# Per Scholas - Quality Engineering

# Java EE Servlets - 02

## Servlet Context and Config

ServletConfig

- An object specific to the servlet specified by <init-param>
- The servlet container is an object used for holding the configuration details of the servlets in a web application when a request to a servlet comes, the web container will initialize the servlet
- the web container will parse the web.xml file and read the configurations of the servlet to be initialized
- the web container will then create a servlet config object and load the configuration details of the servlet in the object
- A single instance of ServletConfig is created for each servlet
- ServletConfig object is passed as an argument to the init( ) method of the servlet by the web container during servlet initialization
- The init(ServletConfig config) method needs to be overridden to use ServletConfig
- during servlet initialization, the container passes the config object as an argument to the init method
- initialization parameters are used by developers to set parameter values for the servlet to use (stored as key/value pairs)
- these parameters can be configured in the web.xml file using <init-param> tags
- initialization parameters will be loaded to the ServletConfig object during the servlet initialization phase of it's life cycle
- initialization parameters can be read using the getInitParameter( ) method of the ServletConfig object
- ServletConfig demonstration
- using the Eclipse "Initialization Parameters" option when creating a servlet
- using the web.xml file through <init-param> tags

- https://javaee.github.io/javaeespec/javadocs/javax/servlet/ServletConfig.html
- https://beginnersbook.com/2017/07/servletcoHYPERLINK "https://beginnersbook.com/2017/07/servletconfig-interface-with-example/"n HYPERLINK"https://beginnersbook.com/2017/07/servletconfig-interface-with-example/"fig-interface-with-example/
- https://www.youtube.com/watch?v=uKoBbSp0J3Y

Servlet Context

- Refers to the context in which the servlet runs
- There will only be one instance of the servlet context per web application shared by all the servlets of the application
- Used for storing values which can be shared across all the servlets in the web application
- An object which will be shared by all the servlets specified by <context-param>
- Used in development projects to store:

  master data values which are used by the presentation tier (jsp)

  menu links which are used across several jsp

  /servlets in an application

  any other data which needs to be shared across the jsp and servlets in a web application
- Context level init parameters can be set which can be reused by all the servlets in the context
- Context parameters for an application can be declared by using <context-param> tags in the web.xml file (deployment descriptor)
- Context parameters can be read using the getInitParameter( ) method of the ServletContext interface
- example: String title = context.getInitParameter( “Title”);
- Context attributes are values which are dynamically set by the application for sharing it across the application
- used for sharing values across servlets

- attributes set in the context live throughout the life span of the application until the application is removed or the attributes removed explicitly by another servlet
- setting an attribute: context.setAttribute(String attributeName, Object value);
- getting an attribute: Object variable = context.getAttribute(attributeName);
- remove an attribute: Context.removeAttribute(attributeName);
- Servlet config and context demonstrations

Response Buffering

By default, any content written to the output stream is immediately sent to the client as HTTP response. Buffering is the process by which the content are written into a buffer (temporary storage in server) before being sent to the client, thus providing the servlet a better control on the response being sent. Buffering also makes effective use of network bandwidth since response is send to the client after a specified buffer size is reached

The Java servlet API supports response buffering that allows the servlet to control how the servlet container buffers responses, and when to send a response to a client.

- http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.dc00466.0600/html/easwapp/easwapp57.htm

Response Buffer API

The following methods of HTTP Response interface helps in the buffering the response

- setBufferSize(int size) : informs the web container the buffer size to be set for the servlet response, the values represents the number of bytes of data.
- getBufferSize() : returns an int indicating the current buffer size.
- isCommitted() : returns a boolean indicating whether any part of the response has actually been sent. If this method returns true, the status code and headers cannot be changed.
- reset() : method can be used any time before commit to empty the buffer and unset the headers.
- flushBuffer() : sends content in the buffer to the client and commits the response.

## Servlet Chaining

It is a technique in which two or more servlets orchestrate in servicing a single request. In servlet chaining, one servlet's response is piped to the next servlet's input. This process continues until the last servlet is reached.Its output is then sent back to the client. The same request and response object are available across all the servlets in the process.

- include
- forward
- RequestDispatcher
- sendRedirect
- http://www.pskills.in/servlet/servlet-chaining.jsp

Request Forwarding

The HttpServletRequest interface has a method, getRequestDispatcher(), that returns a RequestDispatcher interface, which has a method, forward() to forward the request to another path.

- http://www.w3processing.com/index.php?childMenuId=39&subMenuId=100&environmentPath=EC/WL

RequestDispatcher Interface

Request Dispatcher is an object which accepts the request from the client and redirects them to any resource (Servlets, jsp, html etc). The servlet container creates the RequestDispatcher object, which is used as a wrapper around a server resource located at a particular path or identified by a particular name.

- How to create Dispatcher object ?

```
RequestDispatcher dispatcher= request.getRequestDispatcher("FooterServlet");
```

Where, FooterServlet is the servlet to which the request needs to be dispatched

- How to include files in servlet response?
- https://www.javatpoint.com/requestdispatcher-in-servlet

Methods in RequestDispatcher

- forward() : Used to forward request to another resource.

Syntax: `dispatcher.forward(request, response);`

- include() : Includes the content of a resource (servlet, JSP page, HTML file) in the response.

Syntax: dispatcher.include(request,response);

Include vs Forward

Where to use include?

- Include is used for reusing common code. Like a servlet to print the user login information in all pages, the common servlet can be included in all the pages.
- Use include if you want to present a collated response from a set of components (Servlets, jsp etc).
- Include can also be used for including static content to a page such as the page footer can be reused across all the pages in the application.

Where to use Forward?

- Forward is also used again for code reusability.
- Forward are typically used to forward a request to a success or error page after some processing.

Example: After login credentials validated if success the control will be forwarded to home page else sent to error page.

- Difference between forward and send redirect.
- http://www.java67.com/2016/03/6-difference-between-forward-and-sendredirect-in-Servlet-JSP.html

## Session Management

Session Management is a mechanism for maintaining state across multiple HTTP requests. This is managed by the Web container.  It is a technique to hold some values passed by user across multiple HTTP requests arising out from a single browser instance.

Session life cycle is managed for each web browser instance opened and it exists till the browser is closed (or) till the session time outs (set in the server configurations).

Need for Session:

HTTP is a stateless protocol. So, if developers need to develop pages where he needs to maintain the application user's state across multiple request, he can use session.

Session Management Techniques

The following are the session management techniques. These are used for managing the data between pages of the user in a session.

1. Hidden Fields 2. URL Rewriting 3. Cookies 4. Session Object

What is a Hidden Field?

- Hidden fields are nothing but normal HTML form element with type Hidden to store the user data across HTTP request.

How it is used for session tracking?

- The user state to be preserved across HTTP request can be stored as a hidden element in the HTML pages which can be read from other pages

How to access Hidden Fields Values?

- Hidden field values can be read using the request.getParameter("Key") method.

What is URL rewriting?

- The mechanism by which the user state or information's are appended to the URL for tracking the state/session of the user.

What is a Cookie?

- Simple piece of textual information (in key value pair format) stored on the client (browser machine).
- Cookies information are returned to the server with every request from the client.

How cookies are identified in client machine?

- The browser matches the cookies present with the site URL. If a match is found, that cookie is returned with the request

Steps for Using Cookie

Step 1: Create the cookie object

Step 2: Set the cookie object to the HTTP response

Step 3: Read the cookies from the next HTTP request

Step 4: Validate the cookie value for session tracking

Cookies are Created using the Cookie class of Servlet API.

Syntax: Cookie cookie = new Cookie(identifier,value);

Where identifier is the name of the state information, value represents the value of

the state.

How to Set Cookies to the response?

- Since cookies are stored at the client, cookies are set to the response object and send to the client using the addCookie() method of the HttpServletResponse interface

- https://examples.javacodegeeks.com/enterprise-java/servlet/java-servlet-session-management-example/

What is a Session Object?

- Session Object is a container used for storing user states in server.
- The session objects lifecycle is maintained by web container.
- The Servlet API HttpSession interface provides features for Session tracking.
- HttpSession objects are objects used for storing client session information

HTTP SESSION APIs

- https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpSession.html

General Tips for using Session Management Techniques

- Typically, in Projects we use Session technique for holding the user attributes.
- Cookies are rarely used as there could be users accessing browsers without cookie support.
- If there is some state which is maintained in one or few pages, we use URL rewriting or hidden fields.
- Most Importantly never load session object with bulky objects. Load only the states which are used across the web application.

Attribute Scopes in Servlet

- http://r4r.in/servlet/topic/?ct=3&subct=71&tp=370

## Servlet Filter

Servlet filter are programs that runs on the server which intercepts HTTP request/ response for transformations/processing/filtering.

A filter program used for transforming/ filtering HTTP request or transforming HTTP response.

How does FILTER work?

Step 1: Intercepts the client HTTP request and preprocess the request.
Step 2: Invokes the requested resource with the pre-processed HTTP request.
Step 3: Intercepts the HTTP response and transforms it.
Step 4: The transformed HTTP response will be sent back to the client.

Use of Filters:

- Used for authenticating all HTTP request coming to the server.
- All the HTTP requests can be logged and audited in a flat file or database for tracking users of a web application.
- To perform some formatting of HTTP response, say display a message in the header (or) bread crumbs for all the pages of the application.

Filter Interfaces
1.Filter: Contains the methods for filtering the request/response
2.FilterChain: Contains the methods for chaining the Filters
3. FilterConfig: A filter configuration object used by a servlet container to pass information to a filter during initialization. Contains the methods for accessing the filter initialization parameters and getting the servlet context in which the filter runs.

All filters must implement javax.servlet.Filter.
This interface comprises three methods which needs to be overridden and implemented,
1. doFilter()
2. init()
3. destroy()

- https://www.journaldev.com/1933/java-servlet-filter-example-tutorial

Filter Chaining:

- Filter chaining is the process of applying more than one filter to a Servlet.
- This is done by configuring more than one filter mapping for a servlet in the web.xml file
- The filters are invoked in the order they are declared in the web.xml file

Summary of Filters:

- A filter can be used for filtering more than one servlet, by configuring the same filter with multiple servlets.
- More than one filter can applied to a single servlet, applying two filters with a servlet called servlet chaining.
- The following are some instances where filter are used on application development,
  Authentication/Authorization of all HTTP request to web applications to ensure that authorized user access we application.
  To pre/post process request like checking message format, validating form data etc.

## Servlet Listeners

- Listeners are used to perform some logic on trigger of certain events.
- Listeners are java classes that is used to implement the event handling mechanism
- https://www.javatpoint.com/Event-and-Listener-in-Servlet

ServletContextEvents

- https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletContextEvent.html

Classification of Events

- https://docs.oracle.com/cd/E13222_01/wls/docs90/webapp/app_events.html

## Servlet Filters and Listeners

- https://docs.oracle.com/cd/B14099_19/web.1012/b14017/filters.htm