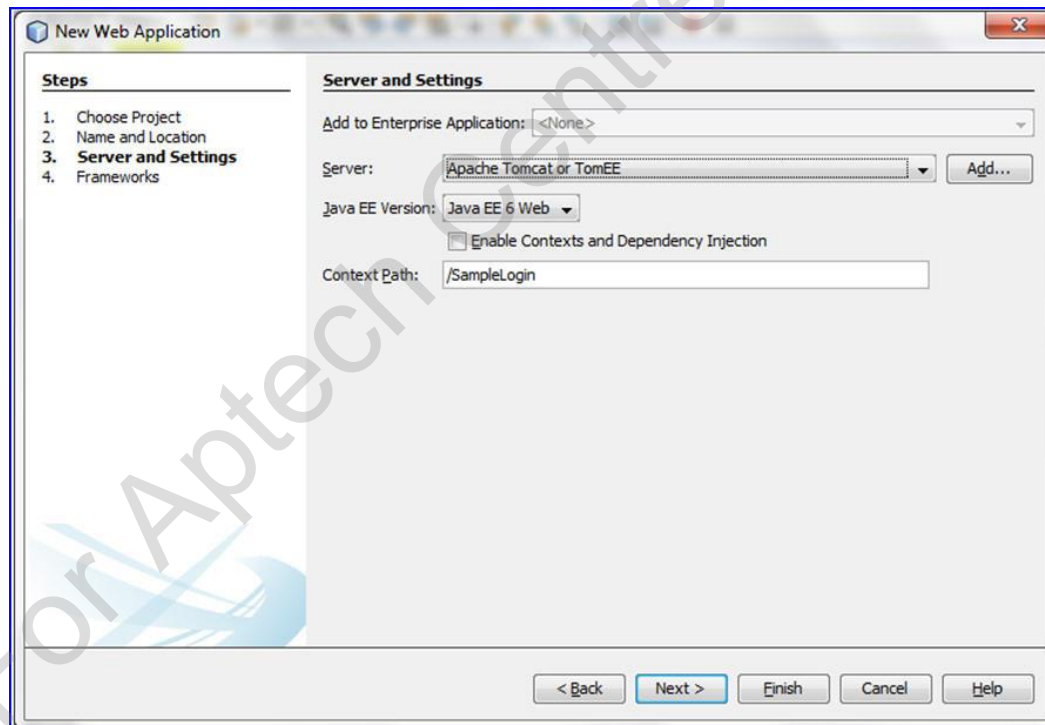   ❑ A debugger helps pin-point the exact spot of a problem or error in the source code.

Creating a New Web Project in NetBeans IDE

❖ Open NetBeans IDE and select **New Project** from the **File** menu.

❖ Select **Java Web** from the **Categories** list.

❖ Select **Web Application** from the **Projects** list.

❖ Click **Next** to specify the name and location.

❖ Figure shows how to create a project.
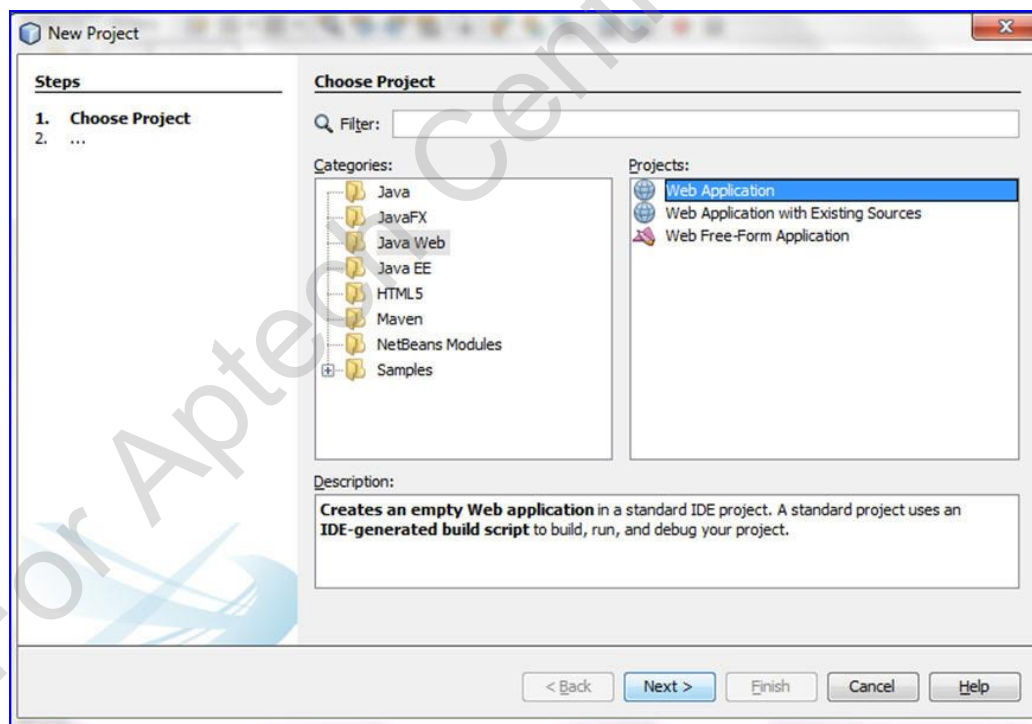
Specifying Name and Location for the Project 1-2

❖ Enter the name of the project, for example, **FirstWebApp.**

❖ Click the **Browse** button and select a location for saving the project.

❖ Click **Next** to specify the server used for deploying Web application.

❖ Click **Server** drop-down list and select the Web server for the application as shown in the figure.

❖ Click **Next** and select frameworks used in Web application. Currently, the Web application is not using any framework.

❖ Click **Finish**.

❖ Figure shows the structure of the **FirstWebApp** project created in NetBeans IDE.
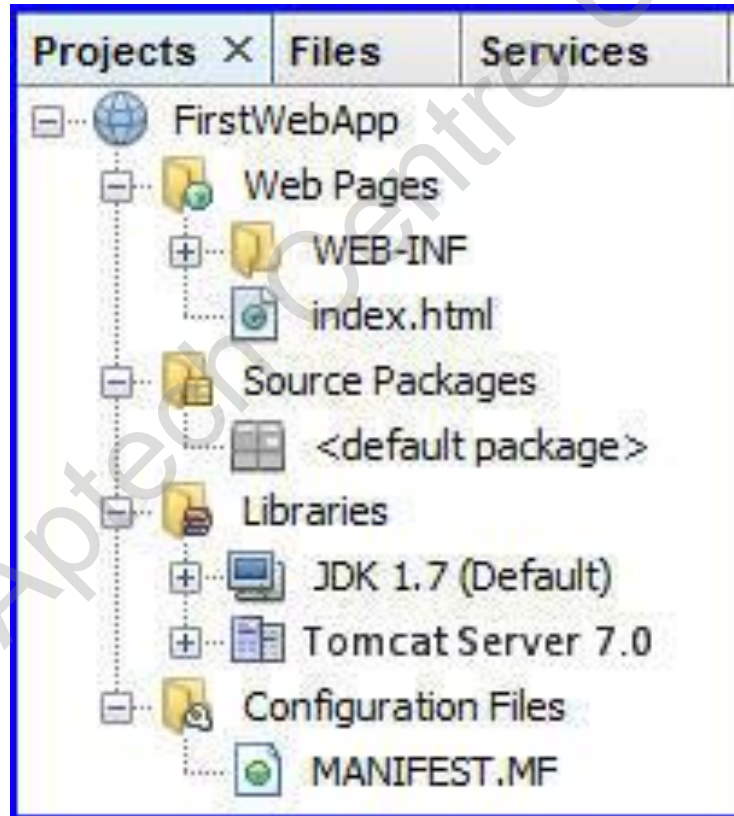
Writing the Code for Servlet 1-3

❖ Right-click the **Source Packages** and select **Servlet** from the context menu. This opens **New Servlet** window.

❖ Type the name of the Servlet as `FirstServlet` and package as `com.controller` as shown in the following figure.

❑ Click the **Finish** button.

Writing the Code for Servlet 2-3

❖ The code snippet shows the code generated for the 'FirstServlet' class.

```
import javax.servlet.http.*;
public class FirstServlet extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet FirstServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet FirstServlet at " + request.
getContextPath() + "</h1>");
        out.println("</body>");
        out.println("</html>");
}
. . .
```

❖ The out object of PrintWriter class is created to display the message on the Web page in response.

```
protected void doGet(HttpServletRequest request, HtttpServletResponse
response)throws ServletException, IOException {

processRequest(request, response);

}

. . .

protected void doPost(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException {

processRequest(request, response);

}

}
```

❖ The `processRequest()` method in NetBeans IDE contains the default generated Servlet code.

❖ This method is invoked by `doGet()`/`doPost()` method based on how the data is sent by the client in HTTP request.

❖ Change the `out.println("<h1>Servlet FirstServlet at " + request. getContextPath() + "</h1>");` with this **`out.println("<h1> Hello World </ h1>");`**

Creating the Deployment Descriptor

❖ The code snippet shows the `web.xml` file that is updated with the configuration settings of the `FirstServlet`.

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

<servlet>

<servlet-name>FirstServlet</servlet-name>

<servlet-class>com.controller.FirstServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>FirstServlet</servlet-name>

<url-pattern>/myservlet</url-pattern>

</servlet-mapping>

</web-app>
```

Build and Execute the Web Application

❖ To run the application, right-click the project and select **Run** from the context menu.

❖ When you run the application in NetBeans IDE:

  ❑ It first compiles the Servlet code and then deploys the resource to the Web server.

  ❑ If Web server is not running, first start the server and then deploy the Web application to server.

  ❑ After building and deployment is completed successfully, NetBeans will launch the Servlet page in the browser as shown in figure.

Summary

❖ An application is a collection of programs designed to perform a particular task. Different types of applications are designed for different purposes.

❖ A Web application is a software application that runs on a Web server. The Web application basically has three components: the presentation layer, the application layer, also known as business layer, and the data access layer.

❖ The most common technologies for communication on the Web are HTML and HTTP. HTML is a presentation language which enables Web designer to create visually attractive Web pages.

❖ The requests and responses sent to a Web application from one computer to another computer are sent using HTTP. HTTP does not store any connection information about the page and hence, it is referred to as a stateless protocol.

❖ The HTTP request messages uses the HTTP methods for transmitting request data over the Web.

❖ Java Web application comprises servlets, JSP pages, images, HTML files, JavaBeans, Applets, and Java classes.

❖ The Web applications are packaged in the .war file which allows the content of the Web application are accessed from the application's context root.

❖ A deployment descriptor describes how the Web application should be deployed. It is written using XML with name web.xml and placed under the WEB-INF folder, that is, WEB-INF/web.xml.

❖ Java Web applications are developed in NetBeans IDE which provides fully-integrated environment for building and executing the Web applications.