JAVA Servlet

Date
Page

What is servlet

→ A servlet is a java class that extends the javax.servlet.~~http~~.HttpServlet / jakarta.servlet.ht HttpServlet class (in latest version) class and is used to handle request and responses in a web application. it acts as a middleman between the web server and the application, processing incoming request and generating the ~~appropri~~ processing incoming requests and generating the appropriate responses

Key features and functionalities of servlet

1) Request handling

2) Response generation

3) Session management

4) Filter chaining

5) Asynchronous processing

Behind the Scene

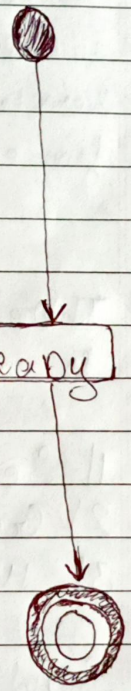
Date
Page

Servlet life cycle

The life cycle of a servlet instance
1) Servlet ~~is~~ Class is loaded
2) Servlet instance is created
3) init method is invoked
4) Service method is invoked
5) Destroy method is invoked.

} Imp.

1. load servlet Class
2. Create servlet instance
3. Call the init(-) method



4. Call the service(_, _) method

5. Call the destroy() method.

1) Init() Method
- init() method calls only once after creating the servlet instance.
- The init method is used to initialize the servlet.

2) Service method is invoked
Service method is called each time when request for the servlet is received by browser.

3) destroy method invoked
destroy method before removing the servlet instance from the service. ~~It~~ It gives the servlet an opportunity to clean up any resources for example memory, threads etc.

* There are there three types of ~~class~~ Class to implement

1) Servlet (interface)
2) GenericServlet (abstract class)
3) HttpServlet.

SendRedirect in Servlet

The SendRedirect() method of HttpServletResponse interface can be used to redirect response to another resource, it may be servlet, JSP or HTML

```
Response.SendRedirect("URL");
```

Date
Page

RequestDispatcher

A RequestDispatcher in servlet is an object that allows you to forward a request from one servlet to another within the same web application.

It provide a mechanism to transfer control and data from one servlet to another servlet, ensuring smooth flow of execution and efficient handling of user requests.

Methods of RequestDispatcher

1) forward(ServletRequest request,
ServletResponse response)
throws ServletException, IOException
forward a request from a servlet to another resource (servlet, JSP file or HTML file)

include(ServletRequest request,
ServletRequest response) throws
ServletException, IOException
Include the content of a resources (Servlet, JSP page or HTML file) in the response

Date
Page

Syntax

```
RequestDispatcher rd = request.RequestDispatcher(
                        "resource");

~~rd~~ rd.include(request, response);

rd.forward(request, response);
```

URL Mapping in web.xml

```
<Servlet>
<Servlet-name> MyServlet </Servlet-name>
<Servlet-class> com.example.MyServlet
</Servlet-class>
</Servlet>

<Servlet-mapping>
<Servlet-name> MyServlet </
~~Servlet~~-
<url-pattern> /MyServlet

</Servlet-mapping>
```

Welcome File List

- A collection of file names that a servlet container checks in a web application's dictionary when a user requests a specific URL without specifying a resource.

* Servers automatically identifies Index.html or index.jsp

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

parameters and attributes in Servlet

parameters

- parameters are those value which are provided by user to any servlet to process the request during the request operation.

- Servlet only read that value for request processing

- parameter ~~only~~ mostly "Data send using form", initialization parameter etc.

How to get parameter
String name = request.getparameter("name of parameter");

Attributes

- Attributes are the objects that are attached by one servlet to object (Session, request, config, context etc) and other servlet ~~can~~ can fetch that object to process to logic

- Servlet can easily modify, ~~and~~ add and remove the content of attribute when required

- How to perform operation with attributes
req.SetAttribute(String name, object variable)

Object value = req.getAttribute(String name)
req.removeAttribute(String name)

Session Tracking in servlet

Session Tracking is a way to maintain state (data) of an user. It is also known as session management in servlet

Note: Http protocol is a stateless so we need to maintain state using session tracking techniques.
Each time user requests to the server, server treats the request as the new request

Session Tracking Techniques

1. Cookies
2. Hidden Form field
3. URL Rewriting
4. HttpSession

Date
Page

Cookies

A Cookie is small piece of information that is persisted between the multiple client request. (It consist pair of name and value)

In Cookie technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser

Types of Cookie

1. Non-persistent cookie :
It is valid for single session only. It is removed each time when user closes the browser

2. Persistent cookie :
It is valid for multiple session. It is not removed each time when user closes the browser. It is removed only if user logout or signout

Cookie class

Jakarta.servlet.http.Cookie class provides the funtionality of using cookies. It provides a lot of useful methods for cookies.

Syntax
Cookie c = new Cookie()

Date
Page

Constructor of Cookie Class

Cookie() : Constructs a cookie

Cookie(String name, String value) : Constructs a cookie with a specified name and value

Methods of Cookie Class

setMaxAge(int expiry) : sets maximum age of cookie seconds

getName() : Returns the name of the cookie

getValue() : Returns the value of the cookie

setName(String name) : ~~Sh~~ Set's the name of the cookie

~~setName()~~ :
setName(String value) : set's or changes the value of the cookie

Important method

1) addCookie(Cookie ck) : method of HttpServletResponse interface is used to add cookie in response object.

2) getCookies() : Method of HttpServletRequest interface is used to ~~set~~ return all the Cookies from the browser
- It returns array of cookie's

How to Create Cookie

```
Cookie ck = new Cookie("user","sonoo jaiswal");
response.addCookie(ck);
```

How to Delete Cookie

```
Cookie ck = new Cookie("user","  ")
  ck.setMaxAge(0);
// change the maximum age to 0 second
response.addCookie(ck)
```

How to get cookies

```
Cookie ck[] = request.getCookies();
for(int i=0; i<ck.length; i++){
  out.print("<br>" + ck[i].getName() + "  " +
            ck[i].getValue());
}
```

2) Hidden Form field

A hidden (invisible) textfield is used for maintaining the state of an user at server

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

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

3 URL Rewritting

- 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 = value & name2 = value & . . . .

- when user clicks the hyperlink, the parameter name/value pair will be passed to the server

- From servlet, we can use getParameter() method to obtain a parameter value

4) ~~HT~~ HttpSession interface

- In HttpSession, Container Creates a session id for each user.
- The Container uses the id to identify the particular user

How to get HttpSession

The HttpSession interface provide two method to get the object of HttpSession:

- getSession() :
Returns the current session associated with this request, or if the request not have a session, Creates one

getSession(boolean Create) :
Returns the current HttpSession associated with this request or, if there is no current Session and Create is true, return a new session

```
HttpSession session = request.getSession();
```

Methods of HttpSession interface

1) getId() : Return a string containing the unique identifier value

2) getCreationTime() : Returns the time when this session was created measured in milliseconds since midnight January 1, 1970 GMT

3) getLastAccessedTime() : Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT

4) invalidate() : Invalidates this session then unbinds (remove) any objects bound to it

Date
Page

Servlet Filters.

A filter is an object that is invoked at the preprocessing and postprocessing of a request

It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc.

The servlet filter is pluggable, i.e. it's entry is defined in the web.xml file.
If we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change

Usage of Filter

- recording all incomming requests
- logs the IP addresses of the computers from which the requests originate
- conversion
- data compression
- encryption and decryption
- ~~Invalidation~~ Input validation etc.

1) Filter interface - life cycle methods

1) init(FilterConfig c) - invoked only once. used to initialize the filter

2) doFilter(HttpServletRequest req,
HttpServletResponse res,
FilterChain chain)

- Method is invoked every time when user request to any resource, to which the filter is mapped. It is used to perform filtering task.

destroy() - invoked only once when filter is taken out for the service.

FilterChain interface

The object of FilterChain is responsible to invoke the next filter or resource in the chain.

How to define Filter

we can define filter same as servlet

```
<web-app>

<filter>
    <filter-name> ...  </filtersname>
    ~~<filter~~
    <filter-class> ...  </filter-class>
</filter>

<filter-mapping>
  <filter-name> ....  </filter-name>
  <url-pattern> ....  </url-pattern>
</filter-mapping>

</web-app>
```

JSP (Web)

What is JSP?

Java Server page. It's a technology that allows to create dynamic web pages using HTML, CSS, Javascript, and Java code. JSP files are compiled into Servlet, which are ~~exect~~ executed on a web server to generate HTML output.

~~Fat~~ Features of JSP

HTML Embedding:

You can embed Java code directly into your HTML pages using JSP tags. This makes it easier to create dynamic content without writing seperate Java classes.

Custom Tags:

JSP supports the creation of custom tags, which can be used to encapsulate common funtionality and improve code readability

Expression Language (EL):

EL provides a simple way to access JavaBeans properties and perform basic calculations within JSP pages.

JSTL (JavaServer Pages Standard Tag Library):

JSTL provides a set of predefined tags for common task such as iteration, conditional logic and database access.

JSP Scripting elements

~~JSP Scripting tag~~

1) JSP Scriptlet tag

A Scriptlet tag is used to executed java code in JSP.

Syntax:

<% java source code %>

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

<%= "Statement" %>

3) JSP Declaration Tag

The JSP declaration tag is used to declare fields and methods

Syntax

<%! field or method declaration %>

eg.

<%! int data = 50; %>

<%= "Value of the variable is:" + data %>

JSP directive Elements

JSP directives

The JSP directive are message that tells the web container how to translate a JSP page into the corresponding Servlet

JSP page directive

The page directive defines attributes that apply to an entire JSP page

Syntax

<%@ page attributes="value" %>

Attributes of page directive

1) import = used to import class, interface, Sinilary to import in Java

2) ContentType = defines MIME (multipurpose Internet mail Extension) type of HTTP response default is "text/html"; Charset = ISO-8859-1

3) extends = defines the parent class that will be inherited

4) info = Simply sets the information of the JSP page which is retrieved later by using getServletInfo()

5) buffer = sets the buffer size in Kilobytes to handle output generated.

6 language = Specifies scripting language use in JSP default Value is "Java"

7) isELIgnored = we can ignore Expression Language (EL) in JSP set true or false

8) isTreadSafe = used to control multitreading Set to true or false.

9) errorPage = used to define error page

10) isErrorPage = use to declare that current page is error page or not

Date
Page

JSP include Directive.

The include directive is used to include the contents of any other resource it may be JSP file, html file or text file.

★ The JSP Page is translated only once so it will be better to include Static resource.

Syntax

<%@ include file="resourceName" %>

<JSP: include page="resourceName" />

JSP Taglib directive

The JSP taglib directive is used to define a tag library that defines many tags.

★ It will be better to learn it in custom tag.

Syntax

<% @ taglib uri="url-pattern" prefix="t" %>

~~use tag~~ using tag.

<t: tag-name> <\/t: tagname>

JSP Exception

Exception handling

Exception handling is the process to handle the runtime errors.

There are two ways to perform exception handling

1) By errorPage and isErrorPage attributes of page directive.

2) By <error-page> element in web.xml file

↳ If you want to handle any exception

```
<error-page>
<exception-type> java.lang.Exception </exception-type>
<location> /error.jsp </location>
</error-page>
```

↳ If you want to handle exception for a specific error

```
<error-page>
<error-code> 500 </error-code>
<location> /error.jsp <location>
</error-page>
```

JSP Custom Tag

Custom tag used with taglib directive to use a html tag

Steps to create custom tag

1) Create the Tag handler class and perform action at the start or at the end of the tag

2) Create the Tag Library Descriptor (TLD) file and define tag's

3) Create the JSP file that uses the custom tag defined in the TLD file

1) Create the Tag handler class
To create the tag Handler, we are inheriting the TagSupport class and overriding its method doStartTag().

```
public class MyTagHandler extends TagSupport {

    public int doStartTag() throws JspException {
        JspWriter out = pageContext.getOut();
        // used to print string values in browser
           like printWriter class in servlet

        out.print("<h1> Hello world </h1>");
    return Skip }
    Body }
```

2) Create the TLD file

Tag Library Descriptor (TLD) file contains information of Tag and Handler classes. It must be contained inside the web-INF directory

mytags.tld.

~~3) Create the JSP file.~~ mytags.tld

```
<taglib>
<tlib-version> 1.0 </tlib-version>
<jsp-version> 1.2 </jsp-version>
<short-name> simple </short-name
<uri> . . . . . </uri>

<tag>
<name> today </name>
<tag-class> class path </tag-class>
</tag>

</taglib>
```

Date
Page

Create the JSP file

index.jsp

```
<%@taglib uri="WEB-INF/mytags.tld"
          prefix="m" %>

Current Date and time is : <m:today/>
```

Attributes with Custom tag

Steps 1

- Define the property in the TagHandler class with the attribute name and define the SetterMethod

- Define the attribute element inside the tag element in the TLD file

mytag.tld

```
<tag>
  <name> Sample </name>
  <tag-class> . . . . . </tag-class>

  <attribute>
      <name> number </name>
      <required> true </required>
      </attribute>
  </tag>
```

Date
Page

Removing .jsp Extension from URL

Update jsp in web.xml file

```
<servlet>
    <servlet-name> Sample </servlet-name>
    <jsp-file> Sample.jsp </jsp-file>
</servlet>

</servlet-mapping>
    </servlet-name> Sample </servlet-name>
    ~~<jsp-~~
    <url-pattern> / Sample </ url- pattern>
</servlet-mapping>
```

Implicit object in JSP

Implicit objects in JSP are predefined objects that are automatically available within a JSP page without explicit declaration.

request :- object of ~~HttpServletResponce~~
HttpServletRequest

response :- object of HttpServletResponse

Session :- object of HttpSession

out :- object of printWriter