What is a servlet

* A servlet is a Java program used to extend the capabilities of servers that host applications accessed by means of a request-response programming model
* The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. All servlets must implement the Servlet interface, which defines lifecycle methods. When implementing a generic service

Servlet Life cycle

The lifecycle of a servlet is controlled by the container in which the servlet has been deployed. When a request is mapped to a servlet, the container performs the following steps.

1. If an instance of the servlet does not exist, the web container
   1. Loads the servlet class.
   2. Creates an instance of the servlet class.
   3. Initializes the servlet instance by calling the init method. Initialization is covered in Creating and Initializing a Servlet.
2. Invokes the service method, passing request and response objects. Service methods are discussed in Writing Service Methods.

3. The web container may determine that a servlet should be removed from service (for example, when a container wants to reclaim memory resources or when it is being shut down). In such a case, the container calls the destroy method of the Servlet interface. In this method, you release any resources the servlet is using and save any persistent state. The destroy method releases the database object created in the init method .

A servlet’s service methods should all be complete when a servlet is removed. The server tries to ensure this by calling the destroy method only after all service requests have returned or after a server-specific grace period, whichever comes first. If your servlet has operations that may run longer than the server’s grace period, the operations could still be running when destroy is called. You must make sure that any threads still handling client requests complete.

DIFFERENT WAYS OF CREATING SERVLET

1. 1By implementing Servlet interface,
2. By inheriting GenericServlet class, (or)
3. By inheriting HttpServlet class

DIFFERENCE BETWEEN GENERIC SERVLET AND HTTP SERVLET

Protocol dependency

A prime difference between GenericServlet and HttpServlet is that the GenericServlet is protocol-independent while the HttpServlet is protocol-dependent.

Service method

Also, in GenericServlet, the service method is abstract. However, in HttpServlet, the service method is non-abstract. Thus, this is another important difference between GenericServlet and HttpServlet.

Request Dispatcher

public interface **RequestDispatcher**

Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server. The servlet container creates the RequestDispatcher object, which is used as a wrapper around a server resource located at a particular path or given by a particular name.

Difference between forward and send redirect

1. First and most important difference between the forward() and sendRedirect() method is that in the case of the forward, redirect happens at the server end and not visible to the client, but in case of sendRedirect(), redirection happens at the client end and it's visible to the client.  
  
2. Another key difference between forward() and sendRedirect() is that forward is marginally faster than the redirect..  
  
3. Third difference between forward() and sendRedirect() method is that in the case of forward() original URL remains intact, while in the case of sendRedirect() browser knows that it's making a new request, so the original [URL](http://java67.blogspot.com/2013/01/difference-between-url-uri-and-urn.html) changes.

4. In the case of forwarding, Any browser reloads of the resulting page will simply repeat the original request, with the original URL. While in the case of sendRedirect() A browser reloads the second URL, will not repeat the original request, but will rather fetch the second URL.  
  
5. In order to use the forward() method, Both resources must be part of the same [context](http://java67.blogspot.com/2012/09/difference-between-servletconfig-and-servletcontext-j2ee-jsp.html)(But the sendRedirect() method can be used to redirect users to resources that are not part of the current context,  
  
6. In the case of forward, Since both resources are part of the same context, the original request context is retained. On the other hand, since send redirect involves a new request, the previous request scope objects, with all of their parameters and attributes are no longer available after a redirect. (Variables will need to be passed by via the session object).

Difference between get and post

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
| Encoding type | application/x-www-form-urlencoded | application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data |
| History | Parameters remain in browser history | Parameters are not saved in browser history |
| Restrictions on data length | Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters) | No restrictions |
| Restrictions on data type | Only ASCII characters allowed | No restrictions. Binary data is also allowed |
| Security | GET is less secure compared to POST because data sent is part of the URL  Never use GET when sending passwords or other sensitive information! | POST is a little safer than GET because the parameters are not stored in browser history or in web server logs |
| Visibility | Data is visible to everyone in the URL | Data is not displayed in the URL |

[04:01 pm] Minakshi Sharma

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="[http://www.springframework.org/schema/beans"](http://www.springframework.org/schema/beans%22)  
xmlns:xsi="[http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance%22)  
xsi:schemaLocation="<http://www.springframework.org/schema/beans>  
[http://www.springframework.org/schema/beans/spring-beans.xsd">](http://www.springframework.org/schema/beans/spring-beans.xsd%22%3e)

<bean id="e" class="com.emp.Employee">  
<property name="eid" value="100"/>  
<property name="ename" value="Mohan"/>  
<property name="salary" value="8900"/>  
</bean>

</beans>