Servlet API

The javax.servlet and javax.servlet.http packages represent interfaces and classes for servlet api.

The **javax.servlet** package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.

The **javax.servlet.http** package contains interfaces and classes that are responsible for http requests only.

javax.servlet package

Interfaces	**Classes**
1. Servlet	1. GenericServlet
2. ServletRequest	2. ServletInputStream
3. ServletResponse	3. ServletOutputStream
4. RequestDispatcher	4. ServletRequestWrapper
5. ServletConfig	5. ServletResponseWrapper
6. ServletContext	6. ServletRequestEvent
7. SingleThreadModel	7. ServletContextEvent
8. Filter	8. ServletRequestAttributeEvent
9. FilterConfig	9. ServletContextAttributeEvent
10. FilterChain	10. ServletException
11. ServletRequestListener	11. UnavailableException
12. ServletRequestAttributeListener	
13. ServletContextListener	
14. ServletContextAttributeListener	

javax.servlet.http package

Interfaces	**Classes**
1. HttpServletRequest	1. HttpServlet
2. HttpServletResponse	2. Cookie
3. HttpSession	3. HttpServletRequestWrapper
4. HttpSessionListener	4. HttpServletResponseWrapper
5. HttpSessionAttributeListener	5. HttpSessionEvent
6. HttpSessionBindingListener	6. HttpSessionBindingEvent
7. HttpSessionActivationListener	7. HttpUtils (deprecated now)
8. HttpSessionContext (deprecated now)	

Servlet Interface

- **Servlet interface** provides common behavior to all the servlets.
- Servlet interface needs to be implemented for creating any servlet (either directly or indirectly). It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

Methods of Servlet interface

There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

1. **public void init(ServletConfig config) :** initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
2. **public void service(ServletRequest request, ServletResponse response) :** provides response for the incoming request. It is invoked at each request by the web container.
3. **public void destroy() :** is invoked only once and indicates that servlet is being destroyed.
4. **public ServletConfig getServletConfig() :** returns the object of ServletConfig, which contains initialization and startup parameters for this servlet.
5. **public String getServletInfo() :** returns information about servlet such as author, copyright, version etc.

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.
- You may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

Methods of GenericServlet class

1. **public void init(ServletConfig config)** is used to initialize the servlet.
2. **public abstract void service(ServletRequest request, ServletResponse response)** provides service for the incoming request. It is invoked at each time when user requests for a servlet.
3. **public void destroy()** is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
4. **public ServletConfig getServletConfig()** returns the object of ServletConfig.
5. **public String getServletInfo()** returns information about servlet such as author, copyright, version etc.
6. **public void init()** it is a convenient method for the servlet programmers, so there is no need to call super.init(config)
7. **public ServletContext getServletContext()** returns the object of ServletContext.
8. **public String getInitParameter(String name)** returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.

9. **public Enumeration getInitParameterNames()** returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.
10. **public String getServletName()** returns the name of the servlet object.
11. **public void log(String msg)** writes the given message in the servlet log file.
12. **public void log(String msg,Throwable t)** writes the explanatory message in the servlet log file and a stack trace.

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.

Methods of HttpServlet class

1. **public void service(ServletRequest req,ServletResponse res)** dispatches the request to the protected service method by converting the request and response object into http type.

2. **protected void service(HttpServletRequest req, HttpServletResponse res)** receives the request from the service method, and dispatches the request to the doXXX() method depending on the incoming http request type.

3. **protected void doGet(HttpServletRequest req, HttpServletResponse res)** handles the GET request. It is invoked by the web container.

4. **protected void doPost(HttpServletRequest req, HttpServletResponse res)** handles the POST request. It is invoked by the web container.

5. **protected void doHead(HttpServletRequest req, HttpServletResponse res)** handles the HEAD request. It is invoked by the web container.

6. **protected void doOptions(HttpServletRequest req, HttpServletResponse res)** handles the OPTIONS request. It is invoked by the web container.

7. **protected void doPut(HttpServletRequest req, HttpServletResponse res)** handles the PUT request. It is invoked by the web container.

8. **protected void doTrace(HttpServletRequest req, HttpServletResponse res)** handles the TRACE request. It is invoked by the web container.

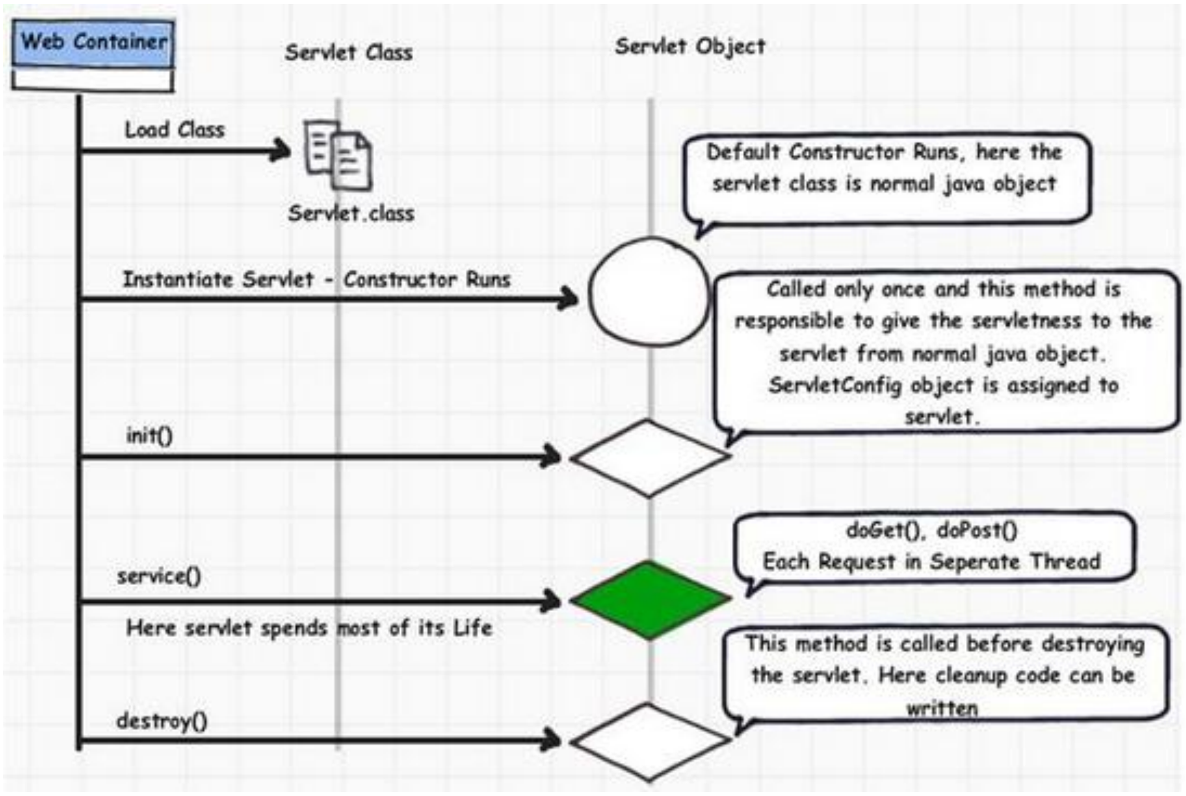
9. **protected void doDelete(HttpServletRequest req, HttpServletResponse res)** handles the DELETE request. It is invoked by the web container.

10. **protected long getLastModified(HttpServletRequest req)** returns the time when HttpServletRequest was last modified since midnight January 1, 1970 GMT.

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.


Web Container
Servlet Class
Servlet Object
Load Class
Servlet.class
Default Constructor Runs, here the servlet class is normal java object
Instantiate Servlet - Constructor Runs
Called only once and this method is responsible to give the servletness to the servlet from normal java object. ServletConfig object is assigned to servlet.
init()
doGet(), doPost() Each Request in Seperate Thread
service()
Here servlet spends most of its Life
This method is called before destroying the servlet. Here cleanup code can be written
destroy()

1. Servlet class is loaded

The class loader 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 response)
   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()

How Servlet works?

It is important to learn how servlet works for understanding the servlet well. Here, we are going to get the internal detail about the first servlet program.

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 undeploying the application.

How web container handles the servlet request?

The web container is responsible to handle the request. Let's see how it handles the request.

- maps the request with the servlet in the web.xml file.
- creates request and response objects for this request
- calls the service method on the thread
- The public service method internally calls the protected service method
- The protected service method calls the doXXX method depending on the type of request.
- The doXXX method generates the response and it is passed to the client.

- After sending the response, the web container deletes the request and response objects. The thread is contained in the thread pool or deleted depends on the server implementation.

What is written inside the public service method?

The public service method converts the ServletRequest object into the HttpServletRequest type and ServletResponse object into the HttpServletResponse type. Then, it calls the protected service method passing these objects.

```
public void service(ServletRequest req, ServletResponse res)
                                            throws ServletException, IOException
    {
        HttpServletRequest request;
        HttpServletResponse response;
        try
        {
            request = (HttpServletRequest) req;
            response = (HttpServletResponse) res;
        }
        catch(ClassCastException e)
        {
            throw new ServletException("non-HTTP request or response");
        }
        service(request, response);
    }
```

What is written inside the protected service method?

The protected service method checks the type of request, if request type is get it calls doGet method, if request type is post it calls doPost method, so on.

```
protected void service(HttpServletRequest req, HttpServletResponse resp)
                                            throws ServletException, IOException
    {
        String method = req.getMethod();
        if(method.equals("GET"))
        {
            long lastModified = getLastModified(req);
            if(lastModified == -1L)
            {
                doGet(req, resp);
            }
    //rest of the code
        }
    }
```