Regularity (2)	Content (4)	Viva-voce (2)	Timely Submission (2)	Total (10)	Dated Sign of Subject Teacher

---

Assignment No. 05 Date of Performance: ……

Title: Understand servlet life cycle, create login page and apply proper validations with appropriate messages using doGet()/ doPost() methods.

Objective: To study the basic concepts, implementation & working of Servlet.

Theory: Servlet is a Java programming language class used to extend the capabilities of a server. Although servlets can respond to any types of requests, they are commonly used to extend the applications hosted by web servers, so they can be thought of as Java applets that run on servers instead of in web browsers. These kinds of servlets are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET.

Servlets are most often used to:

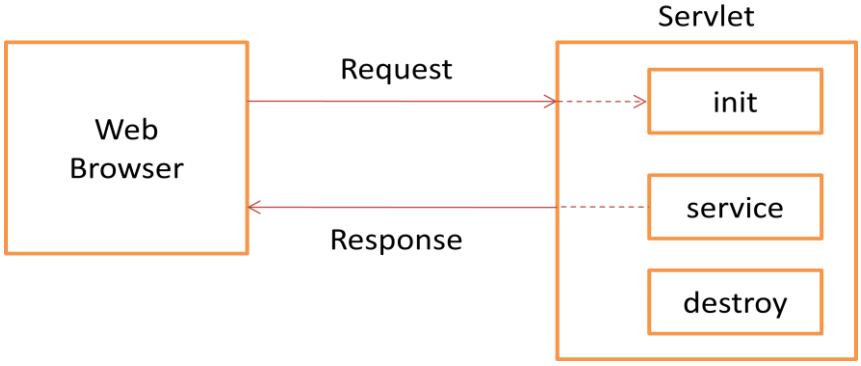
- Process or store data that was submitted from an HTML form.
- Provide dynamic content such as the results of a database query.
- Manage state information that does not exist in the stateless HTTP protocol, such as filling the articles into the shopping cart of the appropriate customer.

Technically speaking, a "servlet" is a Java class in Java EE that conforms to the Java Servlet API, a standard for implementing Java classes which respond to requests. Servlets could in principle communicate over any client–server protocol, but they are most often used with the HTTP protocol. Thus "servlet" is often used as shorthand for "HTTP servlet". Thus, a software developer may use a servlet to add dynamic content to a web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML.

To deploy and run a servlet, a web container must be used. A web container (also known as a servlet container) is essentially the component of a web server that interacts with the servlets. The web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.

Life Cycle of Servlet

Three methods are central to the life cycle of a servlet. These *are init(), service(),* and *destroy()*. They are implemented by every servlet and are invoked at specific times by the server.



- During initialization stage of the servlet life cycle, the web container initializes the servlet instance by calling the init() method, passing an object implementing the javax.servlet.ServletConfig interface. This configuration object allows the servlet to access name-value initialization parameters from the web application.
- After initialization, the servlet instance can service client requests. Each request is serviced in its own separate thread. The web container calls the service() method of the servlet for every request. The service() method determines the kind of request being made and dispatches it to an appropriate method to handle the request. The developer of the servlet must provide an implementation for these methods. If a request is made for a method that is not implemented by the servlet, the method of the parent class is called, typically resulting in an error being returned to the requester.
- Finally, the web container calls the destroy() method that takes the servlet out of service. The destroy() method, like init(), is called only once in the lifecycle of a servlet.

In Java, 2 packages are used to implement the Servlet. Different classes & interfaces defined in these packages are used for implementing the Servlet.

1. javax.servlet
2. javax.servlet.http

The javax.servlet Package

- Contains many classes & interfaces for implementing the servlet
- Out of which, Servlet interface is very important
- All the servlet must implement this interface while implementation of servlet
- Various interfaces in this package are,

Interface	Description
Servlet	Defines all the life cycle methods
ServletConfig	Obtains initialization parameters
ServletContext	Used to log the events
ServletRequest	Useful in reading the data from client request
ServletResponse	Useful in writing the data to client response

- Various classes in this package are,

Class	Description
GenericServlet	Implements Servlet & ServletConfig interfaces
ServletInputStream	Provides input stream for reading client's request
ServletOutputStream	Provides output stream for writing client's response
ServletException	Used to raise the exception when error occurs

The javax.servlet.http Package

- The javax.servlet.http package contains a number of classes and interfaces
- These classes and interfaces describe and define the contracts between a servlet class running under the HTTP protocol and the runtime environment provided for an instance of such a class by a conforming servlet container.

- Various interfaces in this package are,

Interface	Description
HttpServletRequest	Extends the ServletRequest interface to provide request information for HTTP servlets.
HttpServletResponse	Extends the ServletResponse interface to provide HTTP-specific functionality in sending a response.
HttpSession	Provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.
HttpSessionActivationListener	Objects that are bound to a session may listen to container events notifying them that sessions will be passivated and that session will be activated.
HttpSessionAttributeListener	This listener interface can be implemented in order to get notifications of changes to the attribute lists of sessions within this web application.
HttpSessionBindingListener	Causes an object to be notified when it is bound to or unbound from a session.
HttpSessionContext	Deprecated. As of Java(tm) Servlet API 2.1 for security reasons, with no replacement.
HttpSessionListener	Implementations of this interface are notified of changes to the list of active sessions in a web application.

- Various classes in this package are,

Class	Description
Cookie	Creates a cookie, a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server.
HttpServlet	Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site.
HttpServletRequestWrapper	Provides a convenient implementation of the HttpServletRequest interface that can be subclassed by developers wishing to adapt the request to a Servlet.
HttpServletResponseWrapper	Provides a convenient implementation of the HttpServletResponse interface that can be subclassed by developers wishing to adapt the response from a Servlet.
HttpSessionBindingEvent	Events of this type are either sent to an object that implements HttpSessionBindingListener when it is bound or unbound from a session, or to a HttpSessionAttributeListener that has been configured in the deployment descriptor when any attribute is bound, unbound or replaced in a session.
HttpSessionEvent	This is the class representing event notifications for changes to sessions within a web application.
HttpUtils	Deprecated. As of Java(tm) Servlet API 2.3.

GET Method:

- The GET method sends the encoded user information appended to the page request.
- The page and the encoded information are separated by the ? character as follows:

```
http://www.test.com/hello?key1=value1&key2=value2
```

- The GET method is the default method to pass information from browser to web server and it produces a long string that appears in your browser's Location:box.
- Never use GET method if you have password or other sensitive information to pass to a server.
- The GET method has size limitation: only 1024 characters can be in a request string.
- Servlet handles this type of requests using **doGet()** method.

POST Method:

- Generally, more reliable method of passing information to a backend program is the POST method.
- This packages the information in exactly the same way as GET methods, but instead of sending it as a text string after a ? in the URL, it sends it as a separate message.
- This message comes to the backend program in the form of the standard input which you can parse and use for your processing.
- Servlet handles this type of requests using **doPost()** method.

Reading Form Data using Servlet:

- Servlets handles form data & parsing it automatically using following different methods depending on the situation:
    - **getParameter():** You call request.getParameter() method to get the value of a form parameter.
    - **getParameterValues():** Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
    - **getParameterNames():** Call this method if you want a complete list of all parameters in the current request.

Conclusion: Thus, we have studied servlet life cycle, its basics, doGet() & doPost() method for the development of the website.
