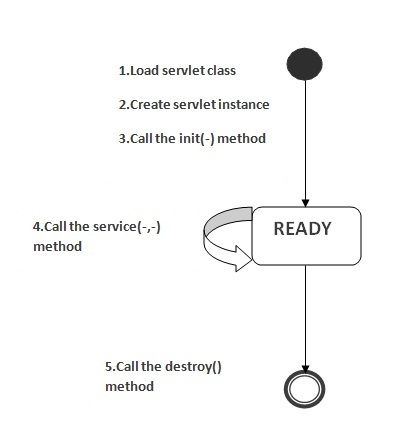
Life Cycle of a Servlet (Servlet Life Cycle)

[Life Cycle of a Servlet](https://www.javatpoint.com/life-cycle-of-a-servlet)

* 1. [Servlet class is loaded](https://www.javatpoint.com/life-cycle-of-a-servlet#servletlifecycle1)
  2. [Servlet instance is created](https://www.javatpoint.com/life-cycle-of-a-servlet#servletlifecycle2)
  3. [init method is invoked](https://www.javatpoint.com/life-cycle-of-a-servlet#servletlifecycle3)
  4. [service method is invoked](https://www.javatpoint.com/life-cycle-of-a-servlet#servletlifecycle4)
  5. [destroy method is invoked](https://www.javatpoint.com/life-cycle-of-a-servlet#servletlifecycle5)

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.



As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

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. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below: |

1. **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:

1. **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:

1. **public** **void** destroy()

Servlet interview questions

There is a list of 30 servlet interview questions for beginners and professionals. If you know any servlet interview question that has not been included here, kindly post your question in the Ask Question section.

Note: **Index file should be inside webcontent folder**

1) How many objects of a servlet is created?

Only one object at the time of first request by servlet or web container.

### 4) Who is responsible to create the object of servlet?

The web container or servlet container.

### 5) When servlet object is created?

At the time of first request.

### 6) What is difference between Get and Post method?

|  |  |
| --- | --- |
| **Get** | **Post** |
| * + 1. Limited amount of data can be sent because data is sent in header.255 bytes is the safest length | Large amount of data can be sent because data is sent in body. |
| 2) Not Secured because data is exposed in URL bar. | Secured because data is not exposed in URL bar. |
| 3) Can be bookmarked | Cannot be bookmarked |
| 4) Idempotent | Non-Idempotent |
| 5) It is more efficient and used than Post | It is less efficient and used |

[more details...](https://www.javatpoint.com/http-request)

### 7) What is difference between PrintWriter and ServletOutputStream?

PrintWriter is a character-stream class where as ServletOutputStream is a byte-stream class. The PrintWriter class can be used to write only character-based information whereas ServletOutputStream class can be used to write primitive values as well as character-based information.

### 8) What is difference between GenericServlet and HttpServlet?

The GenericServlet is protocol independent whereas HttpServlet is HTTP protocol specific. HttpServlet provides additional functionalities such as state management etc.

// public abstract class javax.servlet.http.HttpServlet extends javax.servlet.GenericServlet {

### 9) What is servlet collaboration?

When one servlet communicates to another servlet, it is known as servlet collaboration. There are many ways of servlet collaboration:

* RequestDispacher interface
* sendRedirect() method etc.

### 10) What is the purpose of RequestDispatcher Interface?

The RequestDispacher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interceptor can also be used to include the content of antoher resource.

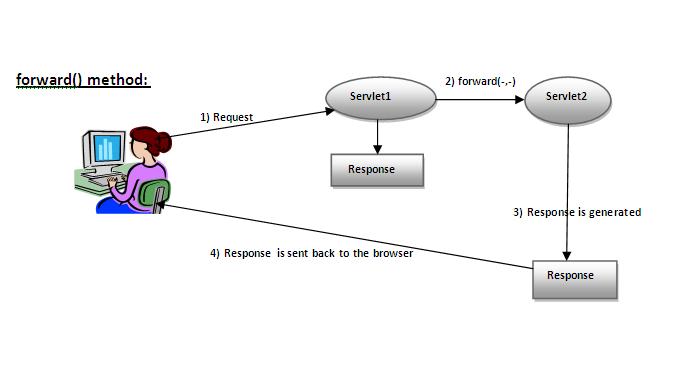
It is one of the way of servlet collaboration.

There are two methods defined in the RequestDispatcher interface.

Methods of RequestDispatcher interface

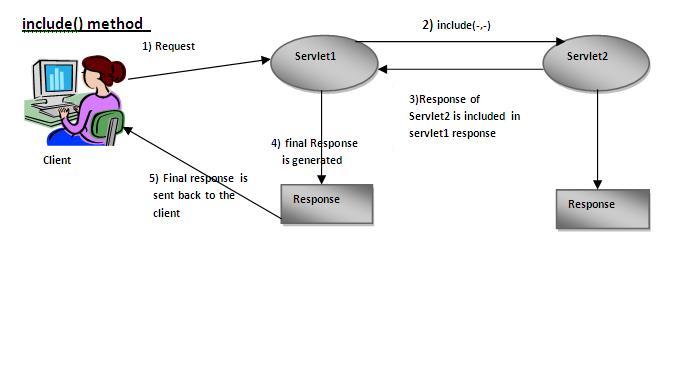
The RequestDispatcher interface provides two methods. They are:

1. **public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

****

1. **public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Includes the content of a resource (servlet, JSP page, or HTML file) in the response.

As you see in the above figure, response of second servlet is sent to the client. Response of the first servlet is not displayed to the user. ----

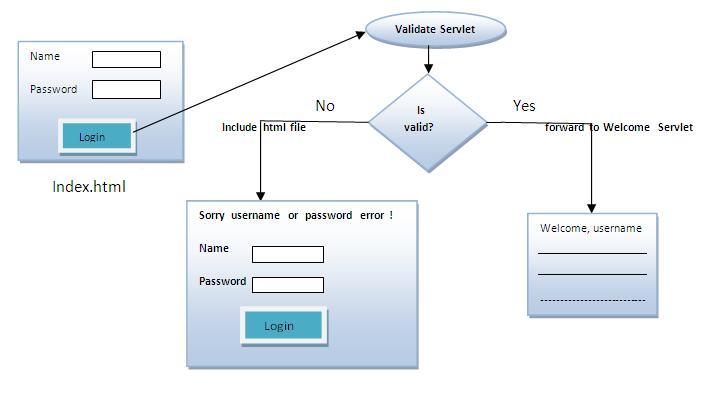


As you can see in the above figure, response of second servlet is included in the response of the first servlet that is being sent to the client.

Example of RequestDispatcher interface

In this example, we are validating the password entered by the user. If password is servlet, it will forward the request to the WelcomeServlet, otherwise will show an error message: sorry username or password error!. In this program, we are cheking for hardcoded information. But you can check it to the database also that we will see in the development chapter. In this example, we have created following files:

* **index.html file:** for getting input from the user.
* **Login.java file:** a servlet class for processing the response. If password is servet, it will forward the request to the welcome servlet.
* **WelcomeServlet.java file:** a servlet class for displaying the welcome message.
* **web.xml file:** a deployment descriptor file that contains the information about the servlet.



### 11) Can you call a jsp from the servlet?

Yes, one of the way is RequestDispatcher interface for example:

1. RequestDispatcher rd=request.getRequestDispatcher("/login.jsp");
2. rd.forward(request,response);

### 12) Difference between forward() method and sendRedirect() method ?

|  |  |
| --- | --- |
| **forward() method** | **sendRedirect() method** |
| 1) forward() sends the same request to another resource. | 1) sendRedirect() method sends new request always because it uses the URL bar of the browser. |
| 2) forward() method works at server side. | 2) sendRedirect() method works at client side. |
| 3) forward() method works within the server only. | 3) sendRedirect() method works within and outside the server. |

### 13) What is difference between ServletConfig and ServletContext?

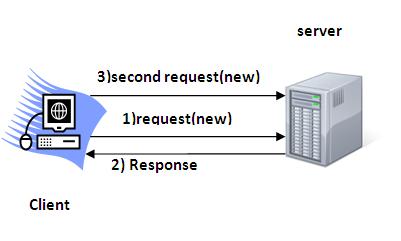
The container creates object of ServletConfig for each servlet whereas object of ServletContext is created for each web application.

**14)** Why use Session Tracking?

**To recognize the user** It is used to recognize the particular user.

### Session simply means a particular interval of time.

Session Tracking is a way to maintain state of an user.Http protocol is a stateless protocol.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:

Session Tracking Techniques

There are four techniques used in Session tracking:

1. **Cookies**
2. **Hidden Form Field**
3. **URL Rewriting**
4. **HttpSession**

### 15) What are Cookies?

A cookie is a small piece of information that is persisted between the multiple client requests. A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

How Cookie works

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



Types of Cookie

There are 2 types of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie
3. Non-persistent cookie
4. It is **valid for single session** only. It is removed each time when user closes the browser.
5. Persistent cookie
6. It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Advantage of Cookies

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

Disadvantage of Cookies

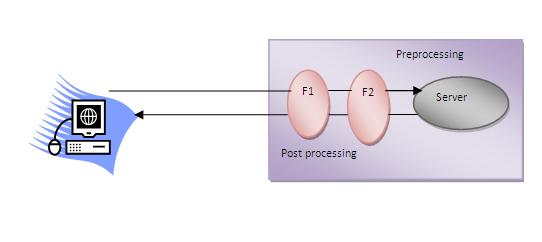
1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

### 16) What is difference between Cookies and HttpSession?

Cookie works at client side whereas HttpSession works at server side.

### 17) What is filter? A filter is an object that is invoked at the preprocessing and postprocessing of a request.

It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc.The **servlet filter is pluggable**, i.e. its entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet. So maintenance cost will be less.



#### Note: Unlike Servlet, One filter doesn't have dependency on another filter.

### Usage of Filter

* recording all incoming requests
* logs the IP addresses of the computers from which the requests originate
* conversion
* data compression
* encryption and decryption
* input validation etc.

### Advantage of Filter

1. Filter is pluggable.
2. One filter don't have dependency onto another resource.
3. Less Maintenance

Filter API

Like servlet filter have its own API. The javax.servlet package contains the three interfaces of Filter API.

1. Filter
2. FilterChain
3. FilterConfig

1) Filter interface

For creating any filter, you must implement the Filter interface. Filter interface provides the life cycle methods for a filter.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void init(FilterConfig config) | init() method is invoked only once. It is used to initialize the filter. |
| public void doFilter(HttpServletRequest request,HttpServletResponse response, FilterChain chain) | doFilter() method is invoked every time when user request to any resource, to which the filter is mapped .It is used to perform filtering tasks. |
| public void destroy() | This is invoked only once when filter is taken out of the service. |

2) FilterChain interface

The object of FilterChain is responsible to invoke the next filter or resource in the chain.This object is passed in the doFilter method of Filter interface.The FilterChain interface contains only one method:

1. **public void doFilter(HttpServletRequest request, HttpServletResponse response):** it passes the control to the next filter or resource.

### 18) How can we perform any action at the time of deploying the project?

By the help of ServletContextListener interface.

### 19) What is the disadvantage of cookies?

It will not work if cookie is disabled from the browser.

[more details...](https://www.javatpoint.com/cookies-in-servlet)

### 20) How can we upload the file to the server using servlet?

One of the way is by MultipartRequest class provided by third party.

Cos jar provide MultipartRequest

### 21) What is load-on-startup in servlet?

# Load on startup in web.xml

The **load-on-startup** element of **web-app** loads the servlet at the time of deployment or server start if value is positive. It is also known as **pre initialization of servlet**.

You can pass positive and negative value for the servlet.

#### Advantage of load-on-startup element

As you know well, servlet is loaded at first request. That means it consumes more time at first request. If you specify the load-on-startup in web.xml, servlet will be loaded at project deployment time or server start. So, it will take **less time** for responding to first request.

#### Passing positive value

If you pass the positive value, the lower integer value servlet will be loaded before the higher integer value servlet. In other words, container loads the servlets in ascending integer value. The 0 value will be loaded first then 1, 2, 3 and so on.

Let's try to understand it by the example given below:

*web.xml*

|  |
| --- |
| 1. **<servlet>** 2. **<servlet-name>**servlet1**</servlet-name>** 3. **<servlet-class>**com.javatpoint.FirstServlet**</servlet-class>** 4. **<load-on-startup>**0**</load-on-startup>** 5. **</servlet>** 7. **<servlet>** 8. **<servlet-name>**servlet2**</servlet-name>** 9. **<servlet-class>**com.javatpoint.SecondServlet**</servlet-class>** 10. **<load-on-startup>**1**</load-on-startup>** 11. **</servlet>** |

There are defined 2 servlets, both servlets will be loaded at the time of project deployment or server start. But, servlet1 will be loaded first then servlet2.

#### Passing negative value

If you pass the negative value, servlet will be loaded at request time, at first request.

### 22) What if we pass negative value in load-on-startup?

It will not affect the container, now servlet will be loaded at first request.

[more details...](https://www.javatpoint.com/load-on-startup)

### 23) What is war file?

A war (web archive) file specifies the web elements. A servlet or jsp project can be converted into a war file. Moving one servlet project from one place to another will be fast as it is combined into a single file.

## Advantage of war file

**saves time**: The war file combines all the files into a single unit. So it takes less time while transferring file from client to server.

## How to deploy the war file?

There are two ways to deploy the war file.

1. By server console panel
2. By manually having the war file in specific folder of server.

If you want to deploy the war file in **apache tomcat** server manually, go to the **webapps** directory of apache tomcat and paste the war file here.

Now, you are able to access the web project through browser.

#### Note: server will extract the war file internally.

### 24) How to create war file?

The war file can be created using jar tool found in jdk/bin directory. If you are using Eclipse or Netbeans IDE, you can export your project as a war file.

To create war file from console, you can write following code.

1. jar -cvf abc.war \*

Now all the files of current directory will be converted into abc.war file.

[more details...](https://www.javatpoint.com/war-file)

### 25) What are the annotations used in Servlet 3?

There are mainly 3 annotations used for the servlet.

1. @WebServlet : for servlet class.
2. @WebListener : for listener class.
3. @WebFilter : for filter class.

### 26) Which event is fired at the time of project deployment and undeployment?

ServletContextEvent.

ServletContextEvent and ServletContextListener

The ServletContextEvent is notified when web application is deployed on the server.

If you want to perform some action at the time of deploying the web application such as creating database connection, creating all the tables of the project etc, you need to implement ServletContextListener interface and provide the implementation of its methods.

Method of ServletContextEvent class

There is only one method defined in the ServletContextEvent class:

1. **public ServletContext getServletContext()**: returns the instance of ServletContext.

Methods of ServletContextListener interface

There are two methods declared in the ServletContextListener interface which must be implemented by the servlet programmer to perform some action such as creating database connection etc.

1. **public void contextInitialized(ServletContextEvent e)**: is invoked when application is deployed on the server.
2. **public void contextDestroyed(ServletContextEvent e)**: is invoked when application is undeployed from the server.

|  |
| --- |
| 1.Insert data data   1. **public** **class** MyListener **implements** ServletContextListener{ 2. **public** **void** contextInitialized(ServletContextEvent event) { 3. **try**{ 4. Class.forName("oracle.jdbc.driver.OracleDriver"); 5. Connection con=DriverManager.getConnection( 6. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle"); 8. //storing connection object as an attribute in ServletContext 9. ServletContext ctx=event.getServletContext(); 10. ctx.setAttribute("mycon", con); 12. }**catch**(Exception e){e.printStackTrace();} 13. } 15. **public** **void** contextDestroyed(ServletContextEvent arg0) {} 16. } |
| 2.fetch data   1. **public** **class** FetchData **extends** HttpServlet { 3. **public** **void** doGet(HttpServletRequest request, HttpServletResponse response) 4. **throws** ServletException, IOException { 6. response.setContentType("text/html"); 7. PrintWriter out = response.getWriter(); 9. **try**{ 10. //Retrieving connection object from ServletContext object 11. ServletContext ctx=getServletContext(); 12. Connection con=(Connection)ctx.getAttribute("mycon"); 14. //retieving data  from emp32 table 15. PreparedStatement ps=con.prepareStatement("select \* from emp32", 16. ResultSet.TYPE\_SCROLL\_SENSITIVE,ResultSet.CONCUR\_UPDATABLE); 18. ResultSet rs=ps.executeQuery(); 19. **while**(rs.next()){ 20. out.print("<br>"+rs.getString(1)+" "+rs.getString(2)); 21. } 23. con.close(); 24. }**catch**(Exception e){e.printStackTrace();} 26. out.close(); 27. } 28. } |

### 27) Which event is fired at the time of session creation and destroy?

HttpSessionEvent.

[more details...](https://www.javatpoint.com/HttpSessionEvent)

### 28) Which event is fired at the time of setting, getting or removing attribute from application scope?

ServletContextAttributeEvent.

### 29) What is the use of welcome-file-list?

It is used to specify the welcome file for the project.

welcome-file-list in web.xml

The **welcome-file-list** element of **web-app**, is used to define a list of welcome files. Its sub element is **welcome-file** that is used to define the welcome file.

A **welcome file** is the file that is invoked automatically by the server, if you don't specify any file name.

By default server looks for the welcome file in following order:

1. welcome-file-list in web.xml
2. index.html
3. index.htm
4. index.jsp

If none of these files are found, server renders 404 error.

If you have specified welcome-file in web.xml, and all the files index.html, index.htm and index.jsp exists, priority goes to welcome-file.

If welcome-file-list entry doesn't exist in web.xml file, priority goes to index.html file then index.htm and at last index.jsp file.

### 30) What is the use of attribute in servlets?

Attribute is a map object that can be used to set, get or remove in request, session or application scope. It is mainly used to share information between one servlet to another.

# Attribute in Servlet

1. [Attribute in Servlet](https://www.javatpoint.com/attribute)
2. [Attribute specific methods](https://www.javatpoint.com/attribute#attributemethod)
3. [Example of ServletContext to set and get attribute](https://www.javatpoint.com/attribute#attributeex)
4. [Difference between ServletConfig and ServletContext](https://www.javatpoint.com/attribute#diffcontext)

An **attribute in servlet** is an object that can be set, get or removed from one of the following scopes:

1. request scope
2. session scope
3. application scope

The servlet programmer can pass informations from one servlet to another using attributes. It is just like passing object from one class to another so that we can reuse the same object again and again.

### Attribute specific methods of ServletRequest, HttpSession and ServletContext interface

|  |
| --- |
| There are following 4 attribute specific methods. They are as follows:   1. **public void setAttribute(String name,Object object):**sets the given object in the application scope. 2. **public Object getAttribute(String name):**Returns the attribute for the specified name. 3. **public Enumeration getInitParameterNames():**Returns the names of the context's initialization parameters as an Enumeration of String objects. 4. **public void removeAttribute(String name):**Removes the attribute with the given name from the servlet context. |

### Example of ServletContext to set and get attribute

|  |
| --- |
| In this example, we are setting the attribute in the application scope and getting that value from another servlet. |

### DemoServlet1.java

|  |
| --- |
|  |

**import** java.io.\*;

**import** javax.servlet.\*;

**import** javax.servlet.http.\*;

**public** **class** DemoServlet1 **extends** HttpServlet{

**public** **void** doGet(HttpServletRequest req,HttpServletResponse res)

{

**try**{

res.setContentType("text/html");

PrintWriter out=res.getWriter();

ServletContext context=getServletContext();

context.setAttribute("company","IBM");

out.println("Welcome to first servlet");

out.println("<a href='servlet2'>visit</a>");

out.close();

}**catch**(Exception e){out.println(e);}

}}

### DemoServlet2.java

**import** java.io.\*;

**import** javax.servlet.\*;

**import** javax.servlet.http.\*;

**public** **class** DemoServlet2 **extends** HttpServlet{

**public** **void** doGet(HttpServletRequest req,HttpServletResponse res)

{

**try**{

res.setContentType("text/html");

PrintWriter out=res.getWriter();

ServletContext context=getServletContext();

String n=(String)context.getAttribute("company");

out.println("Welcome to "+n);

out.close();

}**catch**(Exception e){out.println(e);}

}}

### web.xml

<web-app>

<servlet>

<servlet-name>s1</servlet-name>

<servlet-**class**>DemoServlet1</servlet-**class**>

</servlet>

<servlet-mapping>

<servlet-name>s1</servlet-name>

<url-pattern>/servlet1</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>s2</servlet-name>

<servlet-**class**>DemoServlet2</servlet-**class**>

</servlet>

<servlet-mapping>

<servlet-name>s2</servlet-name>

<url-pattern>/servlet2</url-pattern>

</servlet-mapping>

</web-app>

Difference between ServletConfig and ServletContext

|  |
| --- |
| The servletconfig object refers to the single servlet whereas servletcontext object refers to the whole web application. |