Servlet Interview Questions and Answers

What is different between web server and application server?

A web server responsibility is to handler HTTP requests from client browsers and respond with HTML response. A web server understands HTTP language and runs on HTTP protocol.
Apache Web Server is kind of a web server and then we have specific containers that can execute servlets and JSPs known as servlet container, for example Tomcat.
Application Servers provide additional features such as Enterprise JavaBeans support, JMS Messaging support, Transaction Management etc. So we can say that Application server is a web server with additional functionalities to help developers with enterprise applications.

Which HTTP method is non-idempotent?

A HTTP method is said to be idempotent if it returns the same result every time. HTTP methods GET, PUT, DELETE, HEAD, and OPTIONS are idempotent method and we should implement our application to make sure these methods always return same result. HTTP method POST is non-idempotent method and we should use post method when implementing something that changes with every request.

For example, to access an HTML page or image, we should use GET because it will always return the same object but if we have to save customer information to database, we should use POST method. Idempotent methods are also known as safe methods and we don't care about the repetitive request from the client for safe methods.

What is the difference between GET and POST method?

GET is a safe method (idempotent) where POST is non-idempotent method.
We can send limited data with GET method and it's sent in the header request URL whereas we can send large amount of data with POST because it's part of the body.
GET method is not secure because data is exposed in the URL and we can easily bookmark it and send similar request again, POST is secure because data is sent in request body and we can't bookmark it.
GET is the default HTTP method whereas we need to specify method as POST to send request with POST method.
Hyperlinks in a page uses GET method.

What is MIME Type?

The "Content-Type" response header is known as MIME Type. Server sends MIME type to client to let them know the kind of data it's sending. It helps client in rendering the data for user. Some of the mostly used mime types are text/html, text/xml, application/xml etc.

We can use ServletContext getMimeType() method to get the correct MIME type of the file and use it to set the response content type. It's very useful in downloading file through servlet from server.

What is a web application and what is it's directory structure?

Web Applications are modules that run on server to provide both static and dynamic content to the client browser. Apache web server supports PHP and we can create web application using PHP. Java provides web application support through Servlets and JSPs that can run in a servlet container and provide dynamic content to client browser.

Java Web Applications are packaged as Web Archive (WAR) and it has a defined structure like below image.

What is a servlet?

Java Servlet is server side technologies to extend the capability of web servers by providing support for dynamic response and data persistence.

The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing our own servlets.
All servlets must implement the javax.servlet.Servlet interface, which defines servlet lifecycle methods. When implementing a generic service, we can extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet() and doPost(), for handling HTTP-specific services.

Most of the times, web applications are accessed using HTTP protocol and thats why we mostly extend HttpServlet class. Servlet API hierarchy is shown in below image.

What are the advantages of Servlet over CGI?

Servlet technology was introduced to overcome the shortcomings of CGI technology.

Servlets provide better performance that CGI in terms of processing time, memory utilization because servlets uses benefits of multithreading and for each request a new thread is created, that is faster than loading creating new Object for each request with CGI.
Servlets and platform and system independent, the web application developed with Servlet can be run on any standard web container such as Tomcat, JBoss, Glassfish servers and on operating systems such as Windows, Linux, Unix, Solaris, Mac etc.
Servlets are robust because container takes care of life cycle of servlet and we don't need to worry about memory leaks, security, garbage collection etc.
Servlets are maintainable and learning curve is small because all we need to take care is business logic for our application.

What is ServletConfig object?

javax.servlet.ServletConfig is used to pass configuration information to Servlet. Every servlet has it's own ServletConfig object and servlet container is responsible for instantiating this object. We can provide servlet init parameters in web.xml file or through use of WebInitParam annotation. We can use getServletConfig() method to get the ServletConfig object of the servlet.

What are common tasks performed by Servlet Container?

Servlet containers are also known as web container, for example Tomcat. Some of the important tasks of servlet container are:

Communication Support: Servlet Container provides easy way of communication between web client (Browsers) and the servlets and JSPs. Because of container, we don't need to build a server socket to listen for any request from web client, parse the request and generate response. All these important and complex tasks are done by container and all we need to focus is on business logic for the applications.

Lifecycle and Resource Management: Servlet Container takes care of managing the life cycle of servlet. From the loading of servlets into memory, initializing servlets, invoking servlet methods and to destroy them. Container also provides utility like JNDI for resource pooling and management.

Multithreading Support: Container creates new thread for every request to the servlet and provide them request and response objects to process. So servlets are not initialized for each request and saves time and memory.

JSP Support: JSPs doesn't look like normal java classes but every JSP in the application is compiled by container and converted to Servlet and then container manages them like other servlets.

Miscellaneous Task: Servlet container manages the resource pool, perform memory optimizations, execute garbage collector, provides security configurations, support for multiple applications, hot deployment and several other tasks behind the scene that makes a developer life easier.

What is ServletContext object?

javax.servlet.ServletContext interface provides access to web application parameters to the servlet. The ServletContext is unique object and available to all the servlets in the web application. When we want some init parameters to be available to multiple or all of the servlets in the web application, we can use ServletContext object and define parameters in web.xml using <context-param> element. We can get the ServletContext object via the getServletContext() method of ServletConfig. Servlet containers may also provide context objects that are unique to a group of servlets and which is tied to a specific portion of the URL path namespace of the host.

ServletContext is enhanced in Servlet Specs 3 to introduce methods through which we can programmatically add Listeners and Filters and Servlet to the application. It also provides some utility methods such as getMimeType(), getResourceAsStream() etc.

What is difference between ServletConfig and ServletContext?

Some of the differences between ServletConfig and ServletContext are:

ServletConfig is a unique object per servlet whereas ServletContext is a unique object for complete application.

ServletConfig is used to provide init parameters to the servlet whereas ServletContext is used to provide application level init parameters that all other servlets can use.

We can't set attributes in ServletConfig object whereas we can set attributes in ServletContext that other servlets can use in their implementation.

What is Request Dispatcher?

RequestDispatcher interface is used to forward the request to another resource that can be HTML, JSP or another servlet in same application. We can also use this to include the content of another resource to the response. This interface is used for inter-servlet communication in the same context.

There are two methods defined in this interface:

void forward(ServletRequest request, ServletResponse response) – forwards the request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
void include(ServletRequest request, ServletResponse response) – includes the content of a resource (servlet, JSP page, HTML file) in the response.

We can get RequestDispatcher in a servlet using ServletContext getRequestDispatcher(String path) method. The path must begin with a / and is interpreted as relative to the current context root.

What is difference between PrintWriter and ServletOutputStream?

PrintWriter is a character-stream class whereas ServletOutputStream is a byte-stream class. We can use PrintWriter to write character based information such as character array and String to the response whereas we can use ServletOutputStream to write byte array data to the response.

We can use ServletResponse getWriter() to get the PrintWriter instance whereas we can use ServletResponse getOutputStream() method to get the ServletOutputStream object reference.

Can we get PrintWriter and ServletOutputStream both in a servlet?

We can't get instances of both PrintWriter and ServletOutputStream in a single servlet method, if we invoke both the methods; getWriter() and getOutputStream() on response; we will get java.lang.IllegalStateException at runtime with message as other method has already been called for this response.

How can we create deadlock situation in servlet?

We can create deadlock in servlet by making a loop of method invocation, just call doPost() method from doGet() method and doGet() method to doPost() method to create deadlock situation in servlet.

What is the use of servlet wrapper classes?

Servlet HTTP API provides two wrapper classes – HttpServletRequestWrapper and HttpServletResponseWrapper. These wrapper classes are provided to help developers with custom implementation of servlet request and response types. We can extend these classes and override only specific methods we need to implement for custom request and response objects. These classes are not used in normal servlet programming.

What is SingleThreadModel interface?
SingleThreadModel interface was provided for thread safety and it guarantees that no two threads will execute concurrently in the servlet's service method. However SingleThreadModel does not solve all thread safety issues. For example, session attributes and static variables can still be accessed by multiple requests on multiple threads at the same time, even when SingleThreadModel servlets are used. Also it takes out all the benefits of multithreading support of servlets, thats why this interface is Deprecated in Servlet 2.4.

Do we need to override service() method?
When servlet container receives client request, it invokes the service() method which in turn invokes the doGet(), doPost() methods based on the HTTP method of request. I don't see any use case where we would like to override service() method. The whole purpose of service() method is to forward to request to corresponding HTTP method implementations. If we have to do some pre-processing of request, we can always use servlet filters and listeners.

Is it good idea to create servlet constructor?
We can define a constructor for servlet but I don't think its of any use because we won't be having access to the ServletConfig object until unless servlet is initialized by container. Ideally if we have to initialize any resource for servlet, we should override init() method where we can access servlet init parameters using ServletConfig object.

What is difference between GenericServlet and HttpServlet?

GenericServlet is protocol independent implementation of Servlet interface whereas HttpServlet is HTTP protocol specific implementation. Most of the times we use servlet for creating web application and that's why we extend HttpServlet class. HttpServlet class extends GenericServlet and also provide some other methods specific to HTTP protocol.

What is the inter-servlet communication?
When we want to invoke another servlet from a servlet service methods, we use inter-servlet communication mechanisms. We can invoke another servlet using RequestDispatcher forward() and include() methods and provide additional attributes in request for other servlet use.

Are Servlets Thread Safe? How to achieve thread safety in servlets?

HttpServlet init() method and destroy() method are called only once in servlet life cycle, so we don't need to worry about their synchronization. But service methods such as doGet() or doPost() are getting called in every client request and since servlet uses multithreading, we should provide thread safety in these methods.

If there are any local variables in service methods, we don't need to worry about their thread safety because they are specific to each thread but if we have a shared resource then we can use synchronization to achieve thread safety in servlets when working with shared resources.

The thread safety mechanisms are similar to thread safety in standalone java application, read more about them at Thread Safety in Java.

What is servlet attributes and their scope?

Servlet attributes are used for inter-servlet communication, we can set, get and remove attributes in web application. There are three scopes for servlet attributes – request scope, session scope and application scope.

ServletRequest, HttpSession and ServletContext interfaces provide methods to get/set/remove attributes from request, session and application scope respectively.

Servlet attributes are different from init parameters defined in web.xml for ServletConfig or ServletContext.

How do we call one servlet from another servlet?

We can use RequestDispatcher forward() method to forward the processing of a request to another servlet. If we want to include the another servlet output to the response, we can use RequestDispatcher include() method.

How can we invoke another servlet in a different application?
We can't use RequestDispatcher to invoke servlet from another application because it's specific for the application. If we have to forward the request to a resource in another application, we can use ServletResponse sendRedirect() method and provide complete URL of another servlet. This sends the response to client with response code as 302 to forward the request to another URL. If we have to send some data also, we can use cookies that will be part of the servlet response and sent in the request to another servlet.

What is difference between ServletResponse sendRedirect() and RequestDispatcher forward() method?

RequestDispatcher forward() is used to forward the same request to another resource whereas ServletResponse sendRedirect() is a two step process. In sendRedirect(), web application returns the response to client with status code 302 (redirect) with URL to send the request. The request sent is a completely new request.

forward() is handled internally by the container whereas sednRedirect() is handled by browser.

We should use forward() when accessing resources in the same application because it's faster than sendRedirect() method that required an extra network call.

In forward() browser is unaware of the actual processing resource and the URL in address bar remains same whereas in sendRedirect() URL in address bar change to the forwarded resource.

forward() can't be used to invoke a servlet in another context, we can only use sendRedirect() in this case.

Why HttpServlet class is declared abstract?

HttpServlet class provide HTTP protocol implementation of servlet but it's left abstract because there is no implementation logic in service methods such as doGet() and doPost() and we should override at least one of the service methods. That's why there is no point in having an instance of HttpServlet and is declared abstract class.

What are the phases of servlet life cycle?

We know that Servlet Container manages the life cycle of Servlet, there are four phases of servlet life cycle.

Servlet Class Loading – When container receives request for a servlet, it first loads the class into memory and calls it's default no-args constructor.
Servlet Class Initialization – Once the servlet class is loaded, container initializes the ServletContext object for the servlet and then invoke it's init method by passing servlet config object. This is the place where a servlet class transforms from normal class to servlet.
Request Handling – Once servlet is initialized, its ready to handle the client requests. For every client request, servlet container spawns a new thread and invokes the service() method by passing the request and response object reference.
Removal from Service – When container stops or we stop the application, servlet container destroys the servlet class by invoking it's destroy() method.

What are life cycle methods of a servlet?

Servlet Life Cycle consists of three methods:

public void init(ServletConfig config) – This method is used by container to initialize the servlet, this method is invoked only once in the lifecycle of servlet.
public void service(ServletRequest request, ServletResponse response) – This method is called once for every request, container can't invoke service() method until unless init() method is executed.
public void destroy() – This method is invoked once when servlet is unloaded from memory.

Why we should override only no-agrs init() method.?

If we have to initialize some resource before we want our servlet to process client requests, we should override init() method. If we override init(ServletConfig config) method, then the first statement should be super(config) to make sure superclass init(ServletConfig config) method is invoked first. That's why GenericServlet provides another helper init() method without argument that get's called at the end of init(ServletConfig config) method. We should always utilize this method for overriding init() method to avoid any issues as we may forget to add super() call in overriding init method with ServletConfig argument.

What is URL Encoding?

URL Encoding is the process of converting data into CGI form so that it can travel across the network without any issues. URL Encoding strip the white spaces and replace special characters with escape characters. We can use java.net.URLEncoder.encode(String str, String unicode) to encode a String. URL Decoding is the reverse process of encoding and we can use java.net.URLDecoder.decode(String str, String unicode) to decode the encoded string. For example "Pankaj's Data" is encoded to "Pankaj %27s+Data".

What are different methods of session management in servlets?

Session is a conversional state between client and server and it can consists of multiple request and response between client and server. Since HTTP and Web Server both are stateless, the only way to maintain a session is when some unique information about the session (session id) is passed between server and client in every request and response.

Some of the common ways of session management in servlets are:

- User Authentication
- HTML Hidden Field
- Cookies
- URL Rewriting
- Session Management API

Read more about these session management approaches in detail at Servlet Session Management Tutorial.

What is URL Rewriting?

We can use HttpSession for session management in servlets but it works with Cookies and we can disable the cookie in client browser. Servlet API provides support for URL rewriting that we can use to manage session in this case.

The best part is that from coding point of view, it's very easy to use and involves one step – encoding the URL. Another good thing with Servlet URL Encoding is that it's a fallback approach and it kicks in only if browser cookies are disabled.

We can encode URL with HttpServletResponse encodeURL() method and if we have to redirect the request to another resource and we want to provide session information, we can use encodeRedirectURL() method.

How does Cookies work in Servlets?

Cookies are used a lot in web client-server communication, it's not something specific to java. Cookies are text data sent by server to the client and it gets saved at the client local machine.

Servlet API provides cookies support through javax.servlet.http.Cookie class that implements Serializable and Cloneable interfaces.

HttpServletRequest getCookies() method is provided to get the array of Cookies from request, since there is no point of adding Cookie to request, there are no methods to set or add cookie to request.

Similarly HttpServletResponse addCookie(Cookie c) method is provided to attach cookie in response header, there are no getter methods for cookie.

How to notify an object in session when session is invalidated or timed-out?

If we have to make sure an object gets notified when session is destroyed, the object should implement javax.servlet.http.HttpSessionBindingListener interface. This interface defines two callback methods – valueBound() and valueUnbound() that we can define to implement processing logic when the object is added as attribute to the session and when session is destroyed.

What is the difference between encodeRedirectUrl and encodeURL?

HttpServletResponse provide method to encode URL in HTML hyperlinks so that the special characters and white spaces are escaped and append session id to the URL. It behaves similar to URLEncoder encode method with additional process to append jsessionid parameter at the end of the URL.

However HttpServletResponse encodeRedirectUrl() method is used specially for encode the redirect URL in response.

So when we are providing URL rewriting support, for hyperlinks in HTML response, we should use encodeURL() method whereas for redirect URL we should use encodeRedirectUrl() method.

Why do we have servlet filters?

Servlet Filters are pluggable java components that we can use to intercept and process requests before they are sent to servlets and response after servlet code is finished and before container sends the response back to the client.

Some common tasks that we can do with filters are:

- Logging request parameters to log files.
- Authentication and autherization of request for resources.
- Formatting of request body or header before sending it to servlet.
- Compressing the response data sent to the client.
- Alter response by adding some cookies, header information etc.

What is the effective way to make sure all the servlets are accessible only when user has a valid session?

We know that servlet filters can be used to intercept request between servlet container and servlet, we can utilize it to create authentication filter and check if request contains a valid session or not.

Why do we have servlet listeners?

We know that using ServletContext, we can create an attribute with application scope that all other servlets can access but we can initialize ServletContext init parameters as String only in deployment descriptor (web.xml). What if our application is database oriented and we want to set an attribute in ServletContext for Database Connection.

If you application has a single entry point (user login), then you can do it in the first servlet request but if we have multiple entry points then doing it everywhere will result in a lot of code redundancy. Also if database is down or not configured properly, we won't know until first client request comes to server. To handle these scenario, servlet API provides Listener interfaces that we can implement and configure to listen to an event and do certain operations.

How to handle exceptions thrown by application with another servlet?

If you notice, doGet() and doPost() methods throw ServletException and IOException. Since browser understand only HTML, when our application throw exception, servlet container processes the exception and generate a HTML response. Same goes with other error codes like 404, 403 etc.

Servlet API provides support for custom Exception and Error Handler servlets that we can configure in deployment descriptor, the whole purpose of these servlets are to handle the Exception or Error raised by application and send HTML response that is useful for the user. We can provide link to application home page or some details to let user know what went wrong.

We can configure them in web.xml like below:

```
<error-page>
   <error-code>404</error-code>
   <location>/AppExceptionHandler</location>
</error-page>

<error-page>
   <exception-type>javax.servlet.ServletException</exception-type>
   <location>/AppExceptionHandler</location>
</error-page>
```

What is a deployment descriptor?

Deployment descriptor is a configuration file for the web application and it's name is web.xml and it resides in WEB-INF directory. Servlet container use this file to configure web application servlets, servlet config params, context init params, filters, listeners, welcome pages and error handlers.

With servlet 3.0 annotations, we can remove a lot of clutter from web.xml by configuring servlets, filters and listeners using annotations.

How to make sure a servlet is loaded at the application startup?

Usually servlet container loads a servlet on the first client request but sometimes when the servlet is heavy and takes time to loads, we might want to load it on application startup. We can use load-on-startup element with servlet configuration in web.xml file or use WebServlet annotation loadOnStartup variable to tell container to load the servlet on system startup.

```
<servlet>
   <servlet-name>foo</servlet-name>
   <servlet-class>com.foo.servlets.Foo</servlet-class>
   <load-on-startup>5</load-on-startup>
</servlet>
```

The load-on-startup value should be int, if it's negative integer then servlet container will load the servlet based on client requests and requirement but if it's 0 or positive, then container will load it on application startup.

If there are multiple servlets with load-on-startup value such as 0,1,2,3 then lower integer value servlet will be loaded first.

How to get the actual path of servlet in server?

We can use following code snippet to get the actual path of the servlet in file system.

```
getServletContext().getRealPath(request.getServletPath())
```

How to get the server information in a servlet?

We can use below code snippet to get the servlet information in a servlet through servlet context object.

```
getServletContext().getServerInfo()
```

Write a servlet to upload file on server.

File Upload and Download and common tasks in a java web application. Unfortunately Servlet API doesn't provide easy methods to upload file on server, so we can use Apache FileUpload jar to make our life easier.
Please read File Upload Servlet post that provide all the necessary details with example program to upload and download file using servlets.

How do we go with database connection and log4j integration in servlet?

If you work with database connection a lot in your web application, its best to initialize it in a servlet context listener and set it as a context attribute for other servlets to use.
Integrating Log4j is also very easy in web applications, all we need is a log4j configuration XML or property file and then configure it in a servlet context listener.

How to get the IP address of client in servlet?
We can use request.getRemoteAddr() to get the client IP address in servlet.

What are important features of Servlet 3?
Servlet Specs 3.0 was a major release and some of the important features are:

Servlet Annotations: Prior to Servlet 3, all the servlet mapping and it's init parameters were used to defined in web.xml, this was not convenient and more error prone when number of servlets are huge in an application.

Servlet 3 introduced use of java annotations to define a servlet, filter and listener servlets and init parameters. Some of the important Servlet API annotations are WebServlet, WebInitParam, WebFilter and WebListener. Read more about them at Servlet 3 annotations.

Web Fragments: Prior to servlet specs 3.0, all the web application configurations are required to be present in the web.xml that makes it cluttered with lot of elements and chances of error increases. So servlet 3 specs introduced web fragments where we can have multiple modules in a single web application, all these modules should have web-fragment.xml file in META-INF directory. We can include all the elements of web.xml inside the web-fragment.xml too. This helps us in dividing our web application into separate modules that are included as JAR file in the web application lib directory.

Adding Web Components dynamically: We can use ServletContext object to add servlets, filters and listeners programmatically. This helps us in building dynamic system where we are loading a component only if we need it. These methods are addServlet(), addFilter() and addListener() defined in the servlet context object.

Asynchronous Processing: Asynchronous support was added to delegate the request processing to another thread rather than keeping the servlet thread busy. It can increase the throughput performance of the application. This is an advance topic and I recommend to read Async Servlet tutorial.

What are different ways for servlet authentication?

Servlet Container provides different ways of login based servlet authentication:

HTTP Basic Authentication
HTTP Digest Authentication
HTTPS Authentication
Form Based Login: A standard HTML form for authentication, advantage is that we can change the login page layout as our application requirements rather than using HTTP built-in login mechanisms.

How can we achieve transport layer security for our web application?

We can configure our servlet container to use SSL for message communication over the network. To configure SSL on Tomcat, we need a digital certificate that can be created using Java keytool for development environment. For production environment, you should get the digital certificate from SSL certificate providers, for example, Verisign or Entrust.