# SERVLET INTERVIEW QUESTIONS

#### How can we create a Servlet?

A Servlet can be created by extending the httpServlet or GenericServlet classes.

#### Explaing what you mean by servlets?

Servlets are server side components that provide a powerful mechanism for developing server side programs. Servlets provide component-based, platform-independent methods for building Web-based applications, without the performance limitations of CGI programs.

#### Explain the life cycle methods of a Servlet?

Servlet interface contains three methods known as life-cycle methods.  
Init , service and destroy.  
First the servlet is constructed, then initialized with the init() method. Any request from client are handled initially by the service() method before delegating to the doXxx() methods in the case of HttpServlet. The servlet is removed from service, destroyed with the destroy() method, then garbaged collected and finalized.

#### Explain the common ways used for session tracking?

Cookies  
SSL sessions  
URL- rewriting

#### Explain the difference between HttpServlet and GenericServlet?

A GenericServlet has a service() method to handle the requests. HttpServlet extends GenericServlet and adds support for doGet (), doPost () methods doPut (), doOptions (), doDelete (), doTrace () methods.

#### Explain the difference between ServletConfig and ServletContext?

ServletContext: It defines a set of methods that a servlet uses to communicate with its container.  
ServletConfig: The object created after a servlet is instantiated and its default constructor is read. It is created to pass initialization information to the servlet.

#### Explain what is a session?

The session is an object used by a servlet to track a user’s interaction with a Web application across multiple HTTP requests. The session is stored on the server.

#### What is the use of ServletContext?

Using ServletContext, We can access data from its environment. Servlet context is common to all Servlets so all Servlets share the information through ServletContext.

#### Is there any way to generate PDF’S dynamically in servlets?

We need to use iText. An open source library for java.

#### What is the difference between using getSession(true) and getSession(false) methods?

getSession(true) – This method will check whether already a session is existing for the user. If a session is existing, it will return that session object, otherwise it will create new session object and return that object.  
getSession(false) – This method will check existence of session. If session exists, then it returns the reference of that session object, if not, this methods will return null.

#### Which code line must be set before any of the lines that use the PrintWriter?

setContentType() method must be set.

#### Which protocol will be used by browser and servlet to communicate?

HTTP

#### In how many ways we can track the sessions?

Method 1) By URL rewriting  
Method 2) Using Session object  
Getting Session form HttpServletRequest object  
HttpSession session = request.getSession(true);  
Get a Value from the session  
session.getValue(session.getId());  
Adding values to session  
cart = new Cart();  
session.putValue(session.getId(), cart);  
At the end of the session, we can inactivate the session by using the following command  
session.invalidate();  
Method 3) Using cookies  
Method 4) Using hidden fields

#### How can you invoke other web resources (or other servlet / Jsp)?

Servlet can invoke other Web resources in two ways: indirect and direct.  
Indirect Way: Servlet will return the resultant HTML to the browser which will point to another Servlet (Web resource)   
Direct Way: We can call another Web resource (Servlet / Jsp) from Servlet program itself, by using RequestDispatcher object.  
You can get this object using getRequestDispatcher (“URL”) method. You can get this object from either a request or a Context.  
Example:

[?](http://www.j2eebrain.com/java-J2ee-servlet-interview-questions.html/2)

|  |  |
| --- | --- |
| 1  2  3  4 | RequestDispatcher dispatcher = request.getRequestDispatcher("/jspsample.jsp");  if (dispatcher != null)  dispatcher.forward(request, response);  } |

#### How can you include other Resources in the Response?

Using include method of a RequestDispatcher object.  
Included WebComponent (Servlet / Jsp) cannot set headers or call any method (for example, setCookie) that affects the headers of the response.  
Example: RequestDispatcher dispatcher = getServletContext().getRequestDispatcher(“/banner”);   
if (dispatcher != null)   
dispatcher.include(request, response);  
}

#### What is servlet mapping?

The servlet mapping defines an association between a URL pattern and a servlet. The mapping is used to map requests to Servlets.

#### Can we use the constructor, instead of init(), to initialize servlet?

Yes. But you will not get the servlet specific things from constructor. The original reason for init() was that old versions of Java couldn’t dynamically invoke constructors with arguments, so there was no way to give the constructor a ServletConfig. That no longer applies, but servlet containers still will only call your no-arg constructor. So you won’t have access to a ServletConfig or ServletContext.

#### Can we write a constructor for servlet?

Yes. But the container will always call the default constructor only. If default constructor is not present, the container will throw an exception.

#### What are the parameters for service method?

ServletRequest and ServletResponse.

#### What are the uses of Servlets?

A servlet can handle multiple requests concurrently, and can synchronize requests. Servlets can forward requests to other servers and servlets. Thus servlets can be used to balance load among several servers.

#### When doGET() method will going to execute?

When we specified method=’GET’ in HTML  
Example : < form name=’SSS’ method=’GET’>

#### When doPOST() method will going to execute?

When we specified method=’POST’ in HTML  
< form name=’SSS’ method=’POST’ >

#### What is the difference between Difference between doGet() and doPost()?

GET Method: Using get method we can able to pass 2K data from HTML  
All data we are passing to Server will be displayed in URL (request string).  
POST Method: In this method we does not have any size limitation.  
All data passed to server will be hidden, User cannot able to see this info  
on the browser.

#### What is the servlet life cycle?

When first request came in for the servlet, Server will invoke init() method of the servlet. Thereafter if any user request the servlet program, Server will directly executes the service() method. When Server want to remove the servlet from pool, then it will execute the destroy() method

#### What is the servlet?

Servlet is a script, which resides and executes on server side, to create dynamic HTML. In servlet programming we will use java language. A servlet can handle multiple requests concurrently

#### What is the architecture of servlet package?

Servlet Interface is the central abstraction. All servlets implements this Servlet  
Interface either directly or indirectly   
(may implement or extend Servlet Interfaces sub classes or sub interfaces)  
Servlet  
|  
Generic Servlet  
|  
HttpServlet ( Class ) — we will extend this class to handle GET / PUT HTTP requests  
|  
MyServlet

#### What is the difference between HttpServlet and GenericServlet?

A GenericServlet has a service() method to handle requests.  
HttpServlet extends GenericServlet added new methods  
doGet()  
doPost()  
doHead()  
doPut()  
doOptions()  
doDelete()  
doTrace() methods  
Both these classes are abstract.

#### What’s the difference between servlets and applets?

Servlets executes on Servers. Applets executes on browser. Unlike applets, however, servlets have no graphical user interface.  
What is the difference between the getRequestDispatcher(String path) ServletRequest interface and ServletContext interface?  
The getRequestDispatcher(String path) method of ServletRequest interface accepts parameter the path to the resource to be included or forwarded to, which can be relative to the request of the calling servlet. If the path begins with a “/” it is interpreted as relative to the current context root.  
The getRequestDispatcher(String path) method of ServletContext interface cannot accepts relative paths. All path must start with a “/” and are interpreted as relative to current context root. If the resource is not available, or if the server has not implemented a RequestDispatcher object for that type of resource, getRequestDispatcher will return null. Your servlet should be prepared to deal with this condition.

### Explain the mechanism by which we can generate PDF’S dynamically in servlets?

This can be achieved using iText. An open source library for java. You can refer SourceForge site for sample servlet examples.

### Explain the difference between the getRequestDispatcher(String path) ServletRequest interface and ServletContext interface?

The getRequestDispatcher(String path) method of ServletRequest interface accepts parameter the path to the resource to be included or forwarded to, which can be relative to the request of the calling servlet. If the path begins with a “/” it is interpreted as relative to the current context root.  
The getRequestDispatcher(String path) method of ServletContext interface cannot accepts relative paths. All path must start with a “/” and are interpreted as relative to current context root. If the resource is not available, or if the server has not implemented a RequestDispatcher object for that type of resource, getRequestDispatcher will return null. Your servlet should be prepared to deal with this condition.

### Explain the way by which you can include other Resources in the Response?

Using include method of a RequestDispatcher object.  
Included WebComponent (Servlet / Jsp) cannot set headers or call any method (for example, setCookie) that affects the headers of the response.  
E.g.:

[?](http://www.j2eebrain.com/java-J2ee-servlet-interview-questions.html/4)

|  |  |
| --- | --- |
| 1  2  3  4 | RequestDispatcher dispatcher = getServletContext().getRequestDispatcher(”/banner”);  if (dispatcher != null)  dispatcher.include(request, response);  } |

### Is there a way you can invoke other web resources (or other servlet / Jsp)?

Yes, there is a mechanism. Servlet can invoke other Web resources in two ways: indirect and direct.  
Indirect Way: Servlet will return the resultant HTML to the browser which will point to another Servlet (Web resource)   
Direct Way: We can call another Web resource (Servlet / Jsp) from Servlet program itself, by using RequestDispatcher object.  
You can get this object using getRequestDispatcher(”URL”) method. You can get this object from either a request or a Context.  
E.g.:

[?](http://www.j2eebrain.com/java-J2ee-servlet-interview-questions.html/4)

|  |  |
| --- | --- |
| 1  2  3  4 | RequestDispatcher dispatcher = request.getRequestDispatcher(”/jspsample.jsp”);  if (dispatcher != null)  dispatcher.forward(request, response);  } |

### What do you mean by preinitialization of a Servlet?

A container does not initialize the servlets as soon as it starts up, it initializes a servlet when it receives a request for that servlet first time. This is called lazy loading. The Servlet specification defines the <load-on-startup> element, which can be specified in the deployment descriptor to make the Servlet container load and initialize the Servlet as soon as it starts up. The process of loading a servlet before any request comes in is called preloading or preinitializing a Servlet.

### Explain the difference in using request.getRequestDispatcher() and context.getRequestDispatcher()?

In request.getRequestDispatcher(path) in order to create it we need to give the relative path of the resource. But in resourcecontext.getRequestDispatcher(path) in order to create it we need to give the absolute path of the resource.

### Explain the difference between using getSession(false) and getSession(true) methods?

getSession(true) – This method will check whether already a session is existing for the user. If a session is existing, it will return that session object, otherwise it will create new session object and return that object.  
getSession(false) – This method will check existence of session. If session exists, then it returns the reference of that session object, if not, this methods will return null.

### Explain on what is servlet mapping?

The servlet mapping defines an association between a URL pattern and a servlet. The mapping is used to map requests to Servlets.

### Is there a way by which we can use constructor, instead of init(), to initialize servlet?

Yes. But you will not get the servlet specific things from constructor. The original reason for init () was that ancient versions of Java couldn’t dynamically invoke constructors with arguments, so there was no way to give the constructor a ServletConfig. That no longer applies, but servlet containers still will only call your no- arg constructor. So you won’t have access to a ServletConfig or ServletContext.

### Which are the different ways you can communicate between servlets?

Below are the different ways of communicating between servlets:-

* Using Request Dispatcher object.
* Sharing resource using “ServletContext ()” object.
* Include response of the resource in the response of the servlet.
* Calling public methods of the resource.
* Servlet chaining.

### Explain the difference between Authentication and authorization?

Authentication is the process the application identifies that you are who. For example when a user logs into an application with a username and password, application checks that the entered credentials against its user data store and responds with success or failure. Authorization, on the other hand, is when the application checks to see if you’re allowed to do something. For instance are you allowed to do delete or modify a resource.application and thus can be used to share data between  
servlets.

### What is Server Side Includes (SSI)? Explain in detail

Server Side Includes (SSI) are commands and directives placed in Web pages that are evaluated by the Web server when the Web page is being served. SSI are not supported by all web servers. So before using SSI read the web server documentation for the support. SSI is useful when you want a small part of the page to be dynamically generated rather than loading the whole page again.

### **Servlet Questions Asked in Interview**

**Question 1: In**web**.**xml**file   <load-on-startup>1</load-on-startup> is defined between <servlet></servlet> tag what does it means.**([Detailed answer](http://javarevisited.blogspot.com/2011/12/load-on-startup-servlet-webxml-example.html))

Ans**:**whenever we request for any servlet the servlet container will initialize the servlet and load it which is defined in our config file called web.xml by default it will not initialize when our context is loaded .defining like this <load-on-startup>1</load-on-startup> is also known as pre-initialization of servlet means now the servlet for which we have defined this tag has been initialized in starting when context is loaded before getting any request. When this servlet question was asked to me in an interview few years back , I was not even aware of this element but this questions pointed me to look DTD of web.xml and understand other elements as well.

**Question 2: How can we create deadlock condition on our servlet?**([Detailed answer](http://javarevisited.blogspot.com/2010/10/what-is-deadlock-in-java-how-to-fix-it.html))

Ans: one simple way to call doPost() method inside doGet() and doGet() method inside doPost() it will create deadlock situation for a servlet. This is rather a simple servlet interview questions but yet tricky if you don’t think of it 

**Question 3: For initializing a servlet can we use a constructor in place of init()?**([Detailed answer](http://java67.blogspot.com/2016/02/can-servlet-have-constructor-in-java.html))

Ans: No, we cannot use constructor for initializing a servlet because for initialization we need an object of ServletConfig using this object we get all the parameter which are defined in deployment descriptor for initializing a servlet and in servlet class we have only default constructor according to older version of java so if we want to pass a Config object we don’t have parametrized constructor and apart from this servlet is loaded and initialized by container so it's a job of container to call the method according to servlet specification they have lifecycle method so init() method is called firstly.

More important Java doesn't allow interfaces to declare constructors. These kinds of *servlet interview questions* are quite popular on service based companies who just want to dig one level more. You can also refer [Programming Interviews Exposed](http://www.amazon.com/dp/1118261364/?tag=javamysqlanta-20) for more of such questions.

**Question 4: Why super.init (config) is the first statement inside init(config) method**. ([Detailed answer](http://javarevisited.blogspot.com/2015/02/constructor-vs-init-method-in-servlet.html))

Ans: This will be the first statement if we are overriding the init(config) method by this way we will store the config object for future reference and we can use by getServletConfig() to get information about config object if will not do this config object will be lost and we have only one way to get config object because servlet pass config object only in init method. Without doing this if we call the ServletConfig method will get **NullPointerException.**

**Question5: Can we call destroy() method inside the init() method is yes what will happen?**(Detailed answer)

Ans: Yes we can call like this but if we have not overridden this method container will call the default method and nothing will happen. After calling this if any we have overridden the method then the code written inside is executed.

**Question 6: How can we refresh servlet on client and server side automatically?**([Detailed answer](http://java67.blogspot.com/2012/10/servlet-jsp-interview-questions-answer-faq-experience.html))

Ans: On the client side we can use Meta HTTP refresh and server side we can use server push.

**Question 7: How can you get the information about one servlet context in another servlet**? (Detailed answer)

Ans: In context object we can set the attribute which we want on another servlet and we can get that attribute using their name on another servlet.

Context.setAttribute (“name”,” value”)

Context.getAttribute (“name”)

**Question 8: Why we need to implement Single Thread model in the case of Servlet.**(Detailed answer)

Ans: In J2EE we can implement our servlet in two different ways either by using:

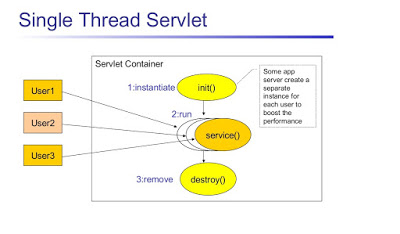
1. Single Thread Model

2. Multithread Model

Depending upon our scenario, if we have implemented single thread means only one instance is going handle one request at a time no two thread will concurrently execute service method of the servlet.

**The example** in banking accounts where sensitive data is handled mostly this scenario was used this interface is deprecated in Servlet API version 2.4.

As the name signifies multi-thread means a servlet is capable of handling multiple requests at the same time. This servlet interview question was quite popular few years back on entry level but now it's losing its shine.

[](https://3.bp.blogspot.com/-mU5exqNhY10/VvSxsvfhYXI/AAAAAAAAFTs/JuIE2-EdbqcId_MUxFlXXtyesXWL1QS0g/s1600/Single+Thread+Servlet.jpg)

**Question 9: What is servlet collaboration?**([Detailed answer](http://java67.blogspot.com/2016/01/6-difference-between-include-directive-and-include-action-in-jsp.html))

Ans communication between two servlets is called servlet collaboration which is achieved by 3 ways.

**1. RequestDispatcher include () and forward() method.**

**2. Using**[**sendRedirect()**](http://javarevisited.blogspot.com/2011/09/sendredirect-forward-jsp-servlet.html)**method of Response object.**

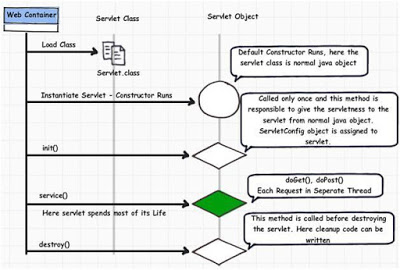
**3. Using servlet Context methods**

**Question 10: What is the difference between ServletConfig and ServletContext?**([Detailed answer](http://java67.blogspot.com/2012/09/difference-between-servletconfig-and-servletcontext-j2ee-jsp.html))

Ans: **ServletConfig** as the name implies provide the information about the configuration of a servlet which is defined inside the web.xml file or we can say deployment descriptor. It’s a specific object for each servlet.

**ServletContext** is an application specific object which is shared by all the servlet belongs to one application in one JVM .this is a single object which represents our application and all the servlet access application specific data using this object.servlet also use their method to communicate with the container.

**Question 11: Explain Servlet Life Cycle in Java EE environment?**  
A picture is worth thousand words, here is a diagram which explains the Servlet life cycle:

[](https://2.bp.blogspot.com/-C7jH3iUVBKc/VvSxXrmNNCI/AAAAAAAAFTw/GcQna15jEqMRmcvoSRbnwlM-rZJcYZI3g/s1600/servlet-life-cycle-in-java.jpg)

**Question 12: What is the difference between HttpServlet and GenericServlet in Servlet API?** ([Answer](http://java67.blogspot.com/2012/12/difference-between-genericservlet-vs-httpservlet-jsp.html))  
GenericServlet provides framework to create a Servlet for any protocol e.g. you can write Servlet to receive content from FTP, SMTP etc, while HttpServlet is built-in Servlet provided by Java for handling HTTP requests. See detailed answer for deep discussion.

These**Servlet interview questions** are good for the quick recap of important concept before appearing on any J2EE interview. Please share if you have come across any other interesting interview question on Servlets.

**1. What is the Servlet?**

A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request- response programming model.

**2. What are the new features added to Servlet 2.5?**

Following are the changes introduced in Servlet 2.5:

* A new dependency on J2SE 5.0
* Support for annotations
* Loading the class
* Several web.xml conveniences
* A handful of removed restrictions
* Some edge case clarifications

**3. What are the uses of Servlet?**

Typical uses for HTTP Servlets include:

* Processing and/or storing data submitted by an HTML form.
* Providing dynamic content, e.g. returning the results of a database query to the client.
* A Servlet can handle multiple request concurrently and be used to develop high performance system
* Managing state information on top of the stateless HTTP, e.g. for an online shopping cart system which manages shopping carts for many concurrent customers and maps every request to the right customer.

**4. What are the advantages of Servlet over CGI?**

Servlets have several advantages over CGI:

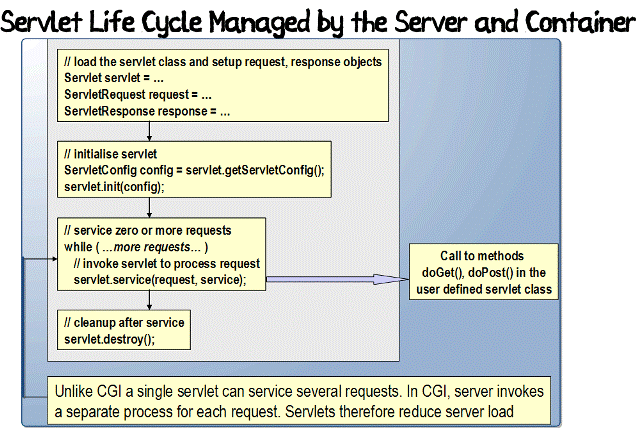
* A Servlet does not run in a separate process. This removes the overhead of creating a new process for each request.
* A Servlet stays in memory between requests. A CGI program (and probably also an extensive runtime system or interpreter) needs to be loaded and started for each CGI request.
* There is only a single instance which answers all requests concurrently. This saves memory and allows a Servlet to easily manage persistent data.
* Several web.xml conveniences
* A handful of removed restrictions
* Some edge case clarifications

**5. What are the phases of the servlet life cycle?**

The life cycle of a servlet consists of the following phases:

* **Servlet class loading**: For each servlet defined in the deployment descriptor of the Web application, the servlet container locates and loads a class of the type of the servlet. This can happen when the servlet engine itself is started, or later when a client request is actually delegated to the servlet.
* **Servlet instantiation**: After loading, it instantiates one or more object instances of the servlet class to service the client requests.
* **Initialization (call the init method)**: After instantiation, the container initializes a servlet before it is ready to handle client requests. The container initializes the servlet by invoking its init() method, passing an object implementing the ServletConfig interface. In the init() method, the servlet can read configuration parameters from the deployment descriptor or perform any other one-time activities, so the init() method is invoked once and only once by the servlet container.
* **Request handling (call the service method)**: After the servlet is initialized, the container may keep it ready for handling client requests. When client requests arrive, they are delegated to the servlet through the service() method, passing the request and response objects as parameters. In the case of HTTP requests, the request and response objects are implementations of HttpServletRequest and HttpServletResponse respectively. In the HttpServlet class, the service() method invokes a different handler method for each type of HTTP request, doGet() method for GET requests, doPost() method for POST requests, and so on.
* **Removal from service (call the destroy method)**: A servlet container may decide to remove a servlet from service for various reasons, such as to conserve memory resources. To do this, the servlet container calls the destroy() method on the servlet. Once the destroy() method has been called, the servlet may not service any more client requests. Now the servlet instance is eligible for garbage collection

The life cycle of a servlet is controlled by the container in which the servlet has been deployed.



**6. Why do we need a constructor in a servlet if we use the init method?**

Even though there is an init method in a servlet which gets called to initialize it, a constructor is still required to instantiate the servlet. Even though you as the developer would never need to explicitly call the servlet's constructor, it is still being used by the container (the container still uses the constructor to create an instance of the servlet). Just like a normal POJO (plain old java object) that might have an init method, it is no use calling the init method if you haven't constructed an object to call it on yet.

**7. How the servlet is loaded?**

A servlet can be loaded when:

* First request is made.
* Server starts up (auto-load).
* There is only a single instance which answers all requests concurrently. This saves memory and allows a Servlet to easily manage persistent data.
* Administrator manually loads.

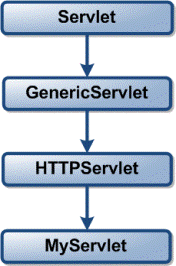
**8. How a Servlet is unloaded?**

A servlet is unloaded when:

* Server shuts down.
* Administrator manually unloads.

**9. What is Servlet interface?**

The central abstraction in the Servlet API is the Servlet interface. All servlets implement this interface, either directly or, more commonly by extending a class that implements it.



*Note: Most Servlets, however, extend one of the standard implementations of that interface, namely* javax.servlet.GenericServlet and javax.servlet.http.HttpServlet.

**10. What is the GenericServlet class?**

GenericServlet is an abstract class that implements the Servlet interface and the ServletConfig interface. In addition to the methods declared in these two interfaces, this class also provides simple versions of the lifecycle methods init and destroy, and implements the log method declared in the ServletContext interface.   
*Note: This class is known as generic servlet, since it is not specific to any protocol.*

**11. What’s the difference between GenericServlet and HttpServlet?**

|  |  |
| --- | --- |
| **GenericServlet** | **HttpServlet** |
| The GenericServlet is an abstract class that is extended by HttpServlet to provide HTTP protocol-specific methods. | An abstract class that simplifies writing HTTP servlets. It extends the GenericServlet base class and provides a framework for handling the HTTP protocol. |
| The GenericServlet does not include protocol-specific methods for handling request parameters, cookies, sessions and setting response headers. | The HttpServlet subclass passes generic service method requests to the relevant doGet() or doPost() method. |
| GenericServlet is not specific to any protocol. | HttpServlet only supports HTTP and HTTPS protocol. |

|  |
| --- |
|  |

**12. Why is HttpServlet declared abstract?**

The HttpServlet class is declared abstract because the default implementations of the main service methods do nothing and must be overridden. This is a convenience implementation of the Servlet interface, which means that developers do not need to implement all service methods. If your servlet is required to handle doGet() requests for example, there is no need to write a doPost() method too.

**13. Can servlet have a constructor?**

One can definitely have constructor in servlet. Even you can use the constructor in servlet for initialization purpose, but this type of approach is not so common. You can perform common operations with the constructor as you normally do. The only thing is that you cannot call that constructor explicitly by the new keyword as we normally do. In the case of servlet, servlet container is responsible for instantiating the servlet, so the constructor is also called by servlet container only.

**14. What are the types of protocols supported by HttpServlet?**

It extends the GenericServlet base class and provides a framework for handling the HTTP protocol. So, HttpServlet only supports HTTP and HTTPS protocol.

**15. What is the difference between doGet() and doPost()?**

|  |  |  |
| --- | --- | --- |
| **#** | **doGet()** | **doPost()** |
| 1 | In doGet() the parameters are appended to the URL and sent along with header information. | In doPost(), on the other hand will (typically) send the information through a socket back to the webserver and it won't show up in the URL bar. |
| 2 | The amount of information you can send back using a GET is restricted as URLs can only be 1024 characters. | You can send much more information to the server this way - and it's not restricted to textual data either. It is possible to send files and even binary data such as serialized Java objects! |
| 3 | doGet() is a request for information; it does not (or should not) change anything on the server. (doGet() should be idempotent) | doPost() provides information (such as placing an order for merchandise) that the server is expected to remember |
| 4 | Parameters are not encrypted | Parameters are encrypted |
| 5 | doGet() is faster if we set the response content length since the same connection is used. Thus increasing the performance | doPost() is generally used to update or post some information to the server.doPost is slower compared to doGet since doPost does not write the content length |
| 6 | doGet() should be idempotent. i.e. doget should be able to be repeated safely many times | This method does not need to be idempotent. Operations requested through POST can have side effects for which the user can be held accountable. |
| 7 | doGet() should be safe without any side effects for which user is held responsible | This method does not need to be either safe |
| 8 | It allows bookmarks. | It disallows bookmarks. |

**16. When to use doGet() and when doPost()?**

Always prefer to use GET (As because GET is faster than POST), except mentioned in the following reason:

* If data is sensitive
* Data is greater than 1024 characters
* If your application don't need bookmarks.

**17. How do I support both GET and POST from the same Servlet?**

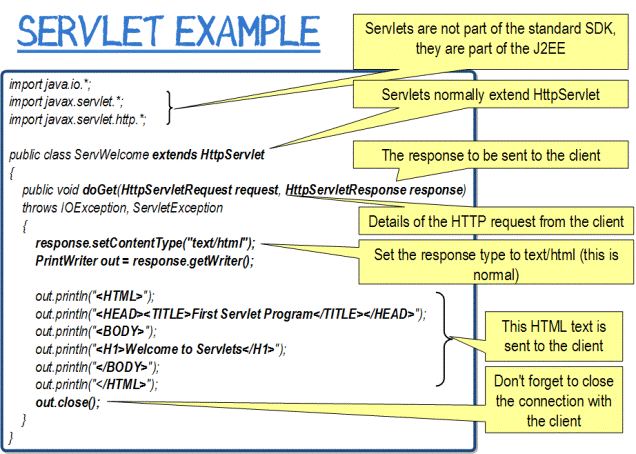
The easy way is, just support POST, then have your doGet method call your doPost method:

**public void doGet(HttpServletRequest request, HttpServletResponse response)  
                        throws ServletException, IOException  
{  
    doPost(request, response);     
}**

**18. Should I override the service() method?**

We never override the service method, since the HTTP Servlets have already taken care of it. The default service function invokes the doXXX() method corresponding to the method of the HTTP request. For example, if the HTTP request method is GET, doGet() method is called by default. A servlet should override the doXXX() method for the HTTP methods that servlet supports. Because HTTP service method check the request method and calls the appropriate handler method, it is not necessary to override the service method itself. Only override the appropriate doXXX() method.

**19. How the typical servlet code look like?**



**20. What is a servlet context object?**

A servlet context object contains the information about the Web application of which the servlet is a part. It also provides access to the resources common to all the servlets in the application. Each Web application in a container has a single servlet context associated with it.

**21. What are the differences between the ServletConfig interface and the ServletContext interface?**

|  |  |
| --- | --- |
| **ServletConfig** | **ServletContext** |
| The ServletConfig interface is implemented by the servlet container in order to pass configuration information to a servlet. The server passes an object that implements the ServletConfig interface to the servlet's init() method. | A ServletContext defines a set of methods that a servlet uses to communicate with its servlet container. |
| There is one ServletConfig parameter per servlet. | There is one ServletContext for the entire webapp and all the servlets in a webapp share it. |
| The param-value pairs for ServletConfig object are specified in the <init-param> within the <servlet> tags in the web.xml file | The param-value pairs for ServletContext object are specified in the <context-param> tags in the web.xml file. |

**22. What’s the difference between forward() and sendRedirect() methods?**

|  |  |
| --- | --- |
| **forward()** | **sendRedirect()** |
| A forward is performed internally by the servlet. | A redirect is a twostep process, where the web application instructs the browser to fetch a second URL, which differs from the original. |
| The browser is completely unaware that it has taken place, so its original URL remains intact. | The browser, in this case, is doing the work and knows that it's making a new request. |
| Any browser reload of the resulting page will simple repeat the original request, with the original URL | A browser reloads of the second URL, will not repeat the original request, but will rather fetch the second URL. |
| Both resources must be part of the same context (Some containers make provisions for cross-context communication but this tends not to be very portable) | This method can be used to redirect users to resources that are not part of the current context, or even in the same domain. |
| Since both resources are part of same context, the original request context is retained | Because this involves a new request, the previous request scope objects, with all of its parameters and attributes are no longer available after a redirect. (Variables will need to be passed by via the session object). |
| Forward is marginally faster than redirect. | redirect is marginally slower than a forward, since it requires two browser requests, not one. |

**23. What is the difference between the include() and forward() methods?**

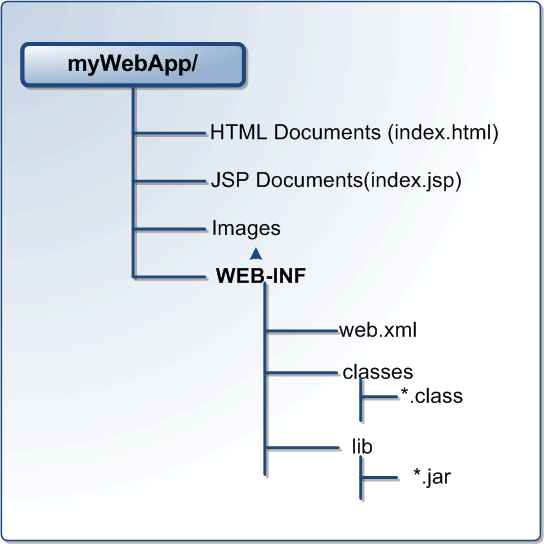
|  |  |
| --- | --- |
| **include()** | **forward()** |
| The RequestDispatcher include() method inserts the contents of the specified resource directly in the flow of the servlet response, as if it were part of the calling servlet. | The RequestDispatcher forward() method is used to show a different resource in place of the servlet that was originally called. |
| If you include a servlet or JSP document, the included resource must not attempt to change the response status code or HTTP headers, any such request will be ignored. | The forwarded resource may be another servlet, JSP or static HTML document, but the response is issued under the same URL that was originally requested. In other words, it is not the same as a redirection. |
| The include() method is often used to include common "boilerplate" text or template markup that may be included by many servlets. | The forward() method is often used where a servlet is taking a controller role; processing some input and deciding the outcome by returning a particular response page. |

|  |
| --- |
|  |

**24. What’s the use of the servlet wrapper classes?**

The HttpServletRequestWrapper and HttpServletResponseWrapper classes are designed to make it *easy* for developers to create custom implementations of the servlet request and response types. The classes are constructed with the standard HttpServletRequest and HttpServletResponse instances respectively and their default behavior is to pass all method calls directly to the underlying objects.

**25. What is the directory structure of a WAR file?**



**26. What is a deployment descriptor?**

A deployment descriptor is an XML document with an .xml extension. It defines a component's deployment settings. It declares transaction attributes and security authorization for an enterprise bean. The information provided by a deployment descriptor is declarative and therefore it can be modified without changing the source code of a bean.  
The JavaEE server reads the deployment descriptor at run time and acts upon the component accordingly.

**27. What is the difference between the getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface and javax.servlet.ServletContext interface?**

|  |  |
| --- | --- |
| **ServletRequest.getRequestDispatcher(String path)** | **ServletContext.getRequestDispatcher(String path)** |
| The getRequestDispatcher(String path) method javax.servlet.ServletRequest interface accepts parameter the path to the resource to be included or forwarded to, which can be relative to the request of the calling servlet. If the path begins with a “/” it is interpreted as relative to the current context root. | The getRequestDispatcher(String path) method of javax.servlet.ServletContext interface cannot accept relative paths. All path must start with a “/” and are   interpreted as relative to current context root. |

|  |
| --- |
|  |

**28. What is preinitialization of a servlet?**

A container does not initialize the servlets as soon as it starts up, it initializes a servlet when it receives a request for that servlet first time. This is called lazy loading. The servlet specification defines the element, which can be specified in the deployment descriptor to make the servlet container load and initialize the servlet as soon as it starts up. The process of loading a servlet before any request comes in is called preloading or preinitializing a servlet.

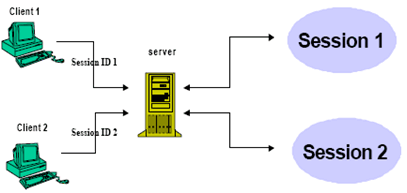
**29. What is the <load-on-startup> element?**

The <load-on-startup> element of a deployment descriptor is used to load a servlet file when the server starts instead of waiting for the first request. It is also used to specify the order in which the files are to be loaded. The <load-on-startup> element is written in the deployment descriptor as follows:

<Servlet>  
   <servlet-name>Servlet Name</servlet-name>  
   <servlet-class>Class Name</servlet-class>  
   <load-on-startup>1</load-on-startup>  
</servlet>

**30. What is session?**

A session refers to all the requests that a single client might make to a server in the course of viewing any pages associated with a given application. Sessions are specific to both the individual user and the application. As a result, every user of an application has a separate session and has access to a separate set of session variables.



**31. What is Session Tracking?**

Session tracking is a mechanism that servlets use to maintain state about a series of requests from the same user (that is, requests originating from the same browser) across some period of time.

**32. What is the need of Session Tracking in web application?**

HTTP is a stateless protocol i.e., every request is treated as new request. For web applications to be more realistic they have to retain information across multiple requests. Such information which is part of the application is referred as "state". To keep track of this state we need session tracking.   
  
Typical example: Putting things one at a time into a shopping cart, then checking out--each page request must somehow be associated with previous requests.

**33. What are the types of Session Tracking?**

Sessions need to work with all web browsers and take into account the users security preferences. Therefore there are a variety of ways to send and receive the identifier:

* **URL rewriting:**URL rewriting is a method of session tracking in which some extra data (session ID) is appended at the end of each URL. This extra data identifies the session. The server can associate this session identifier with the data it has stored about that session. This method is used with browsers that do not support cookies or where the user has disabled the cookies.
* **Hidden Form Fields:** Similar to URL rewriting. The server embeds new hidden fields in every dynamically generated form page for the client. When the client submits the form to the server the hidden fields identify the client.
* **Cookies:**Cookie is a small amount of information sent by a servlet to a Web browser. Saved by the browser, and later sent back to the server in subsequent requests. A cookie has a name, a single value, and optional attributes. A cookie's value can uniquely identify a client.
* **Secure Socket Layer (SSL) Sessions:** Web browsers that support Secure Socket Layer communication can use SSL's support via HTTPS for generating a unique session key as part of the encrypted conversation.

**34. How do I use cookies to store session state on the client?**

In a servlet, the HttpServletResponse and HttpServletRequest objects passed to method HttpServlet.service() can be used to create cookies on the client and use cookie information transmitted during client requests. JSPs can also use cookies, in scriptlet code or, preferably, from within custom tag code.

* To set a cookie on the client, use the addCookie() method in class HttpServletResponse. Multiple cookies may be set for the same request, and a single cookie name may have multiple values.
* To get all of the cookies associated with a single HTTP request, use the getCookies() method of class HttpServletRequest

**35. What are some advantages of storing session state in cookies?**

* Cookies are usually persistent, so for low-security sites, user data that needs to be stored long-term (such as a user ID, historical information, etc.) can be maintained easily with no server interaction.
* For small- and medium-sized session data, the entire session data (instead of just the session ID) can be kept in the cookie.

**36. What are some disadvantages of storing session state in cookies?**

* Cookies are controlled by programming a low-level API, which is more difficult to implement than some other approaches.
* All data for a session are kept on the client. Corruption, expiration or purging of cookie files can all result in incomplete, inconsistent, or missing information.
* Cookies may not be available for many reasons: the user may have disabled them, the browser version may not support them, and the browser may be behind a firewall that filters cookies, and so on. Servlets and JSP pages that rely exclusively on cookies for client-side session state will not operate properly for all clients. Using cookies, and then switching to an alternate client-side session state strategy in cases where cookies aren't available, complicates development and maintenance.
* Browser instances share cookies, so users cannot have multiple simultaneous sessions.
* Cookie-based solutions work only for HTTP clients. This is because cookies are a feature of the HTTP protocol. Notice that the while package javax.servlet.http supports session management (via class HttpSession), package javax.servlet has no such support.

**37. What is URL rewriting?**

URL rewriting is a method of session tracking in which some extra data is appended at the end of each URL. This extra data identifies the session. The server can associate this session identifier with the data it has stored about that session.

Every URL on the page must be encoded using method HttpServletResponse.encodeURL(). Each time a URL is output, the servlet passes the URL to encodeURL(), which encodes session ID in the URL if the browser isn't accepting cookies, or if the session tracking is turned off.  
E.g., http://abc/path/index.jsp;jsessionid=123465hfhs

**Advantages**

* URL rewriting works just about everywhere, especially when cookies are turned off.
* Multiple simultaneous sessions are possible for a single user. Session information is local to each browser instance, since it's stored in URLs in each page being displayed. This scheme isn't foolproof, though, since users can start a new browser instance using a URL for an active session, and confuse the server by interacting with the same session through two instances.
* Entirely static pages cannot be used with URL rewriting, since every link must be dynamically written with the session state. It is possible to combine static and dynamic content, using (for example) templating or server-side includes. This limitation is also a barrier to integrating legacy web pages with newer, servlet-based pages.

**Disadvantages:**

* Every URL on a page which needs the session information must be rewritten each time a page is served. Not only is this expensive computationally, but it can greatly increase communication overhead.
* URL rewriting limits the client's interaction with the server to HTTP GETs, which can result in awkward restrictions on the page.
* URL rewriting does not work well with JSP technology.
* If a client workstation crashes, all of the URLs (and therefore all of the data for that session) are lost.

**38. How can an existing session be invalidated?**

An existing session can be invalidated in the following two ways:

* Setting timeout in the deployment descriptor: This can be done by specifying timeout between the <session-timeout>tags as follows:

<session-config>  
       <**session-timeout>10</session-timeout>**  
</session-config>

This will set the time for session timeout to be ten minutes.

* Setting timeout programmatically: This will set the timeout for a specific session. The syntax for setting the timeout programmatically is as follows:

public void setMaxInactiveInterval(int interval)

The setMaxInactiveInterval() method sets the maximum time in seconds before a session becomes invalid.   
Note: Setting the inactive period as negative (-1), makes the container stop tracking session, i.e., session never expires.

**39. How can the session in Servlet can be destroyed?**

An existing session can be destroyed in the following two ways:

* Programmatically: Using session.invalidate() method, which makes the container abandon the session on which the method is called.
* When the server itself is shutdown.

**40. A client sends requests to two different web components. Both of the components access the session. Will they end up using the same session object or different session?**

Creates only one session i.e., they end up with using same session.

Sessions is specific to the client but not the web components. And there is a 1-1 mapping between client and a session.

**41. What is servlet lazy loading?**

* A container does not initialize the servlets as soon as it starts up, it initializes a servlet when it receives a request for that servlet first time. This is called lazy loading.
* The servlet specification defines the <load-on-startup> element, which can be specified in the deployment descriptor to make the servlet container load and initialize the servlet as soon as it starts up.
* The process of loading a servlet before any request comes in is called preloading or preinitializing a servlet.

**42. What is Servlet Chaining?**

Servlet Chaining is a method where the output of one servlet is piped into a second servlet. The output of the second servlet could be piped into a third servlet, and so on. The last servlet in the chain returns the output to the Web browser.

**43. How are filters?**

Filters are Java components that are used to intercept an incoming request to a Web resource and a response sent back from the resource. It is used to abstract any useful information contained in the request or response. Some of the important functions performed by filters are as follows:

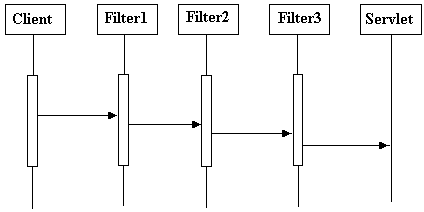
* Security checks
* Modifying the request or response
* Data compression
* Logging and auditing
* Response compression

Filters are configured in the deployment descriptor of a Web application. Hence, a user is not required to recompile anything to change the input or output of the Web application.

**44. What are the functions of an intercepting filter?**

The functions of an intercepting filter are as follows:

* It intercepts the request from a client before it reaches the servlet and modifies the request if required.
* It intercepts the response from the servlet back to the client and modifies the request if required.
* There can be many filters forming a chain, in which case the output of one filter becomes an input to the next filter. Hence, various modifications can be performed on a single request and response.



**45. What are the functions of the Servlet container?**

The functions of the Servlet container are as follows:

* **Lifecycle management**: It manages the life and death of a servlet, such as class loading, instantiation, initialization, service, and making servlet instances eligible for garbage collection.
* **Communication support**: It handles the communication between the servlet and the Web server.
* **Multithreading support**: It automatically creates a new thread for every servlet request received. When the Servlet service() method completes, the thread dies.
* **Declarative security**: It manages the security inside the XML deployment descriptor file.
* **JSP support**: The container is responsible for converting JSPs to servlets and for maintaining them.

# Servlet interview questions and answers

**Q 1. What is servlet?**  
Servlet is a server side programming language which is used for generating dynamic web pages. It generates web-page as a response of the request received from client(browser).

**Q 2. Static webpage vs Dynamic webpage?**  
The webpages which are same for all the users are static webpages and the webpages that are dynamically generated based on the user’s request (that may be different for each user depending on the request) are known as dynamic webpages. Servlet is mainly used for dynamic webpages.

**Q 3. Life cycle of a servlet?**  
Following the stages of servlet life cycle:  
1) Loading of Servlet class: The servlet container finds the servlet class mentioned in web.xml file and loads it.  
2) Servlet instantiation: The object of servlet class gets created in this phase.  
3) Initialization: Servlet initialization by calling init() method.  
4) Servicing the request: In this phase the servlet service the client request by calling the service() method.  
5) Destroy: Last phase of servlet life cycle. The destroy() method free up the servlet instance so that it can be garbage collected.

**Q 4. Why we use Servlets?**  
1) To process the input data submitted by user from the screen.  
2) Generate and return the dynamic response to the user based on the request.  
3) Ideal programming language for interacting with database based on the user’s request.  
4) A servlet can handle multiple request simultaneously which makes them a perfect choice for the high performing applications.

**Q 5. Why Servlet is better than CGI?**  
1) Servlet responses faster than CGI because it uses the multithreading concept to service each request. CGI performance is not that good as it creates a new object for each request while servlet allots a new thread for each request.  
2) Learning and implementing servlet is quite easier compared to CGI.  
3) Memory consumption is low in servlet compared to CGI.

**Q 6. When Servlet is loaded?**  
1) When servlet container receives the first request from client(browser).  
2) Admin of the application manually loads the servlet.  
3) When the webserver(in which the servlet is deployed) gets started.

**Q 7. What is Servlet interface and what’s the use of it?**  
Servlet interface is an API for servlets. Every Servlet should either implement the servlet interface or extends the class which already implements the interface. Javax.servlet.GenericServlet and javax.servlet.http.HttpServlet are the Servlet classes that implements Servlet interface, hence every servlet should either implement Servlet interface directly or by extending any of these classes.

**Q 8. What is ServletConfig?**  
ServletConfig interface belongs to the package javax.servlet. ServletConfig. It is used for passing the configuration parameters to the servlet. Servlet container instantiate it implicitly.

**Q 9. What is ServletContext?**  
Each web application has a common ServletContext. All the servlets in the web application can access the ServletContext. It has the web-application information & resources which are common and accessible to all the servlets present in the web application.

**Q 10. ServletConfig vs ServletContext?**  
Following are the two main differences between ServletConfig and ServletContext:  
1) ServletConfig is used for passing the configuration information to the Servlet while ServletContext provides set of methods which a Servlet class can use to communicate with Servlet container.  
2) Each Servlet has a separate ServletConfig object while ServletContext is common for all the servlets in the web application.  
3) Parameters of ServletConfig are defined under <init-param> tags in web.xml file. Parameters of ServletContext are defined under <context-param> tags in web.xml.

**Q 11. Difference between GenericServlet and HttpServlet?**  
1) GenericServlet is an abstract class which implements Servlet interface while HttpServlet abstract class extends the GenericServlet class. In short: GenericServlet class is a parent class for HttpServlet.  
2) GenericServlet does not support any protocol. HttpServlet support HTTP and HTTPS protocol.  
3) GenericServlet cannot handle cookies and session while HttpServlet can handle them.

**Q 12. Difference between forward() and sendRedirect()?**  
1) In forward() the same request is forwarded to the resource. In sendRedirect() new request is send to the redirected resource.  
2) forward() is taken care by the Servlet container while sendRedirect() is handled by the browser.  
3) In forward() the URL (uniform resource locator) remains same on web browser. In sendRedirect() the URL changes in the web browser address bar.  
4) forward() is faster compared to sendRedirect().

**Q 13. What is deployment descriptor?**  
Web.xml file of a web application is known as deployment descriptor. It is usually placed inside WEB-INF folder of application. It has the information like Servlet name, Servlet mapping etc. This file tells the Servlet container which Servlet class needs to be called for the given URL pattern.

**Q 14. doGet() Vs doPost() methods?**  
1) In doGet(), the parameters are visible in the address bar, they get appended to the URL. In doPost() parameters are not visible in the address bar.  
2) You can maximum transfer 1024 characters through GET request. doPost() doesn’t have any limitations.  
3) doGet() is not good for sensitive data as the parameters do not get encrypted. In doPost() the parameters are encrypted hence it is more secure compared to doGet().  
4) Method doGet() allow you to bookmark the resource. doPost() doesn’t allow bookmarks.  
5) doGet() is faster compared to the doPost() method.

**Q 15. What is the use of <load-on-startup>?**  
<load-on-startup> is used for specifying the Servlet files which needs to be loaded during server startup. The servlet files specified in this element are loaded as soon as the server starts, it does not wait for the first request for loading them up. This is how it is specified in web.xml file.

<Servlet>

   <servlet-name>MyServlet Name Here</servlet-name>

   <servlet-class>Servlet ClassHere-FullyQualified</servlet-class>

   <load-on-startup>1</load-on-startup>

</servlet>

If more than one files are specified then the files will be loaded in the same order in which they have been specified in it.

**Q 20. What are the different types of session tracking mechanism supported by Servlets?**  
1) URL rewriting  
2) Hidden Form Fields  
3) Cookies  
4) Secure Socket Layer(SSL) Sessions

**Q 21. How URL rewriting maintains session?**  
In URL rewriting method, the session tracking data has been appended at the end of the URL to track the session.

**Q 22. Explain Servlet chaining?**  
Servlet chaining is a concept where the request is processed in a chain of servlets. First Servlet processes the request partially and passes to the second one, then second servlet process it and passes to third one and so on. The last servlet returns the response to the client (browser).

**Q 23. How to invalidate a session in servlet?**  
By calling session.invalidate() method.

**Q 24. What are the main functions of Servlet container?**  
1) Servlet life cycle management  
2) maintains the interaction between servlet and webserver.  
3) Providing multithreading support for processing more than one request simultaneously.  
4) Managing of deployment descriptor web.xml file.

**Q 25. What is <session-timeout>?**  
The element <session-timeout> is used for specifying the timeout of a Session. This is how it is defined in the web.xml file.

<Session-config>

       <session-timeout>35</session-timeout>

</session-config>

It would set the session timeout to 25 minutes.

**Q 26. What is Servlet lazy loading and how it can be avoided?**  
The Servlet container does not initialize the Servlet on server startup by default. It only initializes a servlet when the it receives the request from the client. This is called lazy loading of Servlet.  
By specifying **<load-on-startup>**element for a Servlet we can avoid lazy loading. The servlet files specified in <load-on-startup> are loaded as soon as the web server starts.

**Q 27. Why do we need constructor in servlet even though we have an init() method?**  
**init()** method is used for initializing the servlet however constructor is required in order to instantiate the Servlet class. Servlet container instantiate the Servlet class.

**Q 28. When the Servlet is unloaded?**  
1) Admin manually unloads the servlet.  
2) Web server shut down.

**Q 29. How Servlet maintains session using cookies?**  
Cookie is a small piece of information, which is sent by a servlet to the Web browser. Cookies gets stored in the browser and returned back to the server when needed. A cookie has a name, a single value, and few other attributes.

**Q 30. Why using cookies for session tracking is a bad practice?**  
There are several disadvantages of using cookies for session tracking. Few of them are:  
1) Since cookies are stored on client-side (in the client’s browser), It will not be available if client browser clears or disables the cookies.  
2) Implementing cookies for session tracking is much more difficult compared to other session management mechanism.  
3) Cookies only work for HTTP protocol.

**Q 31. How do I get the server info in Servlets?**  
Use this:

getServletContext().getServerInfo()

**Q 32. How to get the client’s IP address in Servlets?**  
Using this:

request.getRemoteAddr()

**Q 33. Why we use filters in Servlet?**  
We use filters for:  
1) Security checks  
2) Modifying the request or response  
3) Data compression  
4) Logging and auditing  
5) Response compression

**Q 34. What all protocols are supported by HttpServlet?**  
HTTP and HTTPS protocols.

**Q 35. What all protocols are supported by GenericServlet?**  
GenericServlet abstract class is not specific to any protocol.

**Q 36. What are the new features added to Servlet 3?**  
1) Servlet Annotations  
2) Web Fragments  
3) Web components addition dynamically  
4) Asynchronous Processing

**Q 37. Do we override service() method?**  
No, we do not override the **service()** method. We generally override the doPost(), doGet() method based on the requirement.

# JAVA SERVLETS INTERVIEW QUESTIONS & ANSWERS

# [wisdomjobs](https://www.wisdomjobs.com/)

1. **Question 1. Can We Use The Constructor, Instead Of Init(), To Initialize Servlet?**

**Answer:**

Yes. But you will not get the Servlet specific things from constructor. The original reason for init() was that ancient versions of Java couldn’t dynamically invoke constructors with arguments, so there was no way to give the constructor a ServletConfig. That no longer applies, but servlet containers still will only call your no-arg constructor. So you won’t have access to a ServletConfig or ServletContext.

1. **Question 2. What Is Servlet Context?**

**Answer:**

The Servlet context is an object that contains a information about the Web application and container. Using the context, a Servlet can log events, obtain URL references to resources, and set and store attributes that other Servlet in the context can use.

1. **Question 3. What Is A Servlet Filter?**

**Answer:**

Servlet filters are pluggable Web components that allow us to implement pre-processing and post-processing logic in our Web applications.

1. **Question 4. What Is A War File?**

**Answer:**

WAR stands for Web Archive. It is a compressed version of your web application. You can use this WAR file to deploy your web application.

1. **Question 5. How Would You Create Deadlock On Your Servlet?**

**Answer:**

Calling a doPost() method inside doGet() and doGet() method inside doPOST() wouleate a deadlock for a servlet.

1. **Question 6. Why HttpServlet Is Declared Abstract?**

**Answer:**

1. The default implementations of the main service methods cannot do anything and need to be overridden. This calls of the HttpServlet class to be declared as abstract.  
2. With its use the developers do not need to implement all the service methods.

1. **Question 7. Why Is A Constructor Needed In A Servlet Even If We Use The Init Method?**

**Answer:**

1. Although the init method of the servlet initializes it, a constructor instantiates it.  
2. A developer might never explicitly call the servlet's constructor but a container uses it to create an instance of the servlet.

1. **Question 8. What Is GenericServlet Class?**

**Answer:**

1. GenericServlet is an abstract class which implements the Servlet interface and the ServletConfig interface.  
2. Other than the methods included in above two interfaces, it also provides simple versions of the lifecycle methods init and destroy, and implements the log method declared in the ServletContext interface.  
3. Since this class is not specific to any protocol, it is known as generic servlet.

1. **Question 9. How Can The Session In Servlet Be Destroyed?**

**Answer:**

There are two ways to destroy a session:  
1. **Programmatically :** By using session.invalidate() method. It makes the container abandon the session on which the method is called.  
2. When the server shuts down.

1. **Question 10. What Is Lazy Loading?**

**Answer:**

The servlets are not initialized by the container from the start. It happens when the servlet is requested for the first time. This is called lazy loading.

1. **Question 11. What Are The Mechanisms Used By A Servlet Container For Maintaining Session Information?**

**Answer:**

For maintaining session information Servlet Container uses:  
. Cookies  
. URL rewriting  
. HTTPS protocol information

1. **Question 12. What Is The Procedure For Initializing A Servlet?**

**Answer:**

- To initialize a servlet init() is used.   
- init() initializes a java program.  
- A constructor can also be used to initialize a servlet.

1. **Question 13. What Is The Web Container?**

**Answer:**

A Servlet and JSP containers are collectively referred to as Web containers.

1. **Question 14. What Are The Uses Of ServletRequest?**

**Answer:**

The ServletRequest gives information such as the names of the parameters passed by the client, the protocol (scheme) being used by the client, and the names of the remote host that made the request and the server that received it. The input stream, ServletInputStream.

1. **Question 15. What Are The Uses Of ServletResponse Interface?**

**Answer:**

ServletResponse allows the servlet to set the content length and MIME type of that response. It provides an output stream, ServletOutputStream and a Writer through which the servlet can send data.

1. **Question 16. How Http Servlet Handles Client Requests?**

**Answer:**

An HTTP Servlet handles client requests through its service method. The service method supports standard HTTP client requests by dispatching each request to a method designed to handle that request.

1. **Question 17. What Is Pre Initialization Of A Servlet?**

**Answer:**

A container doesn't initialize the servlets when it starts up. It initializes a servlet when it receives a request for that servlet first time. This is called lazy loading. The servlet specification defines the <load-on-startup> element, which can be specified in the deployment descriptor to make the servlet container load and initialize the servlet as soon as it starts up. The process of loading a servlet before any request comes in is called preloading or pre initializing a servlet.

1. **Question 18. How Do Servlets Handle Multiple Simultaneous Requests?**

**Answer:**

When a request comes in, the web server will start a new thread and the request is assigned to a thread, which calls a service method of the servlet.

1. **Question 19. What Is Servlet Chaining?**

**Answer:**

Servlet chaining is a technique in which two or more servlets can cooperate in servicing a single request. In servlet chaining, one servlet’s output is the input of next servlet. This process continues until the last servlet is reached. Its output is then sent back to the client. We are achieving Servlet Chaining with the help of RequestDispatcher.

1. **Question 20. How Will You Communicate From An Applet To Servlet?**

**Answer:**

There are three ways to communicate from an applet to servlet and they are: HTTP Communication (Text-based and object-based), Socket Communication and RMI Communication.

1. **Question 21. How Do You Communicate Between The Servlets?**

**Answer:**

 We can communicate between servlets by using RequestDispatcher interface and servlet chaining.

1. **Question 22. What Is The Difference Between Context Init Parameter And Servlet Init Parameter?**

**Answer:**

Servlet init parameters are for a single servlet only. Nobody out side that servlet can access that. It is declared inside the <servlet> tag inside Deployment Descriptor, whereas context init parameter is for the entire web application. Any servlet or JSP in that web application can access context init parameter. Context parameters are declared in a tag <context-param> directly inside the <web-app> tag. The methods for accessing context init parameter is getServletContext ().getInitParamter (“name”) whereas method for accessing servlet init parameter is getServletConfig ().getInitParamter (“name”);

1. **Question 23. What Are The Different Ways For Getting A Servlet Context?**

**Answer:**

We will get ServletContext by calling getServletConfig ().getServletContext (). This is because a ServletConfig always hold a reference to ServletContext. By calling this.getServletContext () also we will get a ServletContext object.

1. **Question 24. What Is Http Tunneling?**

**Answer:**

HTTP tunneling is used to encapsulate other protocols within the HTTP or HTTPS protocols. Normally the intranet is blocked by a firewall and the network is exposed to the outer world only through a specific Web server port that listens for only HTTP requests. To use any other protocol, that by passes the firewall, the protocol is embedded in HTTP and send as HttpRequest.

1. **Question 25. What Are The Differences Between A Session And A Cookie?**

**Answer:**

Session is stored in server but cookie stored in client. Session should work regardless of the settings on the client browser. There is no limit on the amount of data that can be stored on session. But it is limited in cookie. Session can store objects and cookies can store only strings. Cookies are faster than session.

1. **Question 26. Why Should We Go For Inter Servlet Communication?**

**Answer:**

The three major reasons to use inter servlet communication are:   
a) **Direct servlet manipulation** - allows to gain access to the other currently loaded servlets and perform certain tasks (through the ServletContext object)  
b) **Servlet reuse** - allows the servlet to reuse the public methods of another servlet.   
c) **Servlet collaboration** - requires to communicate with each other by sharing specific information (through method invocation).

1. **Question 27. What Is Client Side Refresh?**

**Answer:**

The standard HTTP protocols ways of refreshing the page, which is normally supported by all browsers.  
<META HTTP-EQUIV="Refresh" CONTENT="5; URL=/servlet/MyServlet/">   
this will refresh the page in the browser automatically and loads the new data every 5 seconds.

1. **Question 28. What Is Server Side Push?**

**Answer:**

Server Side push is useful when data needs to change regularly on the clients application or browser, without intervention from client. The mechanism used is, when client first connects to Server, then Server keeps the TCP/IP connection open.

1. **Question 29. What's The Servlet Interface?**

**Answer:**

The central abstraction in the Servlet API is the Servlet interface. All servlets implement this interface, either directly or, more commonly, by extending a class that implements it such as HttpServlet.

1. **Question 30. What Is The Difference Between ServletContext And ServletConfig?**

**Answer:**

The ServletConfig gives the information about the servlet initialization parameters. The servlet engine implements the ServletConfig interface in order to pass configuration information to a servlet. The server passes an object that implements the ServletConfig interface to the servlet's init() method.  
The ServletContext gives information about the container. The ServletContext interface provides information to servlets regarding the environment in which they are running. It also provides standard way for servlets to write events to a log file.