**1. Life Cycle Of a Servlet:-**

The Servlet life cycle mainly goes through four stages,

* Loading a Servlet.
* Initializing the Servlet.
* Request handling.
* Destroying the Servlet.

**Loading a Servlet**: Initializing the context, on configuring the Servlet with a zero or positive integer value.

If the Servlet is not preceding stage, it may delay the loading process until the Web container determines that this Servlet is needed to service a request.

**Initializing a Servlet**: The container initializes the Servlet object by invoking the **Servlet.init(ServletConfig)** method which accepts ServletConfig object reference as parameter. if the Servlet fails to initialize, then it informs the Servlet container by throwing the **ServletException**or **UnavailableException.**

**Handling request**: After initialization, the Servlet instance is ready to serve the client requests.

The **service()** method while processing the request may throw the **ServletException**or **UnavailableException**or **IOException.**

**Destroying a Servlet**: When a Servlet container decides to destroy the Servlet, it performs the following operations,

* It allows all the threads currently running in the service method of the Servlet instance to complete their jobs and get released.
* After currently running threads have completed their jobs, the Servlet container calls the destroy() method on the Servlet instance.

**2. ServletContext:-**

An object of ServletContext is created by the web container at time of deploying the project. This object can be used to get configuration information from web.xml file. There is only one ServletContext object per web application.

If any information is shared to many servlet, it is better to provide it from the web.xml file using the **<context-param>** element.

**3. ServletConfig:-**

An object of ServletConfig is created by the web container for each servlet. This object can be used to get configuration information from web.xml file.

If the configuration information is modified from the web.xml file, we don't need to change the servlet. So it is easier to manage the web application if any specific content is modified from time to time.

**4. getParameterNames():-**

Returns an Enumeration of String objects containing the names of the parameters contained in this request. If the request has no parameters, the method returns an empty Enumeration.

**5. getParameterValues():-**

Returns an array of String objects containing all of the values the given request parameter has, or null if the parameter does not exist. If the parameter has a single value, the array has a length of 1.

**6. request.getquerystring():-**

Returns the query string that is contained in the request URL after the path. This method returns null if the URL does not have a query string. Same as the value of the CGI variable QUERY\_STRING.

The method **getQueryString()**returns a string of data, filled by the user in FORM fields (of text boxes, check boxes etc.), and sent to server.

**7. init() method**:- The **Servlet.init()** method is called by the Servlet container to indicate that this Servlet instance is instantiated successfully and is about to put into service.

**service() method**: The **service()** method of the Servlet is invoked to inform the Servlet about the client requests.

* This method uses **ServletRequest** object to collect the data requested by the client.
* This method uses **ServletResponse** object to generate the output content.

**destroy() method**:- The **destroy()** method runs only once during the lifetime of a Servlet and signals the end of the Servlet instance.

**8. pageContext:-**

|  |
| --- |
| In JSP, pageContext is an implicit object of type PageContext class.The  pageContext object can be used to set,get or remove attribute from one of the  following scopes:   * page * request * session * application |
| In JSP, page scope is the default scope. |

**9. Difference between enum and enumeration:-**

Enum in Java is a datatype which stores a set of constant values. You can use these to store fixed values such as days in a week, months in a year etc.

Eg: enum Days {

SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY

}

public class EnumerationExample {

public static void main(String args[]) {

Days constants[] = Days.values();

System.out.println("Value of constants: ");

for(Days d: constants) {

System.out.println(d);

}

}

}

In java.util package Java provides an interface with name Enumeration, an object implementing this interface generates a series of elements and you can retrieve elements from this using the **nextElement()** method.

Collections like Vector, HashTable etc. has a method named elements() which returns an Enumeration (interface) object containing all the elements in the collection. Using this object, you can get the elements one by one using the nextElement() method.

import java.util.Enumeration;

import java.util.Vector;

public class EnumerationExample {

public static void main(String args[]) {

//instantiating a Vector

Vector<Integer> vec = new Vector<Integer>( );

//Populating the vector

vec.add(1254);

vec.add(4587);

vec.add(5211);

vec.add(4205);

vec.add(1124);

vec.add(8115);

//Retrieving the elements using the Enumeration

Enumeration<Integer> en = vec.elements();

while(en.hasMoreElements()) {

System.out.println(en.nextElement());

}

}

}

**10. Child Selector:-** Child Selector is used to match all the elements which are child of a specified element. It gives the relation between two elements. The element > element selector selects those elements which are the children of specific parent. The operand on the left side of > is the parent and the operand on the right is the children element.

**Syntax:**

element > element {

// CSS Property

}

**Descendant selector:-** Descendant selector is used to select all the elements which are child of the element (not a specific element). It select the elements inside the elements i.e it combines two selectors such that elements matched by the second selector are selected if they have an ancestor element matching the first selector.

Syntax:

element element {

// CSS Property

}