What are Servlets?

Java Servlets are programs that run on a Web or Application server and act as a middle layer between a requests coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

Servlets Packages

Java Servlets are Java classes run by a web server that has an interpreter that supports the Java Servlet specification.

Servlets can be created using the **javax.servlet** and **javax.servlet.http**packages, which are a standard part of the Java's enterprise edition, an expanded version of the Java class library that supports large-scale development projects.

These classes implement the Java Servlet and JSP specifications. At the time of writing this tutorial, the versions are Java Servlet 2.5 and JSP 2.1.

Java servlets have been created and compiled just like any other Java class. After you install the servlet packages and add them to your computer's Classpath, you can compile servlets with the JDK's Java compiler or any other current compiler.

1. Servlets - Life Cycle

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

- The servlet is initialized by calling the **init()** method.
- The servlet calls **service()** method to process a client's request.
- The servlet is terminated by calling the **destroy()** method.
- Finally, servlet is garbage collected by the garbage collector of the JVM.

Now let us discuss the life cycle methods in detail.

The init() Method

The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations, just as with the init method of applets.

The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.

When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet.

The init method definition looks like this –

```
public void init() throws ServletException {
  // Initialization code...
}
```

The service() Method

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. Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

Here is the signature of this method –

```
public void service(ServletRequest request, ServletResponse response)
  throws ServletException, IOException {
}
```

The service () method is called by the container and service method invokes doGet, doPost, doPut, doDelete, etc. methods as appropriate. So you have nothing to do with service() method but you override either doGet() or doPost() depending on what type of request you receive from the client.

The doGet() and doPost() are most frequently used methods with in each service request. Here is the signature of these two methods.

The doGet() Method

A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
  // Servlet code
}
```

The doPost() Method

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
  // Servlet code
}
```

The destroy() Method

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. The destroy method definition looks like this –

```
public void destroy() {
  // Finalization code...
}
```

2. Servlets - Examples

Servlets are Java classes which service HTTP requests and implement the **javax.servlet.Servlet** interface. Web application developers typically write servlets that extend javax.servlet.http.HttpServlet, an abstract class that implements the Servlet interface and is specially designed to handle HTTP requests.

Sample Code

Following is the sample source code structure of a servlet example to show Hello World –

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// Extend HttpServlet class
public class HelloWorld extends HttpServlet {

  private String message;

  public void init() throws ServletException {
    // Do required initialization
    message = "Hello World";
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    // Set response content type
    response.setContentType("text/html");

    // Actual logic goes here.
    PrintWriter out = response.getWriter();
    out.println("<h1>" + message + "</h1>");
  }

  public void destroy() {
    // do nothing.
  }
}
```

Compiling a Servlet

Let us create a file with name HelloWorld.java with the code shown above. Place this file at C:\ServletDevel (in Windows) or at /usr/ServletDevel (in Unix). This path location must be added to CLASSPATH before proceeding further.

Assuming your environment is setup properly, go in **ServletDevel** directory and compile HelloWorld.java as follows –

```
$ javac HelloWorld.java
```

If the servlet depends on any other libraries, you have to include those JAR files on your CLASSPATH as well. I have included only servlet-api.jar JAR file because I'm not using any other library in Hello World program.

This command line uses the built-in javac compiler that comes with the Sun Microsystems Java Software Development Kit (JDK). For this command to work properly, you have to include the location of the Java SDK that you are using in the PATH environment variable.

If everything goes fine, above compilation would produce **HelloWorld.class**file in the same directory. Next section would explain how a compiled servlet would be deployed in production.

Servlet Deployment

By default, a servlet application is located at the path <Tomcat-installationdirectory>/webapps/ROOT and the class file would reside in <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes.

If you have a fully qualified class name of **com.myorg.MyServlet**, then this servlet class must be located in WEB-INF/classes/com/myorg/MyServlet.class.

For now, let us copy HelloWorld.class into <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes and create following entries in **web.xml** file located in <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/

```
<servlet>
  <servlet-name>HelloWorld</servlet-name>
  <servlet-class>HelloWorld</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>HelloWorld</servlet-name>
  <url-pattern>/HelloWorld</url-pattern>
</servlet-mapping>
```

Above entries to be created inside <web-app>...</web-app> tags available in web.xml file. There could be various entries in this table already available, but never mind.

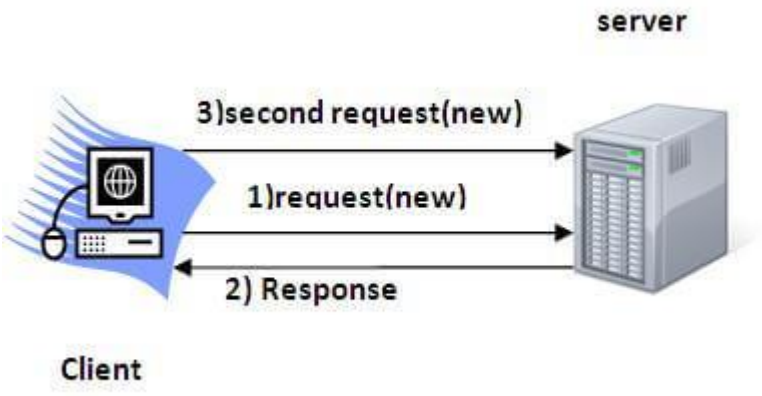
You are almost done, now let us start tomcat server using <Tomcat-installationdirectory>\bin\startup.bat (on Windows) or <Tomcat-installationdirectory>/bin/startup.sh (on Linux/Solaris etc.) and finally type **http://localhost:8080/HelloWorld** in the browser's address box. If everything goes fine, you would get the following result

Session Tracking in Servlets

Session simply means a particular interval of time.
Session Tracking is a way to maintain state (data) of an user. It is also known as session management in servlet.

Http protocol is a stateless so we need to maintain state using session tracking techniques. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.

HTTP is stateless that means each request is considered as the new request. It is shown in the figure given below:



Why use Session Tracking?
To recognize the user It is used to recognize the particular user.

Session Tracking Techniques
There are four techniques used in Session tracking:

- Cookies
- Hidden Form Field
- URL Rewriting
- HttpSession

Servlet Interface

Servlet interface provides commonbehaviorto all the servlets.Servlet interface defines methods that all servlets must implement.
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.

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

Servlet Example By Implementing Servlet Interface

```
import java.io.*;
import javax.servlet.*;

public class First implements Servlet{
ServletConfig config=null;

public void init(ServletConfig config){
this.config=config;
System.out.println("servlet is initialized");
}
```

```
public void service(ServletRequest req,ServletResponse res)
throws IOException,ServletException{

res.setContentType("text/html");

PrintWriter out=res.getWriter();
out.print("<html><body>");
out.print("<b>hello simple servlet</b>");
out.print("</body></html>");

}
public void destroy(){System.out.println("servlet is destroyed");}
public ServletConfig getServletConfig(){return config;}
public String getServletInfo(){return "copyright 2007-1010";}

}
```

ServletConfig Interface

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.

Methods of ServletConfig interface

1. **public String getInitParameter(String name):**Returns the parameter value for the specified parameter name.
2. **public Enumeration getInitParameterNames():**Returns an enumeration of all the initialization parameter names.
3. **public String getServletName():**Returns the name of the servlet.
4. **public ServletContext getServletContext():**Returns an object of ServletContext.

How to get the object of ServletConfig

1. **getServletConfig() method** of Servlet interface returns the object of ServletConfig.

Syntax of getServletConfig() method

```
public ServletConfig getServletConfig();
```

Example of getServletConfig() method

```
ServletConfig config=getServletConfig();
//Now we can call the methods of ServletConfig interface
```

Syntax to provide the initialization parameter for a servlet

The init-param sub-element of servlet is used to specify the initialization parameter for a servlet.

```
<web-app>
  <servlet>
    ......

    <init-param>
      <param-name>parametername</param-name>
      <param-value>parametervalue</param-value>
    </init-param>
    ......
  </servlet>
</web-app>
```

Example of ServletConfig to get initialization parameter

In this example, we are getting the one initialization parameter from the web.xml file and printing this information in the servlet.

DemoServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class DemoServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {

   response.setContentType("text/html");
   PrintWriter out = response.getWriter();

   ServletConfig config=getServletConfig();
   String driver=config.getInitParameter("driver");
   out.print("Driver is: "+driver);
```

```
    out.close();
    }

}
```

web.xml

```
<web-app>

<servlet>
<servlet-name>DemoServlet</servlet-name>
<servlet-class>DemoServlet</servlet-class>

<init-param>
<param-name>driver</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</init-param>

</servlet>

<servlet-mapping>
<servlet-name>DemoServlet</servlet-name>
<url-pattern>/servlet1</url-pattern>
</servlet-mapping>

</web-app>
```

ServletContext Interface

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.

Usage of ServletContext Interface

There can be a lot of usage of ServletContext object. Some of them are as follows:

1. The object of ServletContext provides an interface between the container and servlet.
2. The ServletContext object can be used to get configuration information from the web.xml file.
3. The ServletContext object can be used to set, get or remove attribute from the web.xml file.
4. The ServletContext object can be used to provide inter-application communication.

Commonly used methods of ServletContext interface

1. **public String getInitParameter(String name):**Returns the parameter value for the specified parameter name.
2. **public Enumeration getInitParameterNames():**Returns the names of the context's initialization parameters.
3. **public void setAttribute(String name,Object object):**sets the given object in the application scope.
4. **public Object getAttribute(String name):**Returns the attribute for the specified name.
5. **public Enumeration getInitParameterNames():**Returns the names of the context's initialization parameters as an Enumeration of String objects.
6. **public void removeAttribute(String name):**Removes the attribute with the given name from the servlet context.

Example of ServletContext to get the initialization parameter

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class DemoServlet extends HttpServlet{
public void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException
{
res.setContentType("text/html");
PrintWriter pw=res.getWriter();

//creating ServletContext object
ServletContext context=getServletContext();

//Getting the value of the initialization parameter and printing it
String driverName=context.getInitParameter("dname");
pw.println("driver name is="+driverName);

pw.close();
```

```
}}
```

web.xml

```
<web-app>

<servlet>
<servlet-name>sonoojaiswal</servlet-name>
<servlet-class>DemoServlet</servlet-class>
</servlet>

<context-param>
<param-name>dname</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</context-param>

<servlet-mapping>
<servlet-name>sonoojaiswal</servlet-name>
<url-pattern>/context</url-pattern>
</servlet-mapping>

</web-app>
```

ServletRequest Interface

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

Methods of ServletRequest interface

There are many methods defined in the ServletRequest interface. Some of them are as follows:

Method	Description
public String getParameter(String name)	is used to obtain the value of a parameter by name.
public String[] getParameterValues(String name)	returns an array of String containing all values of given parameter name. It is mainly used to obtain values of a Multi select list box.
java.util.Enumeration getParameterNames()	returns an enumeration of all of the request parameter names.

public int getContentLength()	Returns the size of the request entity data, or -1 if not known.
public String getCharacterEncoding()	Returns the character set encoding for the input of this request.
public String getContentType()	Returns the Internet Media Type of the request entity data, or null if not known.
public ServletInputStream getInputStream() throws IOException	Returns an input stream for reading binary data in the request body.
public abstract String getServerName()	Returns the host name of the server that received the request.
public int getServerPort()	Returns the port number on which this request was received.

Example of ServletRequest to display the name of the user

index.html

```
<form action="welcome" method="get">
Enter your name<input type="text" name="name"><br>
<input type="submit" value="login">
</form>
```

DemoServ.java

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class DemoServ extends HttpServlet{
public void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException
{
res.setContentType("text/html");
PrintWriter pw=res.getWriter();

String name=req.getParameter("name");//will return value
pw.println("Welcome "+name);

pw.close();
}}
```

1. How constructor can be used for a servlet?
   a) Initialization
   b) Constructor function
   c) Initialization and Constructor function
   d) Setup() method
   Answer: c
   Explanation: We cannot declare constructors for interface in Java. This means we cannot enforce this requirement to any class which implements Servlet interface.
   Also, Servlet requires ServletConfig object for initialization which is created by container.

2. Can servlet class declare constructor with ServletConfig object as an argument?

a) True

b) False

Answer: b

Explanation: ServletConfig object is created after the constructor is called and before init() is called. So, servlet init parameters cannot be accessed in the constructor.

3. What is the difference between servlets and applets?
i. Servlets execute on Server; Applets execute on browser
ii. Servlets have no GUI; Applet has GUI
iii. Servlets creates static web pages; Applets creates dynamic web pages
iv. Servlets can handle only a single request; Applet can handle multiple requests
a) i, ii, iii are correct
b) i, ii are correct
c) i, iii are correct
d) i, ii, iii, iv are correct
Answer: b
Explanation: Servlets execute on Server and doesn't have GUI. Applets execute on browser and has GUI.

5. Which of the following code is used to get an attribute in a HTTP Session object in servlets?
   a) session.getAttribute(String name)
   b) session.alterAttribute(String name)
   c) session.updateAttribute(String name)
   d) session.setAttribute(String name)

Answer: a
Explanation: session has various methods for use.

6. Which of the following code retrieves the body of the request as binary data?
   a) DataInputStream data = new InputStream()
   b) DataInputStream data = response.getInputStream()
   c) DataInputStream data = request.getInputStream()
   d) DataInputStream data = request.fetchInputStream()
   Answer: c
   Explanation: InputStream is an abstract class. getInputStream() retrieves the request in binary data.
7. When destroy() method of a filter is called?
   a) The destroy() method is called only once at the end of the life cycle of a filter
   b) The destroy() method is called after the filter has executed doFilter method
   c) The destroy() method is called only once at the begining of the life cycle of a filter
   d) The destroyer() method is called after the filter has executed
   Answer: a
   Explanation: destroy() is an end of life cycle method so it is called at the end of life cycle.

8. Which of the following is true about servlets?
   a) Servlets execute within the address space of web server
   b) Servlets are platform-independent because they are written in java
   c) Servlets can use the full functionality of the Java class libraries
   d) Servlets execute within the address space of web server, platform independent and uses the functionality of java class libraries.
   Answer: d
   Explanation: Servlets execute within the address space of a web server. Since it is written in java it is platform independent. The full functionality is available through libraries.
9. Which are the session tracking techniques?
   i. URL rewriting
   ii. Using session object
   iii.Using response object
   iv. Using hidden fields
   v. Using cookies
   vi. Using servlet object
   a) i, ii, iii, vi
   b) i, ii, iv, v
   c) i, vi, iii, v
   d) i, ii, iii, v
   Answer: b
   Explanation: URL rewriting, using session object, using cookies, using hidden fields are session tracking techniques.
10. Which of the following is used for session migration?
    a) Persisting the session in database

b) URL rewriting
c) Create new database connection
d) Kill session from multiple sessions
Answer: a
Explanation: Session migration is done by persisting session in database. It can also be done by storing session in memory on multiple servers.

11. Which of the following is stored at client side?
a) URL rewriting
b) Hidden form fields
c) SSL sessions
d) Cookies
Answer: d
Explanation: Cookies are stored at client side. Hence, it is advantageous in some cases where clients disable cookies.

12. Which of the following leads to high network traffic?
a) URL rewriting
b) Hidden form fields
c) SSL sessions
d) Cookies
Answer: a
Explanation: WRL rewriting requires large data transfer to and from the server which leads to network traffic and access may be slow.

13. Which of the following is not true about session?
a) All users connect to the same session
b) All users have same session variable
c) Default timeout value for session variable is 20 minutes
d) New session cannot be created for a new user
Answer: c
Explanation: Default timeout value for session variable is 20 minutes. This can be changed as per requirement.

14. 6. SessionIDs are stored in cookies.
a) True
b) False
Answer: a
Explanation: SessionIDs are stored in cookies, URLs and hidden form fields.

15. 7. What is the maximum size of cookie?
a) 4 KB
b) 4 MB
c) 4 bytes
d) 40 KB
Answer: a
Explanation: The 4K is the maximum size for the entire cookie,

including name, value, expiry date etc. To support most browsers, it is suggested to keep the name under 4000 bytes, and the overall cookie size under 4093 bytes.

16. How can we invalidate a session?
a) session.discontinue()
b) session.invalidate()
c) session.disconnect()
d) session.falsify()
Answer: b
Explanation: We can invalidate session by calling session.invalidate() to destroy the session.

17. Which method creates unique fields in the HTML which are not shown to the user?
a) User authentication
b) URL writing
c) HTML Hidden field
d) HTML invisible field
Answer: c
Explanation: HTML Hidden field is the simplest way to pass information but it is not secure and a session can be hacked easily.

18. **Which object of HttpSession can be used to view and manipulate information about a session?**

a. session identifier
b. creation time
c. last accessed time
d. All mentioned above
ANSWER: All mentioned above

19. **Which class provides stream to read binary data such as image etc. from the request object?**
a. ServltInputStream
b. ServletOutputStream
c. Both A & B
d. None of the above
ANSWER: ServltInputStream

20. **Which of these ways used to communicate from an applet to servlet?**
a. RMI Communication
b. HTTP Communication
c. Socket Communication
d. All mentioned above
ANSWER: All mentioned above

21. **Which methods are used to bind the objects on HttpSession instance and get the objects?**

a. setAttribute
b. getAttribute
c. Both A & B
d. None of the above
ANSWER: Both A & B

22. **What type of servlets use these methods doGet(), doPost(),doHead, doDelete(), doTrace()?**
a. Genereic Servlets
b. HttpServlets
c. All of the above
d. None of the above
ANSWER: HttpServlets

23. **Which cookie it is valid for single session only and it is removed each time when the user closes the browser?**
a. Persistent cookie
b. Non-persistent cookie
c. All the above
d. None of the above
ANSWER: Non-persistent cookie

24. **Servlets handle multiple simultaneous requests by using threads.**
a. True
b. False
ANSWER: True

25. **What is the lifecycle of a servlet?**
a. Servlet class is loaded
b. Servlet instance is created
c. init,Service,destroy method is invoked
d. All mentioned above
ANSWER: All mentioned above

Assignment Questions

Q 1 - Which of the following is true about servlets?

A - Servlets execute within the address space of a Web server.

B - Servlets are platform-independent because they are written in Java.

C - The full functionality of the Java class libraries is available to a servlet.

D - All of the above.

Q 2 - Which of the following is true about destroy() method of servlet?

A - After the destroy() method is called, the servlet object is marked for garbage collection.

B - The servlet is terminated by calling the destroy() method.

C - Both of the above.

D - None of the above.

Q 3 - Which of the following code is used to get PrintWriter object in servlet?

A - response.getWriter()

B - request.getWriter()

C - new PrintWriter()

D - None of the above.

Q 4 - Which of the following code retrieves any extra path information associated with the URL the client sent?

A - Header.getPathInfo()

B - response.getPathInfo()

C - request.getPathInfo()

D - None of the above.

Q 5 - Which of the following code returns the port number on which this request was received?

A - response.getServerPort()

B - request.getServerPort()

C - Header.getServerPort()

D - None of the above.

Q 6 - Which of the following code can be used to add a header with the given name and integer value?

A - request.addHeader(name,value)

B - response.addIntHeader(name,value)

C - Header.addDateHeader(name,value)

D - None of the above.

Q 7 - Which of the following is true about filters?

A - Servlet Filters are Java classes that can be used to intercept requests from a client before they access a resource at back end.

B - Servlet Filters are Java classes that can be used to manipulate responses from server before they are sent back to the client.

C - Both of the above.

D - None of the above.

Q 8 - Which element of web.xml is used to specify the error handler in servlets?

A - error-page

B - error-handler

C - exception

D - exception-handler

Q 9 - Which of the following code sends a cookie in servlet?

A - response.addCookie(cookie);

B - response.sendCookie(cookie);

C - response.createCookie(cookie);

D - None of the above

Q 10 - Which of the following code is used to get language name in servlets?

A - response.getDisplayLanguage()

B - Locale.getDisplayLanguage()

C - request.getDisplayLanguage()

D - None of the above.