ADVANCED JAVA (CBCS)

BCA-V SEM

NOTES

Prepared By

Prof. Prabhu Kichadi, M, Tech
KLES SSMS BCA College Athani

UNIT – IV

CONTENTS

Servlet interaction and advanced servlets, life cycle of servlet, Java Servlet Development Kit, javax.servlet package, reading servlet parameters, Reading initialization parameters, The javax.servlet.http package, Handling HTTP, Java Server Pages(JSP), JSP, JSP tags, Request string, Cookies, User session, Session object.

Client: Client is a machine/software which sends requests to the server and waits for response.
Client is a light weight software.
Clients can be web-clients, DB Clients etc.

Server: Server is dedicated machine/software which receives requests from client and processes it and generates responses to the client.
Server is a heavy weight software.

Servers can be web-server, DB-Server.

Application: An application is a computer software package that performs a specific function for an end user.

Web application/Dynamic web project: It is an application which is stored and run through network/internet.

Web Server: It is dedicated server, which maintains and manages web related resources like html, css, js, jsp and servlets. Resources are also called as web components.

Example: Apache Tomcat server.

Web client: It is a light weight software, which sends requests to the server for web resources and waits for response from web server.

Example: Browsers chrome, firefox, safari etc.

Servlet: Servlet technology is used to create a web application (resides at server side and generates a dynamic web page). Servlet is a server-side scripting language used to develop dynamic web applications.

Features of servlet:

1. Servlet follows client-server architecture.

2. Servlet is a web component that is deployed on the server to create dynamic web pages.

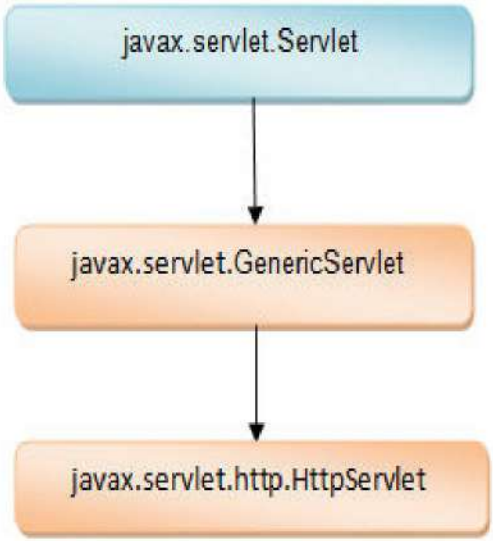
3. Servlet is a child class of **javax.servlet. GenericServlet** OR **javax.servlet.http.HttpServlet.**

4. Servlet is an interface; it must be implemented for creating any servlet.

5. Every servlet class must implement its parent class methods to give service to the clients.

6. Servlet is a web component that is deployed on the server to create a dynamic web page.

Servlet Hierarchy:

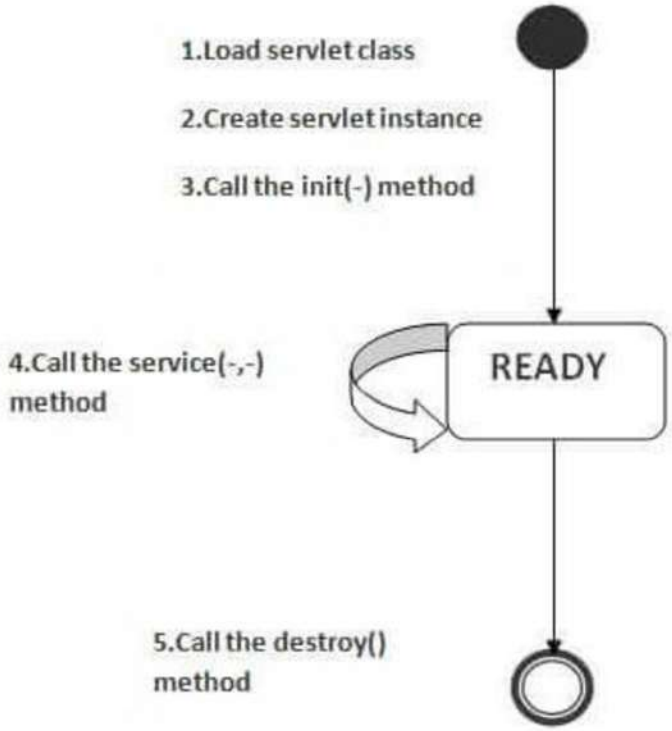


- Every servlet must be a child class of either javax.servlet.GenericServlet or javax.servlet.http.HttpServlet class.
- If you want provide task for a servlet means we must write the code inside service( ) method or doPost( ) & doGet( ) method.
- Whenever web client sends a request to the servlet service( ) method is going to call.

Life Cycle of a Servlet (Servlet Life Cycle)

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.



As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

1) Servlet class is loaded

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

2) Servlet instance is created

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

3) init method is invoked

The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface.
Syntax of the init method is given below:

```
public void init(ServletConfig config) throws ServletException
```

4) service method is invoked

The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

```
public void service(ServletRequest request, ServletResponse respon
se)
throws ServletException, IOException
```

5) destroy method is invoked

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

```
public void destroy( )
```

Reading Initialization Parameters:

Initialization parameters are stored as key value pairs. They are included in web.xml file inside init-param tags of servlet tag. The key is specified using the param-name tags and value is specified using the param-value tags.

web.xml:

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

  <servlet>
    <servlet-name>init</servlet-name>
    <servlet-
class>com.kle.bca.MyServlet</servlet-class>
    <init-param>
      <param-name>param1</param-name>
      <param-value>root</param-value>
    </init-param>
  </servlet>

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

</web-app>
```

index.html:

```
<html>
<body>

<a href="init">Read Initilization
Parameters</a>
```

```
</body>

</html>
```

MyServlet.java

```
package com.kle.bca;

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

public class MyServlet extends GenericServlet
{

    ServletConfig config;


    public void service(ServletRequest req,
ServletResponse resp) throws ServletException,
IOException
    {


        config = getServletConfig();
        PrintWriter out = resp.getWriter();
        out.print("<html><body><h1>"
                + "Init Parameters :
"+config.getInitParameter("param3")+"</h1></bod
y></html>");
        out.flush();
        out.close();

    }

}
```

javax.servlet package:

- **this package contains an abstract class by name GenericServlet.**
- **Serrvlet can be a child class of GenericServlet and it must override**
- **public void service(ServletRequest req, ServletResponse resp) method.**

Ex:

```
package com.kle.bca;
import java.io.*;
import javax.servlet.*;
public class MyServlet extends GenericServlet
{
	public void service(ServletRequest req, ServletResponse resp) throws ServletException, IOException
	{
		PrintWriter pw = resp.getWriter();
		String color = req.getParameter("txt");
		pw.print("<html><body><h1><font color="+color+">Hello World</h1></body></html>");
		pw.flush();
		pw.close();
	}

}
```

index.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>
```

```
    <form action="servlet" method="post">
    <pre>
    Provide Color <input type="text"
name="txt"><br>
                  <input type="submit"
value="Print">

    </pre>
    </form>
</h1>
</body>

</html>
```

Web.xml:

```
<web-app>

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

    <servlet>
        <servlet-name>serv</servlet-name>
        <servlet-
class>com.kle.bca.MyServlet</servlet-class>
    </servlet>

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


</web-app>
```

javax.servlet.http package:

- this package contains a class by name HttpServlet.
- Serrvlet can be a child class of HttpServlet and it must override the following methods based on Http Client form requests,
- If form request method is "get"
  Then we need to override,
- **protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException**

- If form request method is "post"
  Then we need to override,
- **protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException**

Ex:

web.xml

```
<web-app>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>

  <servlet>
    <servlet-name>login</servlet-name>
    <servlet-
class>com.kle.bca.MyLogin</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>login</servlet-name>
    <url-pattern>/login</url-pattern>
```

```
  </servlet-mapping>

</web-app>
```

index.html

```
<html>
<head>
<title>Login Page</title>
</head>
<body bgcolor="yellow">
    <form action="login" method="post">
    <pre>
        UserName <input type="text"
name="txtname"><br>
        Password <input type="password"
name="txtpass"><br>
                 <input type="submit"
value="Login">
    </pre>
    </form>
</body>
</html>
```

MyLogin.java(Servlet)

```
package com.kle.bca;

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

public  class  MyLogin  extends  HttpServlet
{
```

```
    protected void doPost(HttpServletRequest
req, HttpServletResponse resp) throws
ServletException, IOException
    {
        String uname =
req.getParameter("txtname");
        PrintWriter pw = resp.getWriter();

        pw.println("<html<body><h1><b1>"
                +"Welcome User
"+uname+"</b></h1>|</body></html>");
        pw.flush();
        pw.close();

    }
}
```

Java Server Pages(JSP):

Java Server Pages (JSP) is a technology that allows developers to create dynamic web pages using a combination of HTML, XML, and Java code. JSP pages are executed on a web server, and the resulting output is sent to the client's web browser.

JSP provides a way to easily access Java code and objects from within a web page, simplifying the creation of dynamic web pages.

JSP pages are typically used in conjunction with Java servlets, which handle data processing and client requests. JSP is part of the Java EE platform and is supported by most web servers and servlet containers.

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

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.

Advantages of JSP over Servlet

There are many advantages of JSP over the Servlet. They are as follows:

1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

4) Less code than Servlet

In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

Facts about jsp:

Here are some facts about JSP (JavaServer Pages):

- JSP stands for Java Server Pages.
- JSP is a technology to build dynamic web applications.
- JSP is a part of Java Enterprise Edition (Java EE).
- JSP is similar to HTML pages, but they also contain Java code executed on the server side.
- Server-side scripting means the JSP code is processed on the web server rather than the client machine.
- A JSP page is a file with a ".jsp" extension that can contain a combination of HTML Tags and JSP codes.
- To create a web page, JSP uses a combination of HTML or XML markup, JSP tags, expressions, and Java code.
- JSP tags begin with <% and end with %>.
- JSP expressions are used to insert dynamic content into the page and begin with <%= and end with %>.
- JSP can use JavaBeans to store and retrieve data.
- JSP requires a Java development environment and a Java Servlet Container such as Apache Tomcat or Jetty.
- JSP is widely used in the industry for creating enterprise web applications.
- JSP is an improved extended version of Servlet technology.

Advantages of JSP:

- Some advantages of JSP (Java Server Pages) include the following:
- JSP offers an efficient and more straightforward approach to coding dynamic web pages.
- JSP provides a wide variety of pre-built tags and custom tags, which can be used to add functionality to web pages
- JSP allows developers to separate the presentation of the web page from the logic and processing, making it easier to maintain the web application.
- Web Container (or application server like tomcat) handles changes when changes are done in the JSP code and does not need recompilation.
- JSP allows developers to follow the Model-View-Controller (MVC) design pattern, which separates a web application's presentation, logic, and data. This makes it easier to create scalable and maintainable web applications.
- JSP is commonly used with other Java technologies, such as JavaServer Faces (JSF), which provides a framework for building web applications, making the development process more streamlined.
- Provides good security features like session tracking, user authentication, and access restriction.

Creating a simple JSP Page

To create the first JSP page, write some HTML code as given below, and save it by .jsp extension. We have saved this file as index.jsp. Put it in a folder and paste the folder in the web-apps directory in apache tomcat to run the JSP page.

index.jsp (Right click on webcontent, choose jsp file and give name index.jsp)

Let's see the simple example of JSP where we are using the scriptlet tag to put Java code in the JSP page. We will learn scriptlet tag later.

```
<html>
<body>
<% out.print(2*5); %>
</body>
```

```
</html>
```

JSP tags:
In JSP, java code can be written inside the jsp page using the scriptlet tag. Let's see what are the scripting elements first.

JSP Scripting elements

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

- scriptlet tag
- expression tag
- declaration tag

1. JSP scriptlet tag

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

```
<html>
<body>
<% out.print("welcome to jsp"); %>
</body>
</html>
```

2. JSP expression tag

The code placed within **JSP expression tag** is *written to the output stream of the response*. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP expression tag

```
<%= statement %>
```

Ex:

```
<html>
<body>
```

```
<%= "welcome to jsp" %>
</body>
</html>
```

3. JSP Declaration Tag

The **JSP declaration tag** is used *to declare fields and methods.*

In this example of JSP declaration tag, we are declaring the field and printing the value of the declared field using the jsp expression tag.
Ex:

```
<html>
<body>
<%! int data=50; %>
<%= "Value of the variable is:"+data %>
</body>
</html>
```

JSP request implicit object

The **JSP request** is an implicit object of type HttpServletRequest i.e. created for each jsp request by the web container. It can be used to get request information such as parameter, header information, remote address, server name, server port, content type, character encoding etc.

It can also be used to set, get and remove attributes from the jsp request scope.

Let's see the simple example of request implicit object where we are printing the name of the user with welcome message.

index.html:

```
<form action="welcome.jsp">
<input type="text" name="uname">
```

```
<input type="submit" value="go"><br/>
</form>
```

welcome.jsp

```
<%
String name=request.getParameter("uname");
out.print("welcome "+name);
%>
```

Methods of request implicit object:

- **Enumeration getAttributeNames():** is used for returning an Enumeration that contains the attribute names presented to this request.
- **Cookie[] getCookies():** is used for returning an array that contains all of the cookie-objects of the client sent associated with a particular request.
- **Enumeration getParameterNames():** is used for returning an enumeration of String-objects that contain the names of the parameters rested in this request.
- **Enumeration getHeaderNames():** is used for returning an enumeration of all the header names associated with the request.
- **HttpSession getSession():** It is used to return the current session connected with your request or create a session if it does not have any session.
- **HttpSession getSession(boolean create):** is used to return the current HttpSession linked with the request or create a new session if there is no current session.
- **Locale getLocale():** is used for returning the chosen Locale that will be accepted by the client, based upon the Accept-Language header.
- **Object getAttribute(String name):** is used for returning the value of the named attribute as an Object or set as null.
- **ServletInputStream getInputStream():** is used to retrieve the body of the request in the form of binary data through a ServletInputStream.
- **String getAuthType():** is used for returning the name of the authentication scheme (BASIC, SSL, or null) implemented for protecting the servlet.
- **String getCharacterEncoding():** is used for returning the name of the character encoding implemented in the body of a request.

- **String getContentType():** is used for returning the MIME type of the requested content body.
- **String getContextPath():** is used for returning the portion of the request URI, which is used for indicating the context of the request.
- **String getHeader(String name):** is used for returning the value of the specified request header in the form of a String.
- **String getMethod():** is used for returning the name of the HTTP method (GET, PUT, and POST) through which this request was made.
- **String getPathInfo():** is used for returning any extra path information connected to the URL sent by the client when the request is made.
- **String getProtocol():** is used for returning the name and the version of the protocol.
- **String getQueryString():** is used for returning the query string contained in the request URL following the path.
- **String getRemoteAddr():** is used for returning the Internet Protocol (IP) address of the client, which is used by all websites.
- **String getRemoteHost():** is used for returning the fully qualified name of the client who sent the request.
- **String getRemoteUser():** is used to return the user's login, making an authenticated request or null if the user has not yet authenticated.
- **String getRequestURI():** is used for returning the part of a request's URL from the protocol name till the starting line of the HTTP request.
- **String getRequestedSessionId():**is used for returning the particular session ID of the client.
- **String getServletPath():** is used for returning the part of this request's URL, which calls the JSP.
- **String[] getParameterValues(String name)**: is used for returning an array of String objects that will contain all of the values of a requested parameter or otherwise returns null.

User Session:

A session can be defined as an object associated with each user with a unique session ID, and the user's data is based on the account they have registered. Different forms of data can be set in a session; These data related to each user of the site help the user and the website owner in different ways. As you know, HTTP is a "stateless" protocol; Whenever a user visits a web page, the user opens a separate connection with the webserver, and the server does not keep a record of preceding client requests.

Different approaches to maintain a session between client and server are:

1. Cookies: Cookies are text files that allow programmers to store some information on a client computer, and they are kept for usage tracking purposes.
2. Passing Session ID in URL: Adding and passing session ID to URL is also a way to identify a session. However, this method is obsolete and insecure because the URL can be tracked.

User Sessions

A JSP program must be able to track a session as a client moves between HTML pages and JSP programs.

There are three commonly used methods to track a session.

1. Using Hidden Field
2. Using Cookies
3. Using JavaBean

Cookies

- Cookies is a small piece of information created by a JSP program that is stored on the Client's hard disk by the browser.
- You can create and read a cookie by using methods of the cookie class and the response object.

Example: how to create a cookie

```
<HTML>
<HEAD>
```

```
<TITLE>JSP PROGRAMMING </TITLE>
</HEAD>
<BODY>
<%! String MyCookieName = “userId”;
      String MyCookieValue = “Jk12345”;
      Response.addCookie(new Cookie(MyCookieName, MyCookieValue));
%>
</BODY>
</HTML>
```

HOW TO READ A COOKIE

```
<HTML>
  <HEAD>
    <TITLE> JSP Programming</TITLE>
</HEAD>
<BODY>
<%! String MyCookieName=“userID”;
      String MyCookieValue;
       String Cname,Cvalue;
       int found=0;
       Cookie[] cookies=request.getCookies();
       for(int i=0;i<cookies.length;i++){
            CName=cookies[i].getName();
             CValue= cookie[i].getValue();
              if(MyCookieName.equals(cookieNames[i])){
               found=1;
             MyCookieValue=CookieValue;
             }
}
          if(found==1){ %>
         <p> Cookie name =<% MYCookieName%></p>
          <p> Cookie Value=<% MyCookieValue %></p>
          <&}>
</BODY>
</HTML>
```

Session implicit object:

- A session object is the most commonly used implicit object implemented to store user data to make it available on other JSP pages until the user's session is active.
- The session implicit object is an instance of a javax.servlet.http.HttpSession interface.
- This session object has different session methods to manage data within the session scope.

Example:

index.html

```
<html>
  <head>
    <title>User login form</title>
  </head>
  <body>
    <form action="login.jsp">
      Username: <input type="text" name="uname" /> <br />
      <input type="submit" value="Submit Details" />
    </form>
  </body>
</html>
```

login.jsp

```
<%@ page import = " java.util.* " %>
<%
String username = request.getParameter("uname");
if(username.equals("admin"))
{
  session.setAttribute("uname",username);
  response.sendRedirect("home.jsp");
}
else
{
  out.print("Invalid Username");
}
%>
```

home.jsp

```
<%
String session_u_name = (String)session.getAttribute("uname");
out.print("Hi "+session_u_name);
%>
```

JSP session methods:

1. **public Object getAttribute(String name)**: is used for returning the object bound with the specified name for a session and null if there is no object.
2. **public Enumeration getAttributeNames():** is used for returning an Enumeration of String objects that will hold the names of all the objects to this session.
3. **public long getCreationTime():** is used for returning the time when the session was created right from midnight January 1, 1970, GMT.
4. **public String getId():** is used for returning a string that will hold a unique identifier assigned to your session.
5. **public long getLastAccessedTime():** is used for returning the latest time your client sent a request linked with the session.
6. **public int getMaxInactiveInterval():** is used for returning the highest time interval (in seconds), which has to be maintained by the servlet container as a session gets opened between client accesses.
7. **public void invalidate():** is used for invalidating a session and unbinds its objects bound to it.
8. **public boolean isNew():** is used for returning a true when the client does not know anything about the session or when the client chooses not to join the session.
9. **public void removeAttribute(String name):** is used for removing the object bound specifically to a session.
10. **public void setAttribute(String name, Object value):** is used for binding an object to your session with the help of a specified name.

references: https://www.w3schools.in/jsp/intro

Advanced Java Question Bank & Assignment No - 3

UNIT-IV

Submission Due Date: 06/02/2023

2 Marks

1. How to declare variables in jsp? Give the syntax.
2. Define cookie. What are arguments used to define a cookie.
3. What is servlet? Mention servlet packages.
4. Differentiate servlet and JSP.
5. Define jsp?
6. Which two packages constitute servlet API?
7. State the advantages of servlets over CGI
8. Differentiate between servlet and JSP
9. What are the exceptions thrown by a servlet
10. Name the interfaces of javax.servlet package and their purposes?
11. Name the classes of javax.servlet package and their purposes?
12. Name the interfaces of javax.servlet.http package and their purposes?
13. Name the classes of javax.servlet.http package and their purposes?

5/10 Marks

1. Explain JSP tags.
2. Write a jsp program to find factorial of a given number.
3. How to create user session in java servlets.
4. Explain life cycle of a servlet.
5. How to read servlet parameters? Give example.
6. Explain anatomy of java servlet.
7. Write a program to add two numbers in JSP.
8. Write the steps to write and execute the simple servlet program?
9. Write a simple servlet program to display hello world on screen.
10. List and describe the classes and interfaces of javax.servlet package
11. List and describe the classes and interfaces of javax.servlet.http package
12. Explain handling http get requests and responses in servlet with example program?
13. Explain handling http post requests and responses in servlet with example program?
14. Explain the different types of jsp tags?
15. Define variable and write a simple jsp program that declares and uses a

variable

16. write a Java Servlet program to create a cookie and read its contents.
17. Write a Java servlet program to implement a dynamic HTML using servlet. 4. (username and password should be accepted using HTML and displayed using servlet).

Note: Refer all advanced java Lab journal programs on this unit.