FINALS

SCOPE

**SERVER-SIDE WEB SCRIPTING**

Server-side scripts are technique used by back-end web developers for web development which uses scripts on the web server side and make responses to client’s side depending on the request to the web server on the website. These scripts can be written in different server-side programming languages, such as PHP, Python, C#, C++, Java and Erlang. These languages are used for the communication of the server, client and database.

Server-side scripting is used in many ways such as: for customizing interface for the client; hide source codes; Facilitates the transferring of data into the browser; controls on what to share on applications.

* JAVA (SERVLET AND JSP)
* PHP
* Node.js
* ASP.NET (if there’s still time)

**Java Programming Language Platforms:**

Java Platform, Standard Edition (Java SE)

Java Platform, Micro Edition (Java ME)

Java Platform, Enterprise Edition (Java EE)

* It used in larger scale
* Used in Development of Server
* Used for Java Web

Java EE versions:

Java EE 7 Web

Java EE 6 Web

Java EE 8 Web

**Web server/ Application Server:**

Java Web Application

In Netbeans in creating a web services you can either

* Apache Tomcat 8.0.27.0
* Glassfish server 4.1.1

Web Pages (folder)

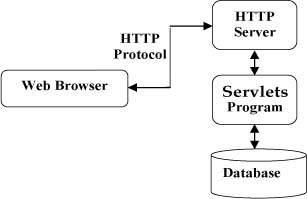
* META-INF
* WEB-INF
* images

**JAVA Servlet**

**What is Java Servlet?**

It is used for building Web-based application which provides a component-based and platform-based method without the performance limitations of CGI programs. It has access to all Java API including the JDBC API to access enterprise database.

It acts as a middle layer between a request coming from a Web browser or other HTTP client and databases or even an application on the HTTP server.

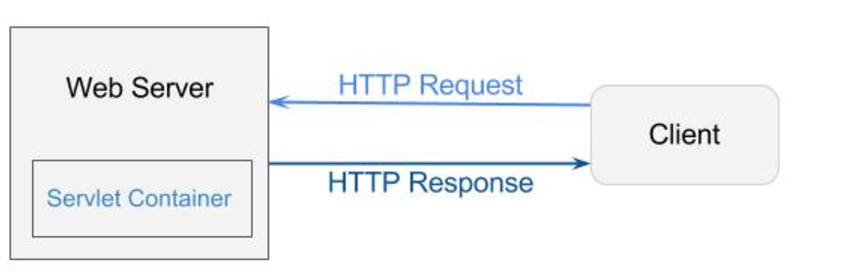
**SERVLET ARCHITECTURE:**

T. (n.d.). Servlets - Environment Setup. Retrieved from

https://www.tutorialspoint.com/servlets/servlets-environment-setup.htm

**SERVLET CONTAINER**

It provides the runtime environment for JavaEE application. It is used for dynamically generate web pages in server side. It part of the server side that interacts with the servlet for handling the dynamic web pages form the client.

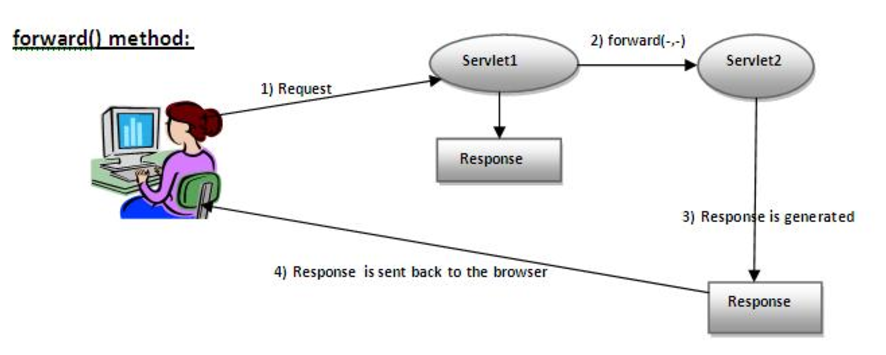


**REQUEST DISPATCH**

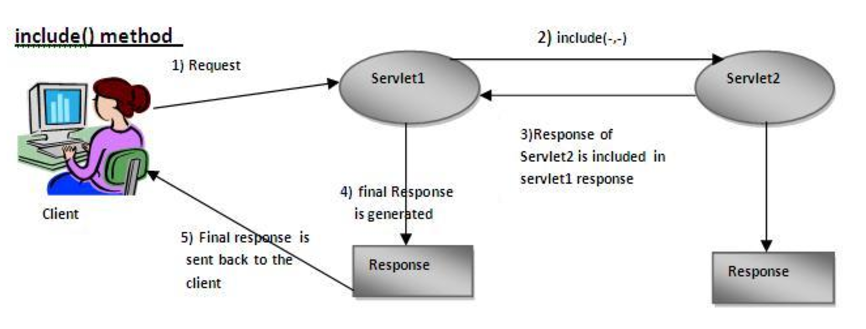
It is used for dispatching request to another resource in HTML, servlet or JSP.

It provides two methods:

* **public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.



* **public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Includes the content of a resource (servlet, JSP page, or HTML file) in the response.



RequestDispatcher in Servlet - javatpoint. (n.d.). Retrieved from

https://www.javatpoint.com/requestdispatcher-in-servlet

**JSP**

It is used for creating web application just like Servlet. It consists of HTML tags and JSP tags.

* **DIRECTIVES**
  + It is used to tell the web container how to translate a JSP page into the corresponding servlet.
  + *Syntax:* <%@ directive attribute="value" %>
  + *Three types of directives:*
    - page directive
      * provides instruction to the container that pertains to the current JSP page
      * <%@ page attribute = "value" %>
    - include directive
      * tells the container to merge the content of the other external files with the current JSP during the translation phase.
      * <%@ include file = "relative url" >
    - taglib directive
      * declares that your JSP page uses a set of custom tags and identifies location of the library.
      * <%@ taglib uri="uri" prefix = "prefixOfTag" >
* **TEMPLATE TEXT**
* **STANDARD ACTIONS**
  + jsp:include
    - Includes a file at the time the page is requested.
  + jsp:useBean
    - Finds or instantiates a JavaBean.
  + jsp:setProperty
    - Sets the property of a JavaBean.
  + jsp:getProperty
    - Inserts the property of a JavaBean into the output.
  + jsp:forward
    - Forwards the requester to a new page.
  + jsp:plugin
    - Generates browser-specific code that makes an OBJECT or EMBED tag for the Java plugin.
  + jsp:element
    - Defines XML elements dynamically.
  + jsp:attribute
    - Defines dynamically-defined XML element's attribute.
  + jsp:body
    - Defines dynamically-defined XML element's body.
  + jsp:text
    - Used to write template text in JSP pages and documents
* **EL EXPRESSIONS**
  + it helps to easily access application data which stored in JavaBeans components and allows to create expression both arithmetic and logical.
  + <jsp:setProperty name = "box" property = "perimeter" value = "100"/>
* **JSP SCRIPLETS**
  + it can contain java language statement, variable or method declarations or expression that are valid in the page scripting language.
  + <% code fragment %>
* **SCRIPLET SCRIPLET**
* **SCRIPLET DECLARATION**
  + declares one or more variable or methods that can be used in Java code in the JSP FILE.
  + <%! int i = 0; %>
  + <%! int a, b, c; %>
  + <%! Circle a = new Circle(2.0); %>
* **SCRIPLLET EXPRESSION**
  + contains scripting language expression that is evaluated, converted to a string and inserted where the expression appears in the JSP file.
  + <%= expression %>
  + <html>

<head><title>A Comment Test</title></head>

<body>

<p>Today's date: <%= (new java.util.Date()).toLocaleString()%></p>

</body>

</html>

* **JSP COMMENTS**
  + marks text or statement that the JSP container should ignore.
  + <%-- This is JSP comment --%>
  + <html>

<head><title>A Comment Test</title></head>

<body>

<h2>A Test of Comments</h2>

<%-- This comment will not be visible in the page source --%>

</body>

</html>

T. (n.d.). JSP Directives. Retrieved from https://www.tutorialspoint.com/jsp/jsp\_directives.htm

T. (n.d.). JSP Syntax. Retrieved from https://www.tutorialspoint.com/jsp/jsp\_syntax.htm

**SESSION HANDLING**

HTTP is a stateless protocol, it means every time a client retrieves a web page, it opens separate connection to the web server and the server doesn’t keep any record of the client’s request.

**MAINTAING SESSION BETWEEN WEB CLIENT AND SERVER**

* **COOKIES**
  + it is a key value pair information and it sent by the server to the browser.
    - There are two types of cookie
      * Session cookies
        + temporary cookie and deleted as soon as the client closes the browser.
      * Persistent cookies
        + remains on the hard drive and it’s only deleted when it’s deleted or it expired.
* **HIDDEN FIELD**
  + similar to other input fields but these fields are not displayed on the page and  its value is sent as other input fields.
* **URL REWRITING**
  + It is an approach in which a session identifier gets appended with each request URL that helps the server identify the user session.
* **SESSION OBJECT**
  + It is a representation of the user’s session. It is available in all request and attributes stored in HTTP session in will be available in any servlet or in a JSP.

**-** Serving Binary Content

- Handling Request Parameters (GET)

- Request Dispatching

**Java Servlet - JSP**

@WebServelet container

@WebServlet(name = “Hello Servlet”, urlPatterns = (“/HelloServlet”)

getInputStream() – Servlet Input Stream

getOutputStream() - Servlet Output Stream

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException { }

Portimage = request.getPart(“image”) – get the name

**Request Dispatching**

* **God Class –** a class that does a lot of things
* **Long Method –**
* **Divide & Conquer**

set/ get Atribute

request.setAttribute(“productlist”, productList);

request.setAttribute(“productlist”);

**include component** – either can be **static**, or **dynamic**

**Session Handling**

- “Stateless Http”

- extending Features : **Cookies**

-State of correlated transaction

**Extended Features**

**Cookies**

–small textual information, generated from the web application

- stored by the client in the browser (Cookie Jar)

- identifier

set.cookie.ck1

- non-persistent cookies

(as long as the browser is open)

- can set the time

- can set by HTTPS

HTTP – only can be access by javascript

Scoped objects

- context scopes

**PHP**

- does have many framework

- striplets to JSP

- there is a PHP API on Web services

- Switch in and out blocks in PHP

-Object-Oriented (close with Java)

echo- PHP print statement

Array – is an associative structured

<form action = “x.php” method = GET>

<input type = “text” name = “user”

<input type = “submit”>

</form>

isset function

isset ($\_GET[‘user’]

**Variable – SUPERGLOBALS:**

$GLOBALS

$\_SERVER

$\_GET

$\_POST

$\_FILES

$\_COOKIE

$\_SESSION

$\_REQUEST

$\_ENV

Example:

<?php fpreach ($\_SERVER as $Key = $values) {

echo “<p> $key = $ values </p>”;

} ?>

**HTTP Request Headers**

HTTP\_HOST = Localhost

HTTP\_CONNECTION = Keep-alive

HTTP\_CACHE\_CONTROL =max-age = 0

HTTP\_UPGRADE\_INSECURE\_ReQUESTS = 2

HTTP\_USER\_AGENT= Mozilla/5.0

HTTP\_ACCEPT

HTTP\_OUT

REMOTE\_PORT = 50625

GATEWAY\_INTERFACE = CGI 1.1

SERVER\_PROTOCOL = HTTP 11.1

REQUEST\_METHOD\_GET

QUERY\_STRING=

REQUEST\_URI =

SCRIPT\_NAME

PHP\_SELF

**NODE.JS Serverside**

**MEAN**

**M –** Monggo DB

**E –** Express JS

**A –** Angular framework

**N –** Node.JS

**BJON** – Binary JSON Format

API Node.JS Features

index.js(simple webserver)

var http require (‘http’)

var server = http.createServer();

server.on(‘request handler’);

function handle(request, response) {

response end(hello, nodeJs!!);

}

function handler (request, response) {

var method = request.method;

var url = request.url;

var headers = request.headers;

console.log(‘Request Method: ${method}’);

console.log(‘Request URL: ${url}’);

console.log(‘Request Headers: \n ${JSON.String}’);

* **asynchronous handler –** callback handler
* **synchronous handler**

**SECURITY**

**OWASP Top Ten Project**

**Top Ten Security Vulnerability in the Web.**

**(2017)**

1. Injection
2. Broken Authentication and Session Management
3. Cross-Site Scripting (XSS)
4. Broken Access Control
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Insufficient Attack Protection (IAP)
8. Cross-site Request Forgery (CSRF)
9. Using components with known vulnerabilities
10. Under protected API

**Digital Certificate**

- encrypting of the Website

letsencrypt.org – website for free Digital certificate

**HASH Algorithm**

- function

- One-way function

-intractable

ex.

MD5

SHA1

SHA2

\_SALTED

(‘pwd’ + salt) = HASH

src:

https://www.upwork.com/hiring/development/server-side-scripting-back-end-web-development-technology/