**Q. Can we call destroy() method inside the init() method is yes what will happen?**

**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.

**Q: 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”)

**Q: 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:

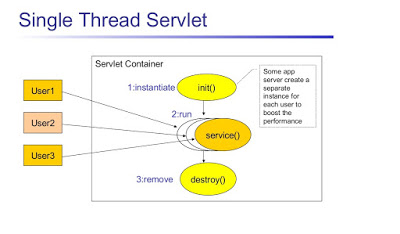
1. Single Thread Model

2. Multithread Model

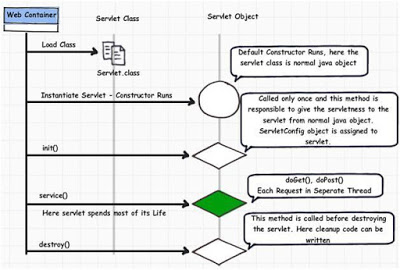
Depending upon our scenario, if we have implemented single thread means only one instance is going handle one request at a time no two thread will concurrently execute service method of the servlet.

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%2BThread%2BServlet.jpg)

**Q: 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)

**Q. What is load-on-startup servlet element in web.xml with Example?**

load-on-startup is an element that appears inside <servlet> tag in web.xml.4 years back load-on-startup was a very popular servlet interview question because not many Java J2EE developer was familiar with this element and how load-on-startup works inside a servlet container like tomcat or WebSphere. In this J2EE Tutorial, we will see what is a load on startup, how to use load-on-startup element, and what are different values we can configure for loadOnStartup inside web.xml.

What is load-on-startup

As stated earlier load-on-startup is a tag element that appears inside <servlet> tag in web.xml. load-on-startup tells the web container about the loading of a particular servlet.   
  
If you don't specify load-on-startup then the container will load a particular servlet when it feels necessary most likely when the first request for that servlet will come, this may lead to longer response time for that query if Servlet is making database connections or performing LDAP authentication which contributes network latency or any other time-consuming job, to avoid this, web container provides you a mean to specify certain servlet to be loaded during deployment time of application by using the load-on-startup parameter.

If you specify load-on-startup parameter inside a servlet than based upon its value Container will load it.you can specify any value to this element but in case of load-on-startup>0 , servlet with less number will be loaded first.   
  
For example, in below web.xml AuthenticationServlet will be loaded before AuthorizationServlet because the load-on-startup value for AuthenticationServlet is less (2) while for AuthorizationServlet is 4.  
  
Btw, if you are new into Servlet and JSP then I also suggest you go through a comprehensive course to learn Servlet and JSP in depth. If you need a recommendations then I suggest you check out JSP, Servlet, and JDBC for Beginners: Build a Database App course on Udemy by Chad Darby. One of the hands-on course to learn Servlet and JSP.

load-on-startup Example in web.xml

here is an example of how to use load on startup tag inside servlet element in web.xml:

<servlet>

<servlet-name>AuthenticationServlet</servlet-name>

<display-name>AuthenticationServlet</display-name>

<servlet-class>com.trading.AuthenticationServlet</servlet-class>

<load-on-startup>2</load-on-startup>

</servlet>

<servlet>

<servlet-name>AuthorizationServlet</servlet-name>

<display-name>AuthorizationServlet</display-name>

<servlet-class>com.trading.AuthorizationServlet</servlet-class>

<load-on-startup>4</load-on-startup>

</servlet>

Important points on the load-on-startup element

1. If <load-on-startup> value is the same for two servlets than they will be loaded in an order on which they are declared inside web.xml file.

2. if <load-on-startup> is 0 or negative integer than Servlet will be loaded when Container feels to load them.

3. <load-on-startup> guarantees loading, initialization and call to init() method of servlet by web container.

4. If there is no <load-on-startup> element for any servlet than they will be loaded when web container decides to load them.

**Q. When to use <load-on-startup> in web.xml**

<load-on-startup> is suitable for those servlets which perform time-consuming jobs e.g. Creating Database Connection pool, downloading files or data from the network, or prepare environment ready for servicing client in terms of initializing cache, clearing pipelines and loading important data in memory.   
  
If any of your servlets perform these jobs then declare them using <load-on-startup> element and specify order as per your business logic or what suites your application.   
  
Remember to lower the value of <load-on-startup>, the servlet will be loaded first. You can also check your web container documentation on how exactly the load on start-up is supported.

Q. How to avoid deadlock in Java?

How to avoid deadlock in Java? Is one of the popular Java interview question and flavor of the season for multi-threading, asked mostly at a senior level with lots of follow up questions. Even though the problem looks very basic but most of the Java developers get stuck once you start going deep. Interview questions start with, "What is a deadlock?" The answer is simple when two or more threads are waiting for each other to release the resource they need (lock) and get stuck for infinite time, the situation is called deadlock. It will only happen in the case of multitasking or multi-threading.  
  
Btw, if you are serious about mastering Java multi-threading and concurrency then I also suggest you take a look at the Java Multithreading, Concurrency, and Performance Optimization course by Michael Pogrebinsy on Udemy. It's an advanced course to become an expert in Multithreading, concurrency, and Parallel programming in Java with a strong emphasis on high performance

**Q. How do you detect deadlock in Java?**

Though this could have many answers, my version is; first I would look at the code if I see a nested synchronized block or calling one synchronized method from other, or trying to get a lock on a different object then there is a good chance of deadlock if a developer is not very careful.  
  
Another way is to find it when you actually get dead-locked while running the application, try to take a thread dump, in Linux you can do this by the command "kill -3", this will print status of all threads in an application log file, and you can see which thread is locked on which object.  
  
You can analyze that thread dump with using tools like fastthread.io which allows you to upload your thread dump and analyze it.  
  
Another way is to use the jConsole/VisualVM, it will show you exactly which threads are getting locked and on which object.  
  
If you are interested to learn about troubleshooting tools and processes to analyze your thread dump, I suggest you take a look at Analyzing Java Thread Dumps course on Pluralsight by Uriah Levy. An advanced practical course to learn more about Java thread dumps, and familiarize you with other popular advanced troubleshooting tools.

I also suggest you join the Java Concurrency in Practice Bundle by Heinz Kabutz, one of the most advanced course material to master concurrency and multi-threading in Java. It's based on the classic Java Concurrency in Practice book by Brian Goetz, which is a recommended reading for every Java developer.

**Q.Write a Java program that will result in deadlock?**

Once you answer the earlier question, they may ask you to write code which will result in a deadlock in Java?  
  
here is one of my version

/\*\*

\* Java program to create a deadlock by imposing circular wait.

\*

\* @author WINDOWS 8

\*

\*/

public class DeadLockDemo {

/\*

\* This method request two locks, first String and then Integer

\*/

public void method1() {

synchronized (String.class) {

System.out.println("Aquired lock on String.class object");

synchronized (Integer.class) {

System.out.println("Aquired lock on Integer.class object");

}

}

}

/\*

\* This method also requests same two lock but in exactly

\* Opposite order i.e. first Integer and then String.

\* This creates potential deadlock, if one thread holds String lock

\* and other holds Integer lock and they wait for each other, forever.

\*/

public void method2() {

synchronized (Integer.class) {

System.out.println("Aquired lock on Integer.class object");

synchronized (String.class) {

System.out.println("Aquired lock on String.class object");

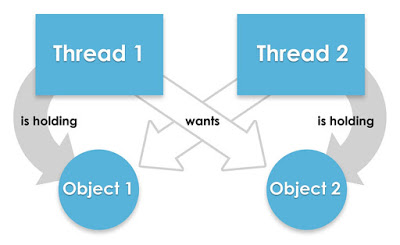
}

}

}

}

If method1() and method2() both will be called by two or many threads, there is a good chance of deadlock because if thread 1 acquires lock on Sting object while executing method1() and thread 2 acquires lock on Integer object while running method2() both will be waiting for each other to release the lock on Integer and String to proceed further which will never happen.  
  
This diagram precisely demonstrates our program, where one thread holds a lock on one object and waiting for other object locks which are owned by other thread.

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fmultithreading-and-parallel-computing-in-java%2F)

You can see that Thread 1 wants the lock on object 2 which is held by Thread 2, and Thread 2 wants a lock on Object 1 which is held by Thread 1. Since no thread is willing to give up, there is a deadlock, and the Java program is stuck.  
  
The idea is that you should know the right way to use common concurrent patterns, and if you are not familiar with them then Applying Concurrency and Multi-threading to Common Java Patterns by Jose Paumard is a good starting point to learn.

**Q. How to avoid deadlock in Java?**

Now the interviewer comes to the final part, one of the most important in my view; How do you fix a deadlock in code? Or How to avoid deadlock in Java?  
If you have looked above code carefully, then you may have figured out that real reason for the deadlock is not multiple threads but the way they are requesting a lock, if you provide ordered access then the problem will be resolved.  
Here is my fixed version, which avoids deadlock by avoiding circular wait with no preemption, one of the four conditions which need for deadlock.  
public class DeadLockFixed {

/\*\*

\* Both method are now requesting lock in same order,

\* first Integer and then String.

\* You could have also done reverse e.g. first String and then Integer,

\* both will solve the problem, as long as both method are requesting lock

\* in consistent order.

\*/

public void method1() {

synchronized (Integer.class) {

System.out.println("Aquired lock on Integer.class object");

synchronized (String.class) {

System.out.println("Aquired lock on String.class object");

}

}

}

public void method2() {

synchronized (Integer.class) {

System.out.println("Aquired lock on Integer.class object");

synchronized (String.class) {

System.out.println("Aquired lock on String.class object");

}

}

}

}

Now there would not be any deadlock because both methods are accessing lock on Integer and String class literal in the same order. So, if thread A acquires a lock on Integer object, thread B will not proceed until thread A releases Integer lock, same way thread A will not be blocked even if thread B holds String lock because now thread B will not expect thread A to release Integer lock to proceed further.  
  
That's all about how to avoid deadlock in Java. If you are serious about improving your multi-threading and concurrency skills there are here are some of my recommended online courses and books for Java developers.

Constructor vs Init method in Servlet - JEE Interview Question

Can we have a Constructor in Java Servlet? or Why do we need a constructor in Servlet if there is already an init() method for initializing Servlet, or what is the difference between init() method and constructor in Servlet are a couple of questions I have seen in various Java web developer interviews. All of these questions are related to the role of the constructor and init() method in the Servlet implementation class. Though I had shared few thoughts on this, when I wrote the top 10 Servlet questions for Java programmers,  I thought to cover it in more detail here.  
  
In this article, I will try to answer each of these questions in detail. The key thing to remember is that Servlet is special in the sense that their life cycle is managed by web containers like Tomcat and Jetty. They are responsible for creating instances of Servlet and destroying them when they don't have enough resources or need to support so many instances of Servlets.  
  
Let me first answer the question, Can we create a constructor in Servlets? and then I will answer why you should be using the init() method for Servlet initialization by describing the difference between constructor and init() method.  
  
Btw, if you are new into Servlet and JSP then I also suggest you go through a comprehensive course to learn Servlet and JSP in depth. If you need a recommendations then I suggest you check out JSP, Servlet, and JDBC for Beginners: Build a Database App course on Udemy by Chad Darby. One of the hands-on course to learn Servlet and JSP.

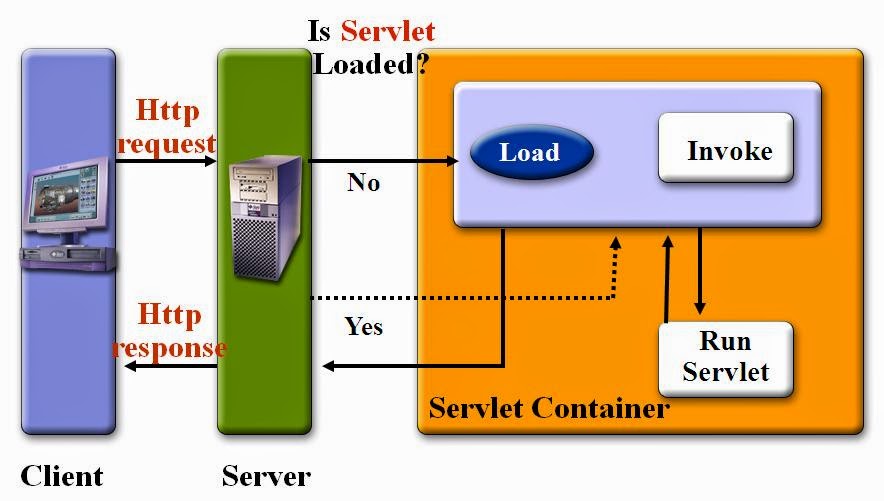
**Q. Can we define Constructor in Servlet?**

The short answer to this question, Yes, Servlet implementation classes can have constructor but they should be using init() method to initialize Servlet because of two reasons, first you cannot declare constructors on an interface in Java.  
  
This means you cannot enforce this requirement to any class which implements the Servlet interface and second, Servlet requires ServletConfig object for initialization which is created by container as it also has a reference of ServletContext object, which is also created by the container.

Servlet is an interface defined in javax.servlet package and HttpServlet is a class and like any other class in Java they can have constructors, but you cannot declare constructor inside interface in Java. If you don't provide an explicit constructor than compiler will add a default no-argument constructor in any Servlet implementation class.  
  
Another reason that you should not initialize Servlet using constructor because Servlets are not directly instantiated by Java code, instead containers create there instance and keep them in the pool.  
  
Since containers from web servers like Tomcat and Jetty uses Java Reflection for creating an instance of Servlet, the presence of no-argument constructor is a must. So, by any chance, if you provide a parametric constructor and forget to write a no-argument constructor, web container will not be able to create an instance of your Servlet, since there is no default constructor.  
  
Remember Java compiler doesn't add default no-argument constructor if there is a parametric constructor present in class. That's why it's not advised to provide a constructor in Servlet class. Now let's see some difference between Constructor and init method in Java Servlet

**Q. Difference between Constructor and init method in Servlet**

In a real-world application, you better use init() method for initialization, because init() method receives a ServletConfig parameter, which may contain any initialization parameters for that Servlet from web.xml file. Since web.xml provides useful information to web container e.g. the name of Servlet to instantiate, ServletConfig instance is used to supply initialization parameter to Servlets.  
  
You can configure your Servlet based upon settings provided in ServletConfig object e.g. you can also provide environment-specific settings e.g. the path of temp directory, database connection parameters (by the way for that you should better leverage JNDI connection pool) and any other configuration parameters.  
  
You can simply deploy your web application with different settings in web.xml file on each environment. Remember, init() method is not chained like a constructor, where superclass constructor is called before subclass constructor executes, also known as constructor chaining.

[](https://1.bp.blogspot.com/-X1iEBeSAB_8/VNocb5Ej9DI/AAAAAAAACew/JDyljCkXoXQ/s1600/Servlet%2BInterview%2BQuestions.jpg)

That's all on this post about the difference between constructor and init method of Servlet. We have seen that the container uses web.xml to get Servlet's name for initialization and uses Java Reflection API, primarily class.newInstance() method to create an instance of Servlet, which means Servlet class must need a default no-argument constructor.  
  
We have also seen that why Servlet cannot have user-defined constructor, mainly because Servlet as the interface cannot guarantee it and web container creates an instance of Servlet and has access to Context and Config object which is not accessible to the developer.  
  
Always prefer to init() method for initializing Servlet than the constructor, because ServletConfig object is supplied to init() method with configuration parameters.

**Q. What is JSESSIONID in Java? When does JSESSIONID gets created ?**

One of my favourite Servlet JSP Interview question for 2 to 4 years experience programmers on web development. JSESSION id is a cookie which is used to manage session in Java web application. JSESSIONID is created by Web Container whenever a new session is created. See What is JSESSIONID in Servlet JSP for more details.

**Q. What is difference between include action and include directive in JSP?**

Another very popular JSP Interview questions, mostly asked to 2 to 3 years experienced J2EE programmer. There are couple of differences, most important of them is that include action is request  time inclusion while include directive is translation time inclusion of another resource e.g. JSP or html pages. This questions is also asked as difference between file include and page include. See Include action vs Include directive for more differences.

**Q: How do you define application wide error page in JSP?**

Almost in every Servlet JSP interview you will see question from error handling. You can define two kinds of error pages in Java web application, one is using tag <error-page> in web.xml and other is by using error page JSP which uses isErrorpage to declare that this jsp page can be used as error page. Other JSP uses that page by using attribute errorpage="error.jsp". Whenever you get an unhandled exception in JSP, request will be routed to error page. See How to use error page in JSP for detailed answer of this question.

**Q: Difference between sendredirect and forward in Servlet ?**

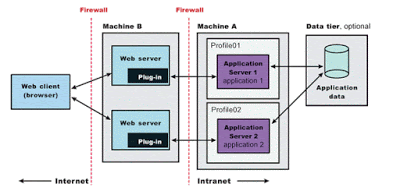
One of the classical Interview Question from Servlet and JSP. This question is as old as Vector vs ArrayList in core Java. Anyway see difference between sendredirect and forward to answer this Servlet Interview question.

**Q: How do remove variable using <c:set> tag from JSTL ?**

This is one of the tricky Servlet JSP question. Many people assumes that <c:set> can only add or set variables in a particular scope but you can also remove any variable from any scope using JSTL <c:set> tag. See How to use <c:set> JSTL tag in JSP for exact way to remove any variable from any scope in JSP page.

**Q 6: What is difference between Web Server and Application Server ?**

This is rather simple Servlet JSP question to answer. If you have used EJB then you should know that , Web Server doesn't contain EJB container and EJB can not be deployed on that. Application Server is used to deploy and run EJB in J2EE environment. See 5 difference between Application and Web Server to see more differences.

[](https://1.bp.blogspot.com/-SDLWxW9RBEY/XoVEnyOBiNI/AAAAAAAAdtw/ZhC74P8o0UANZhQC04r2dUOsyU6_DuMcwCLcBGAsYHQ/s1600/Web%2BServer%2Bvs%2BApplication%2BServer%2Bin%2BJava%2BJEE.gif)

**Q 7: What is difference between URL Encoding and URL rewriting ?**

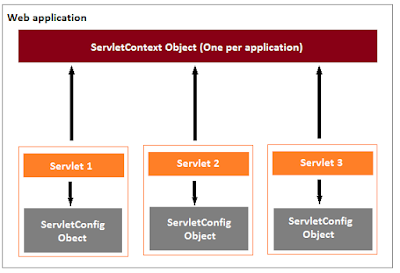
URL Encoding and URL rewriting is general web concept irrespective of Java programming language. URL Encoding refers to encoding URL e.g. replacing space with %20, you can see this when browser sends request to Server. On the other hand URL Rewriting is way to manage Session in web application. See URL Encoding vs URL Rewriting for more differences.

**Q : How do you get ServletContext reference inside Servlet ?**

ServletContext is very important object in Java Web application. ServletContext object is one per web application and serves as application scope and use to store common config and things which doesn't change on per session. Its easy to get reference of ServletContext in jsp using application implicit variable but in Servlet is not available in HttpServletRequest until version 3.0. You need HttpSession object to retrieve ServletContext in any Servlet. A good follow-up questions is How do you get ServletContext in Struts Action class or Spring Controller classes. See How to retrieve ServletContext in Servlet, Spring and Struts for more details.

**Q: What is difference between ServletContext and ServletConfig in Java ?**

One of those classical Servlet Interview Questions which you can't afford to miss. Again ServletContext is used to provide application wide configure while ServletConfig is used to configure and provide initialization parameter to one Servlet. See ServletContext vs ServletConfig for more differences.

[](https://1.bp.blogspot.com/-PQdK2brlyDU/XoVEwy1kGOI/AAAAAAAAdt0/3jMflg3wNMU_iSzAJb1DLbC0Z62zB9HBgCLcBGAsYHQ/s1600/servlet-context-and-servlet-config.png)

**Q: Which open source tag library have you used ?**

This is an interesting Servlet JSP questions and gives an opportunity to show how many tag library you are familiar with and which ones have you used. Most J2EE programmer answer this question with saying JSTL core tag library, Struts tag library , Spring tag library or display tag, which is quite popular tag library to display tabular data and provides lot of feature out of box e.g. paging, sorting and export functionality.

**Q: What is difference between GET and POST method in HTTP protocol?**

Another classical web interview question, not specific to Servlet or JSP but very important in context of web development which is based on HTTP protocol. There several differences between GET and POST method including length of data required to Send to Server. GET is less secure and can only send limited data hence not useful to transfer sensitive information. See GET vs POST HTTP method for more differences.

**Q : What does load-on-start-up element in web.xml do?**

One of the tough Servlet JSP Interview question especially to 2 years experience guy, who may not be exposed to all tags of web.xml. load-on-startup is related to loading of Servlet. See What is load-on-startup tag in web.xml for more details.

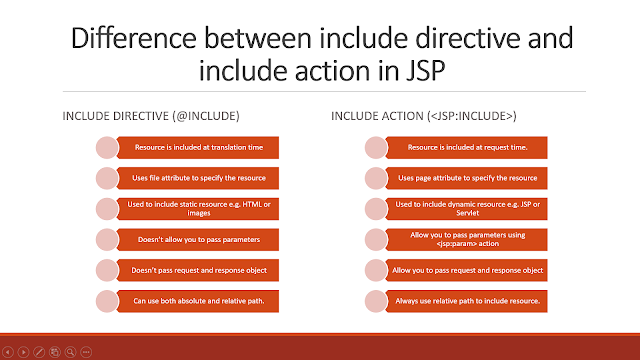
These were some Servlet and JSP Interview questions and answer you can look for quick revision purpose. If you have faced any interesting Servlet JSP question or if you are looking answer for any Servlet JSP question than please share.

**Q Difference between include directive and include action in JSP**

Difference between include directive and include action  
Even though both include directive and include action allows you to include response of one JSP into another, they are quite different from each other e.g. include directive is processed at translation time and used to include the static resources e.g. HTML files, images and CSS etc, while the <jsp:include> action is processed at request time and used to include more dynamic resources e.g. JSP or Servlet. What is the difference between include directive and include action is also one of the most popular JSP interview questions, mostly asked either at telephonic round or first few round of Java web developer interviews? Let's see a couple of more difference between these two to answer this question in more detail. Btw, if you are new in JSP and just learning and preparing for interviews at the same time, then you should also look at Head First Servlet and JSP, one of the best books to learn and prepare for Servlet JSP interviews and certification at the same time.

JSP include directive vs include action

The include directive is one the three directive supported by JSP e.g. @taglib, @include and @page, while the include action is one of the many standard actions supported by JSP i.e. <jsp:include>, <jsp:forward>, <jsp:useBean>, <jsp:setProperty>, <jsp:getProperty>, <jsp:param> and <jsp:plugin>.  
  
Both are used to include the response of one resource  into another e.g. composing JSP page with header portion coming form header.jsp, footer coming from footer.jsp and sidebar coming from sidebar.jsp. But as I said, they are very different from each other.  
  
  
  
  
Here are some important difference between include directive and include action in JSP:  
  
1) The most cirtical difference between @include and <jsp:include> is that the include directive is processed at the translation time but include action i.e. <jsp:include> is processed at the request time i.e. when the request comes for processing.  
  
2) Second important difference between include directive and include action is thtat an include directive can use both relative and absolute path but the include action always uses relative path.  
  
3) Third key diference between include action and include directive is that @include can only include contents of resource e.g. HTML or CSS file but it will not process the dynamic resource, while the include action i.e. <jsp:include> process the dynamic resource and result will be added to calling JSP.  
  
4) One more practical difference between include directive and include action is that with include directive, you can not pass any other parameter e.g request, response or any configuraiton, but with include action, we can pass another parameter also using <jsp:param> tags.  
  
5) Fifth difference is extension of previous difference i.e. with include directive, you cannot pass the request or response object to calling jsp to included file or JSP or vice versa, but with include action, it's possible to pass request/response object.  
  
6) Last but not the least difference between include directive and include action is that former uses the file attribute to specify the resource to be included and that's why know as file include as well. On the other hand include action uses page attribute to specify the resource to be included, hence also known as page include in JSP.  
  
Here is a nice summary of all the differences between file include and page include i.e. include action in JSP:

[](https://pluralsight.pxf.io/c/1193463/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fjava-web-fundamentals)

That's all about the difference between include directive and include action in JSP. As I said, it's one of the classical questions from Servlet and JSP interviews and frequently asked during the telephonic round of Interview. While answering this question you should remember to mention the key points like include action does request time inclusion and you can include dynamic resources like response of another JSP or Servlet using include action

Difference between ServletConfig and ServletContext in JSP Servlet J2EE

Difference between ServletConfig and ServletContext

ServletContext and ServletConfig these two are important interface of Servlet API which is used by Java J2EE programmer during web application development. Correct understanding of What is ServletContext and ServletConfig is very important for any J2EE application developer. Apart from that Difference between ServletContext and ServletConfig are a popular Servlet JSP interview questions and mostly asked on both fresher and experienced Java programmer during J2EE interviews. Both ServletContext and ServletConfig are basically configuration objects which are used by servlet container to initialize various parameter of web application. But they have some difference in terms of scope and availability so first we see what is ServletContext and ServletConfig objects are and then we will see difference between ServletConfig and ServletContext in Java J2EE.

Q. What is ServletConfig in JSP Servlet

ServletConfig  in an interface in Servlet API and ServletConfig object represents or used to initialize single servlet in web application by servlet container. Inside deployment descriptor known as web.xml,  we define Servlet initialization parameter related to that servlet inside &lt;init-param&gt;&lt;/init-param&gt; tag. its a set of name /value pair. Following is signature of ServletConfig interface :

public interface ServletConfig

By using ServletConfig and combination of init-param you can configure any Servlet in J2EE environment.

**Q. What is ServletContext in Servlet JSP**

ServletContext is even more important than ServletConfig and its one per web application, also known as Context. This object is common for all the servlet and they use this object to communicate with the servlet container to get the detail of whole web application or execution environment important thing to remember is, it represents a web application in single JVM.

Signature: public interface ServletContext

By using ServletContext object you can share objects to any Servlet or JSP in whole web application. See how to get ServletContext in Servlet JSP and Struts to find out different ways of getting ServletContext in J2EE application.

**Q. Difference between ServletContext vs ServletConfig**

Now let’s see difference between ServletContext and ServletConfig in Servlets JSP in tabular format

|  |  |
| --- | --- |
| Servlet Config | Servlet Context |
| Servlet config object represent single servlet | It represent whole web application running on particular JVM and common for all the servlet |
| Its like local parameter associated with **particular servlet** | Its like global parameter associated with **whole application** |
| It’s a name value pair defined inside the servlet section of web.xml file so it has servlet wide scope | ServletContext has application wide scope so define outside of servlet tag in web.xml file. |
| **getServletConfig() method** is used to get the config object | **getServletContext() method** is  used to get the context object. |
| for example shopping cart of a user is a specific to particular user so here we can use servlet config | To get the MIME type of a file or application session related information is stored using servlet context object. |

Difference between GenericServlet vs HttpServlet in Servlet JSP - J2EE question

Difference between GenericServlet and HttpServlet is one of the classic Servlet Interview Question, asked on many Servlet and JSP Interviews on 2 to 4 years experience developers. Since both GenericServlet and HttpServlet forms basis of Servlets its important to know What are they and What is main difference between them. From common sense and there names, its obvious that GenericServlet is a generic and protocol independent implementation of Servlet interface while HttpServlet implements HTTP protocol specifics. If you are working in Java web application or J2EE projects, you are most likely to deal with HttpServlet all time as HTTP is main communication protocol of web. In this Servlet JSP article we will outline some important difference between HttpServlet and GenericServlet which is worth knowing and remembering.

**GenericServlet vs HttpServlet**

Here is my list of difference between HttpServlet and GenericServlet in Java Servlet API :

1) GenericServlet provides abstract service(ServletRequest, ServletResponse) method to implement which gets called by container whenever it receives request for processing, On the other hand HttpServlet overrides service method and provides callback on doGet(HttpServletRequest request, HttpServletResponse) and doPost(HttpServletRequest request, HttpServletResponse response) whenever it receives HTTP request from GET or POST method. It also provides several other method based upon various HTTP methods of sending request e.g. doPut() or doDelete() to handle HTTP PUT and HTTP DELETE request.

2) Another difference between GenericServlet and HttpServlet is that later is a subclass of GenericServlet and inherit properties of GenericServlet.

3) Generic servlet provides a rather easier way to extend Servlet, its enough to override service method to implement GenericServlet. Apart from extending Servlet interface, it also implements ServletConfig interface and provides way to accept initialization parameter passed to Servlet from web.xml e.g. by using getInitParamter().

That's all on difference between GenericServlet and HttpServlet in Servlet API. Its very rare you are going to use GenericServlet but its good to know basics as its mostly asked in various Servlet interviews as well. HttpServlet is primary Servlet class which is used in Java web application from handling client request.