Servlet

Servlet is an interface that must be implemented for creating any Servlet. There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

Servlet technology is used to create a web application (resides at server side and generates a dynamic web page). Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests (not just web).

# Advantages of Servlet

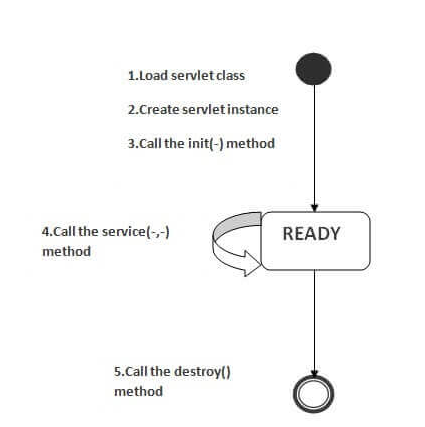
* **Better performance:** because it creates a thread for each request, not process.
* **Portability:** because it uses Java language.
* **Robust:** JVM manages Servlets, so we don't need to worry about the memory leak, garbage collection, etc.
* **Secure:** because it uses java language.

# Servlet API

**Note**: many of the interfaces don’t have an implementing class, the methods work with the interface type as parameter, while the container provide the object to the servlet.

1. Interfaces in javax.servlet package
   * Servlet
   * ServletRequest
   * RequestDispatcher
   * ServletConfig
   * ServletContext
   * SingleThreadModel
   * Filter
   * FilterConfig
   * FilterChain
   * ServletRequestListener
   * ServletRequestAttributeListener
   * ServletContextListener
   * ServletContextAttributeListener
2. Classes in javax.servlet package
   * GenericServlet
   * ServletInputStream
   * ServletOutputStream
   * ServletRequestWrapper
   * ServletResponseWrapper
   * ServletRequestEvent
   * ServletContextEvent
   * ServletRequestAttributeEvent
   * ServletContextAttributeEvent
   * ServletException
   * UnavailableException
3. Interfaces in javax.servlet.http package
   * HttpServletRequest
   * HttpServletResponse
   * HttpSession
   * HttpSessionListener
   * HttpSessionAttributeListener
   * HttpSessionBindingListener
   * HttpSessionActivationListener
   * HttpSessionContext (deprecated now)
4. Classes in javax.servlet.http package
   * HttpServlet
   * Cookie
   * HttpServletRequestWrapper
   * HttpServletResponseWrapper
   * HttpSessionEvent
   * HttpSessionBindingEvent
   * HttpUtils (deprecated now)

# Life Cycle of Servlet:



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.
4. **service method is invoked** the service() method is the main method to perform the actual task. The servlet container (i.e., web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.
5. **destroy method is invoked** The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities. After the destroy() method is called, the servlet object is marked for garbage collection.

# Creating Servlet:

The servlet example can be created by three ways:

* By implementing Servlet interface,
* By inheriting GenericServlet class or HttpServlet class

The mostly used approach is by extending HttpServlet because it provides http request specific method such as doGet(), doPost(), doHead() etc.

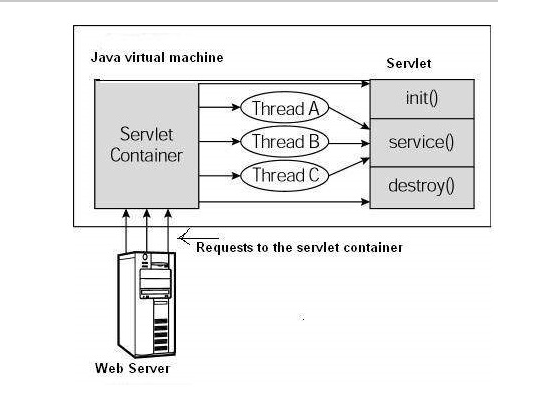
# How Servlet works?

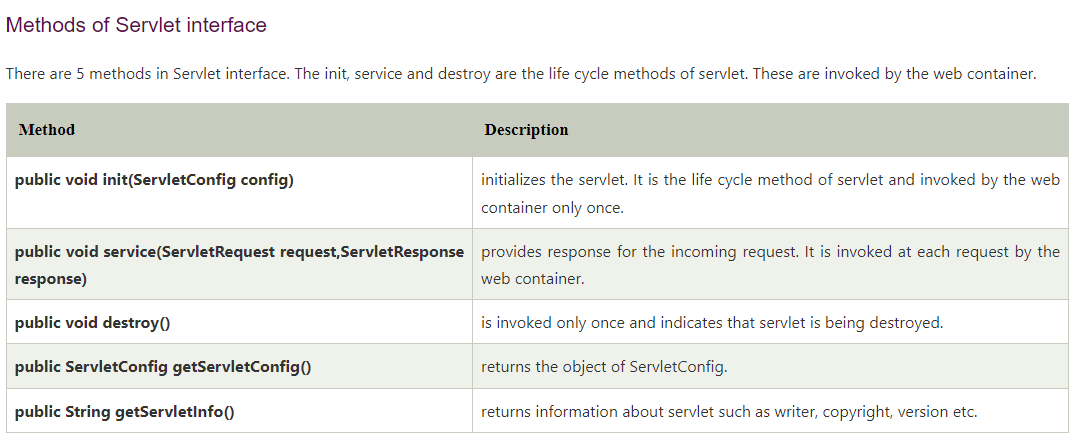
The server checks if the servlet is requested for the first time.

* If yes, web container does the following tasks:
  + loads the servlet class.
  + instantiates the servlet class.
  + calls the init method passing the ServletConfig object
* else calls the service method passing request and response objects

The web container calls the destroy method when it needs to remove the servlet such as at time of stopping server or un-deploying the project.

The **servlet container** handles multiple requests by spawning **multiple threads**, each thread executing the service() method of a single instance of the servlet (depends on the implementation, it can be singleton or pool of instance).



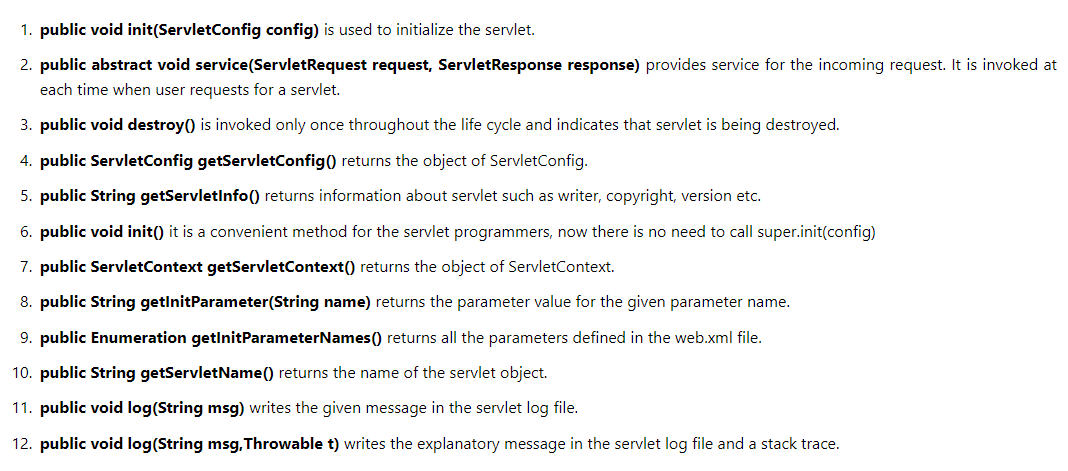


# GenericServlet class

**GenericServlet** class implements **Servlet**, **ServletConfig** and **Serializable** interfaces. It provides the implementation of all the methods of these interfaces except the service method.

* GenericServlet class can handle any type of request so it is protocol-independent.
* We may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

It has many methods.



# HttpServlet class

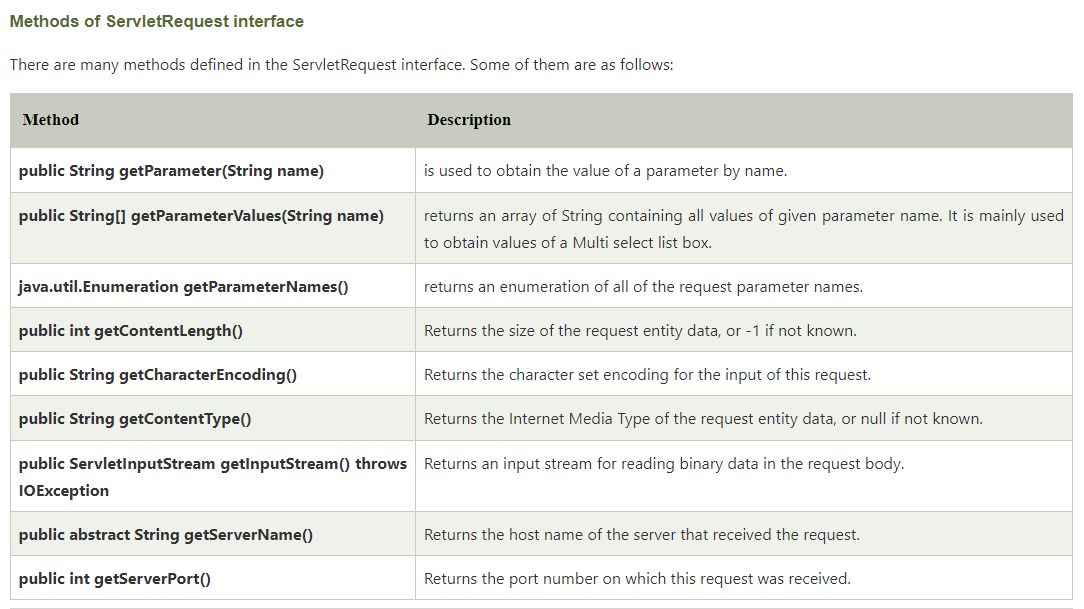
The HttpServlet class extends the **GenericServlet** class and implements **Serializable** interface. It provides http specific methods such as **doGet**, **doPost**, **doHead**, **doTrace** etc.

There are many methods in HttpServlet class.



# ServletRequest Interface

ServletRequest is used to provide the client request information to a servlet such as content type, content length, parameter names and values, header information, attributes etc.



# ServletResponse Interface

The servlet container is connected to the web server that receives Http Requests from client on a certain port.

When client sends a request to web server, the servlet container creates **HttpServletRequest** and **HttpServletResponse** objects and passes them as an argument to the servlet **service()** method.

The response object allows you to format and send the response back to the client.

