**JDBC**

**1) What is the JDBC?**

**Ans:**-JDBC stands for Java Database Connectivity. JDBC is a Java API that communicates with the database and execute SQLquery.

**2) What is a JDBC driver and how many JDBC drivers are available?**

**Ans:-** JDBC driver contains classes and interfaces that help Java application and database.

There are 4 types of JDBC drivers.

* Type 1 driver or JDBC-ODBC bridge driver.
* Type 2 driver or Native-API, partly Java driver.
* Type 3 driver or Network Protocol, pure Java driver.
* Type 4 driver or Native-protocol, pure Java driver.

**3) Which JDBC driver is the fastest driver?**

**Ans:**-Type 4 driver or Native-protocol, pure Java driver, is the fastest driver.

**4) What are the JDBC API components?**

**Ans:**-

* JDBC API
* JDBC Driver Manager
* JDBC Test Suite
* JDBC-ODBC Bridge

**5) What are the JDBC statements?**

**Ans:**-There are 3 types of JDBC Statements, as given below:

**Statement**:  It will execute SQL query (static SQL query) against the database.

**Prepared Statement:**  Used when we want to execute SQL statement repeatedly.  Input data is dynamic and  taken input at the run time.

**Callable Statement:** Used when we want to execute stored procedures.

**6) How can we execute stored procedures?**

**Ans:**-Stored procedures can be executed using JDBCcallable statement. Here is the code.

Connection conn = null;

CallableStatement callStmt = conn.prepareCall("{call myStoreproc(?, ?)}");

callStmt.setString(1, "abcdefg");

**7) What are the advantages of using PreparedStatement in Java?**

**Ans:**-Prepared Statement is used to execute same SQL statements repeatedly. The prepared statement is compiled only once even though it used “n” number of times

**8) What is ResultSet?**

**Ans:**-The java.sql.ResultSet interface means the result set of a SQL query. It means a cursor is pointing a row of a table; it points to before the first row.

**9) What are types of ResultSet?**

**Ans:**-There are three types of ResultSet is available. If we do not declare any ResultSet that means we are calling TYPE\_FORWARD\_ONLY

TYPE\_FORWARD\_ONLY: cursor can move only forward.

TYPE\_SCROLL\_INSENSITIVE: cursor can move forward and backward but not sensitive.

TYPE\_SCROLL\_SENSITIVE: cursor can move forward and backward, but it is sensitive

**10) Explain the difference between RowSet vs. ResultSet in JDBC?**

**Ans:**-In a ResultSet handle connection to a DB, we cannot make Result as a serialized object.

Because of above issue, we cannot pass  Resultset across the network.

RowSet extends the ResultSet interface, so it holds all methods from ResultSet. RowSet is serialized.

So, we can pass Rowset from one class to another class because it has no connection with the database.

**11) Why would you use setAutoCommit(false) in JDBC?**

**Ans:**-If you want to turn Off the Auto Commit then set connection.setAutoCommit(false)

**12) What are database warnings in JDBC and how can we handle database warnings in JDBC?**

**Ans:**-SQL warning or Database warning is the subclass of SQLException class. We can handle it by using getWarnings() method on Connection, Statement, and ResultSet

**13) Can I get a null ResultSet?**

**Ans:**-No, we cannot get null Resultset. ResultSet.next() can return null if the next record does not contain a row.

**14) What do you mean by Metadata and why we are using it?**

**Ans:**-Metadata means data or information about other data. We use metadata to get database product version, driver name, the total number of tables and views.

**15) What is the difference between executing, executeQuery, executeUpdate in JDBC?**

**Ans:**-execute(): it can be used for any kind of SQL Query.

executeQuery() : it can be used for select query.

executeUpdate(): it can be used to change/update table.

**16) What is database connection pooling? Advantages of using a connection pool?**

**Ans:**-Connection pooling means connections will be stored in the cache and we can reuse them in future.

Advantage:

* It is faster
* Connection pooling becomes easier to diagnose and analyze database connection.

**17) What is the function of DriverManager class?**

**Ans:**-It is an interface between user and drivers. DriverManager tracks all the activity between a database and the appropriate driver.

**18) What is the meaning of batch updates?**

**Ans:**-Batch updates means executing a set/group of SQL queries all at once.

Batch updates can be used only for insert, update and delete but not for select query.

**19) How many packages are available in JDBC API?**

**Ans:**-Two types of packages are available in JDBC API

1. java.sql (2) javax.sql

**20) What is the return type of execute, executeQuery and executeUpdate?**

**Ans:**-Return type of execute is Boolean

Return type of executeQuery is ResultSet object

Return type of executeUpdate is int

**21) Result Set’s index Starts with 0 or 1?**

**Ans:**-Result Set’s index starts with 1.

**22) What is the role of Class.forName while loading drivers?**

**Ans:**-Class.forName creates an instance of JDBC driver and register with DriverManager.

**23) JDBC-ODBC Bridge is multi-threaded or not?**

**Ans:**-No, JDBC-ODBC Bridge uses synchronized methods to serialize all the calls made to ODBC.

**24) Which interface handles transaction management in JDBC?**

**Ans:**-Connection interface handles transaction management in JDBC. It provides method for commit (), rollback () etc.

**25) Why “No suitable driver” error occurs?**

**Ans:**-No suitable driver” occurs when we are calling DriverManager.getConnection method,

it may occur because of following reasons:

unable to load exact JDBC drivers before calling the getConnection method.

It may be invalid or wrong JDBC URL.

**26) Prepared Statements are faster. Why?**

**Ans:**-Prepared statement execution is faster than direct execution because the statement is compiled only once. Prepared statements & JDBC driver are connected with each other during execution, and there are no connection overheads.

**27) Is it possible to connect to multiple databases? Using single statement can we update or extract data from two or three or many databases?**

**Ans:**-Yes, it is possible to connect to multiple databases, at the same time, but it depends on the specific driver.

To update and extract data from the different database we can use the single statement. But we need middleware to deal with multiple databases or a single database.

**28) Tell me difference between setMaxRows(int) and SetFetchSize(int)?**

**Ans:**-

|  |  |
| --- | --- |
| **setMaxRows(int)** | **SetFetchSize(int)?** |
| Defines how many rows a resultset can contain at a time | Defines the number of rows that will be read from the database. |

**29) Tell me about special characters?**

**Ans:**-A special character is preceded by an escape character. Example –

SELECT NAME FROM TABLE WHERE NAME LIKE ‘\\_%’ {escape ‘\’}

**30) What is the meaning of “dirty read” in the database?**

**Ans:**-Dirty read means “read the value which may be correct or may not be correct.”

**31) What do you mean by two phase commits?**

**Ans:**-Two phase commit is used in distributed transaction process. If any transaction is executing and it will affect multiple databases. Two phase commits will be used to make all databases synchronized with each other.

**32) How many locking systems are there in JDBC?**

**Ans:**-Two types of locking are available in JDBC by which we can handle more than one user.

If two users are viewing the same record, then no locking is done. If one user is updating a record and the second user is also updating the same record.  At that time, we are going to use locking.

**Optimistic Locking**: it will lock the record only when we are going to “update.”

**Pessimistic Locking**: it will lock the record from the “select” to view, update and commit time.

**33) What are the exceptions in JDBC?**

**Ans:**-There are four types of exceptions in JDBC.

batchUpdate Exception

Data Truncation

SQL Exception

SQL Warning

**34) Give steps to connect to the DB using JDBC?**

**Ans:**-There are two ways to connecting database using JDBC

**Using DriverManager:**

It will load the driver class with the help of class.forName(driver class) and Class.forName().

Post Loading it will pass the control to DriverManager.

DriverManager.getConnection() will create the connection to access the database.

**Using DataSource:**

For DataSource, no need to use DriverManager with the help of JNDI.  It will lookup the DataSource from Naming service server. DataSource.getConnection() method will return Connection object to the DB.

**35) What are the new features available in JDBC 4.0?**

**Ans:**-The new features are

Auto loading by JDBC driver class

Enhanced Connection Management

RowId SQL enabled

Dataset implemented by SQL by using Annotations

Enhancements of SQL exception handling

Supporting SQL XML files

**36) What are the packages are used in JDBC?**

**Ans:**-8 packages are used in JDBC –

* sql.Driver
* Connection
* Statement
* PreparedStatement
* CallableStatement
* ResultSet
* ResultSetMetaData
* DatabaseMetaData

**37) How many RowSet are available in JDBC?**

**Ans:**-There are two types of row sets are available:

**Connected** – A connected RowSet object connects to the database instantaneously. If the application terminates then connected RowSet object also terminates.

**Disconnected** – A disconnected RowSet object connects to the database after execution of the required query.

**38) What is the meaning of connection?**

**Ans:**-Connection interface consists of methods for interaction with the database.

**39) Explain JDBC Savepoint?**

**Ans:**-A savepoint represents a point that the current transaction can roll back to. Instead of rolling all its changes back, it can choose to roll back only some of them.

**40) List the advantages of using DataSource?**

**Ans:**-Data source is dividing work among administrator and programmer/developer.

The administrator creates a DataSource object and ties up it with JNDI registry. A programmer/ developer retrieves the DataSource object from the registry.  Then it will establish the connection with the database.

**41) What is the reason why we need a JdbcRowSet like the wrapper around ResultSet?**

**Ans:**-We can use ResultSet object as a JavaBeans component.

A JdbcRowSet also can be used as a JavaBeans component. That’s why it can be created and configured at design or compile time and executed at run time.

All jdbcRowSet objects are scrollable and updatable.

**42) How many ways that we can view a result set?**

**Ans:**-There are 2 ways to view ResultSet

column

column index.

Example: getInt(String columnName), getInt(int columnIndex)

**43) How many ways can you update a result set?**

**Ans:**-Following methods helps you to update result set

updateRow()

deleteRow()

refreshRow()

cancelRowUpdates()

insertRow()

**44) Why should we close database connections in Java?**

**Ans:**-As a best practice, we must close the resultset, the statement and the connection. If the connection is coming from a pool, on closure, the connection is sent back to the pool for reuse. We are doing this in the finally{} block, because if any exception occurs, then we still get the chance to close this.

**45) Why are we using blob datatypes in JDBC?**

**Ans:**-These are used to store a large amount of data into the database like images, movie, etc.

**46) How to set the attribute Concurrency in ResultSet?**

**Ans:**-There are two concurrency levels

CONCUR\_READ\_ONLY – It is only for reading.

CONCUR\_UPDATABLE − It is for both read and updated.

**47) What is the difference between client and server database cursors?**

**Ans:**-Server side cursor means data & results are saved on the server. Only when requested data is sent to the client. Client side cursor means all data sent to the client location.

**SERVLET**

1. **What is different between web server and application server?**

**Ans:**A web server responsibility is to handler HTTP requests from client browsers and responds 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 the 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 the Application server is a web server with additional functionalities to help developers with enterprise applications.

1. **Which HTTP method is non-idempotent?**

**Ans:** An 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 the 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 the database, we should use the 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.

1. **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 use GET method.

1. **What is MIME Type?**

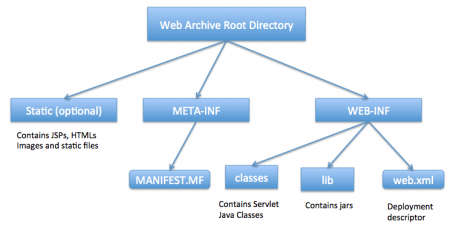
**Ans:** The “Content-Type” response header is known as MIME Type. Server sends MIME type to client to let them know the kind of data it’s sending. It helps client in rendering the data for user. Some of the mostly used mime types are text/html, text/xml, application/xml etc.

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 a file through servlet from the server.

1. **What is a web application and what is its directory structure?**

**Ans:** Web Applications are modules that run on the server to provide both static and dynamic content to the client browser. Apache webserver supports PHP and we can create a 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 the client browser.

[Java Web Applications](https://www.journaldev.com/1854/java-web-application-tutorial-for-beginners) are packaged as Web Archive (WAR) and it has a defined structure like below image.



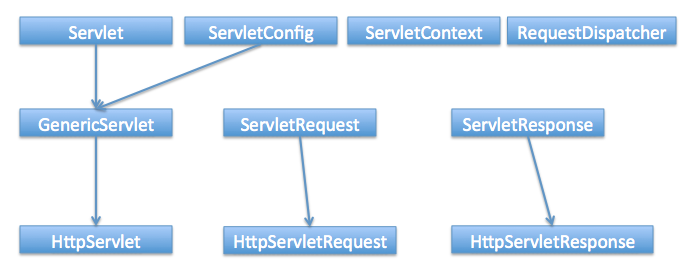
### [What is a servlet?](https://cdn.journaldev.com/wp-content/uploads/2013/08/WAR-directory-structure.png)

[Ans: Java Servlet is server side technologies to extend the capability of web servers by providing support for dynamic response and data persistence.](https://cdn.journaldev.com/wp-content/uploads/2013/08/WAR-directory-structure.png)

[The javax.servlet  and javax.servlet.http packages provide interfaces and classes for writing our own servlets.](https://cdn.journaldev.com/wp-content/uploads/2013/08/WAR-directory-structure.png)

[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.](https://cdn.journaldev.com/wp-content/uploads/2013/08/WAR-directory-structure.png)

[Most of the times, web applications are accessed using HTTP protocol and that’s why we mostly extend HttpServlet class. Servlet API hierarchy is shown in the below image.](https://cdn.journaldev.com/wp-content/uploads/2013/08/WAR-directory-structure.png)

[](https://cdn.journaldev.com/wp-content/uploads/2013/08/WAR-directory-structure.png)

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

**Ans:** Servlet technology was introduced to overcome the shortcomings of CGI technology.Servlets provide better performance than CGI in terms of processing time, memory utilization because servlets use benefits of [multithreading](https://www.journaldev.com/1079/multithreading-in-java) 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 the life cycle of servlet and we don’t need to worry about memory leaks, security, garbage collection, etc.
* Servlets are maintainable and the learning curve is small because all we need to take care is business logic for our application.

1. **What are common tasks performed by Servlet Container?**

**Ans:** 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 the container, we don’t need to build a server socket to listen for any request from the web client, parse the request and generate a 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. The container also provides utility like JNDI for resource pooling and management.
* **Multithreading Support**: Container creates a new thread for every request to the servlet and provides them request and response objects to the processing. So servlets are not initialized for each request and save 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.

1. **What is ServletConfig object?**

**Ans:** 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.

1. **What is ServletContext object?**

**Ans:** 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.

1. **What is difference between ServletConfig and ServletContext?**

**Ans:** 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.

1. **What is Request Dispatcher?**

**Ans:** RequestDispatcher interface is used to forward the request to another resource that can be HTML, JSP or another servlet in the 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.

1. **What is difference between PrintWriter and ServletOutputStream?**

**Ans:** 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.

1. **Can we get PrintWriter and ServletOutputStream both in a servlet?**

**Ans:** 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.

1. **How can we create deadlock situation in servlet?**

**Ans:** 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.

1. **What is the use of servlet wrapper classes?**

**Ans:** 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.

1. **What is SingleThreadModel interface?**

**Ans:** 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, that’s why this interface is Deprecated in Servlet 2.4.

1. **Do we need to override service() method?**

**Ans:** 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 the 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.

1. **Is it good idea to create servlet constructor?**

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

1. **What is difference between GenericServlet and HttpServlet?**

**Ans:** 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.

1. **What is the inter-servlet communication?**

**Ans:** 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.

1. **Are Servlets Thread Safe? How to achieve thread-safety in servlets?**

**Ans:** HttpServlet init() method and destroy() method are called only once in the 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.

1. **What is servlet attributes and their scope?**

**Ans:** 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.

1. **How do we call one servlet from another servlet?**

**Ans:** 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.

1. **How can we invoke another servlet in a different application?**

**Ans:** 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 the ServletResponse sendRedirect() method and provide the complete URL of another servlet. This sends the response to the client with the 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.

1. **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.

1. **Why HttpServlet class is declared abstract?**

**Ans:** 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.

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

**Ans:** 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 a 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 its init method by passing the servlet config object. This is the place where a servlet class transforms from normal class to servlet.
* Request Handling – Once the servlet is initialized, it’s 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 the destroy() method.

1. **What are life cycle methods of a servlet?**

**Ans:** 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.

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

**Ans:** If we have to initialize some resource before we want our servlet to process client requests, we should override the 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.

1. **What is URL Encoding?**

**Ans:** URL Encoding is the process of converting data into CGI form so that it can travel across the network without any issues. URL Encoding strips the white spaces and replaces 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”.

1. **What are different methods of session management in servlets?**

**Ans:** The session is a conversational state between client and server and it can consist 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

1. **What is URL Rewriting?**

**Ans:** 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 a 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.

1. **How does Cookies work in Servlets?**

**Ans:** 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 the request, since there is no point of adding Cookie to request, there are no methods to set or add a cookie to request.

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

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

**Ans:** 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.

1. **What is the difference between encodeRedirectUrl and encodeURL?**

**Ans:** 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.

1. **Why do we have servlet filters?**

**Ans:** 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 authorization 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.

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

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

1. **Why do we have servlet listeners?**

**Ans:** 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 the deployment descriptor (web.xml). What if our application is database-oriented and we want to set an attribute in ServletContext for Database Connection.

If your 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 the database is down or not configured properly, we won’t know until the first client request comes to the server. To handle these scenarios, servlet API provides Listener interfaces that we can implement and configure to listen to an event and do certain operations.

1. **How to handle exceptions thrown by application with another servlet?**

**Ans:** 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 the 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 a link to the application home page or some details to let the 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>

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

**Ans:** The deployment descriptor is a configuration file for the web application and its name is web.xml and it resides in WEB-INF directory. Servlet container uses 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.

1. **How to get the server information in a servlet?**

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

getServletContext().getServerInfo()

1. **How to get the actual path of servlet in server?**

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

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

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

**Ans:** Usually, servlet container loads a servlet on the first client request. Sometimes the servlet is heavy and takes time to loads, we might want to load it on application startup. We can use a load-on-startup element with servlet configuration in the web.xml file or use WebServlet annotation loadOnStartup variable to tell the 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 a negative integer then servlet container will load the servlet based on client requests and requirement but if it’s 0 or positive, then the 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.

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

**Ans:** 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.

1. **How to get the IP address of client in servlet?**

**Ans:** We can use request.getRemoteAddr() to get the client IP address in servlet.

1. **What are different ways for servlet authentication?**

**Ans:** Servlet Container provides different ways of login based servlet authentication:

* 1. **HTTP Basic Authentication**
  2. **HTTP Digest Authentication**
  3. **HTTPS Authentication**
  4. **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.

1. **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 its 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 the 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](https://www.journaldev.com/1877/servlet-tutorial-java#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 a 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 a 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 a 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 a 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.

1. **What is the use of Send Redirect () method?**

**Ans:** Send Redirect () method which works at the client side is used to redirect the response to another resource like Servlet, JSP, HTML.

**Syntax:** void send Redirect(URL);

1. **How Forward () method is different from Send Redirect () method?**

**Ans: Forward () method:**

a) It is used to send the exact same request to another resource.

b) It works on the server side within the server.

**Send Redirect () method:**

a) It always sends a new request to the resources as it uses URL.

b) It works at the client side both outside and within the server.

1. **What are the kinds of HTTP requests?**

**Ans:** Kinds of HTTP request include:

* Get
* Post
* Head
* Options
* Put
* Trace
* Delete

1. **Can you call a jsp from the servlet?**

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

1. RequestDispatcher rd=request.getRequestDispatcher("/login.jsp");
2. rd.forward(request,response);
3. **What are Cookies?**

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

1. **What is difference between Cookies and HttpSession?**

**Ans:** Cookie works at client side whereas HttpSession works at server side.

1. **How can we perform any action at the time of deploying the project?**

**Ans:** By the help of ServletContextListener interface.

1. **What is the disadvantage of cookies?**

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

1. **What are the annotations used in Servlet 3?**

**Ans:** There are mainly 3 annotations used for the servlet.

@WebServlet : for servlet class.

@WebListener : for listener class.

@WebFilter : for filter class.

1. **Which event is fired at the time of project deployment and undeployment?**

**Ans:** ServletContextEvent.

1. **Which event is fired at the time of session creation and destroys?**

**Ans:** HttpSessionEvent.

1. **Which event is fired at the time of setting, getting or removing attribute from application scope?**

**Ans:** ServletContextAttributeEvent.

1. **What is the use of welcome-file-list?**

**Ans:** It is used to specify the welcome file for the project.

1. **Define getParameter() method?**

**Ans:** This getParameter() method is used to get the parameter values from the ServletRequest Object into Servlet Program. This method is available from javax.servlet.ServletRequest.

1. **Define getInitParameter() method?**

**Ans:** This getInitParameter() method is used to get the initializaed parameter valaue from javax.servlet.ServletConfig and javax.servlet.ServletContext.

1. **Describe include() method?**

**Ans:** The include() method of the RequestDispatcher object includes the response of another Servlet into the calling Servlet. You can invoke this method any time from the calling Servlet while servicing request.

1. **Describe forward() method?**

**Ans:** The forward() method of RequestDispatcher forwards requests to the navigated resource, such as Servlet (target Servlet) and this method can be invoked by the Servlet while servicing the request when no output has been committed.

1. **What is diff between <init-param> and <context-param>?**

**Ans:** <init-param> tag part of web.xml will initialize the parameters in ServletConfig object.

<context-param> tag part of web.xml will initialize the parameters in ServletContext object.

1. **What Is Genericservlet Class?**

**Ans:** 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. **How Can The Session In Servlet Be Destroyed?**

**Ans:**There are two ways to destroy a session:  
1. **Programatically :** 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. **What Is Lazy Loading?**

**Ans:**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. **What Is The Procedure For Initializing A Servlet?**

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

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

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

1. **What Are The Uses Of Servletrequest?**

**Ans:**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. **What Are The Uses Of Servletresponse Interface?**

**Ans:**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. **How Http Servlet Handles Client Requests?**

**Ans:**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. **What Is Pre Initialization Of A Servlet?**

**Ans:**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. **How Do Servlets Handle Multiple Simultaneous Requests?**

**Ans:** 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. **What Is Servlet Chaining?**

**Ans:**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. **How Do You Communicate Between The Servlets?**

**Ans:** We can communicate between servlets by using RequestDespatcher interface and servlet chaining.

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

**Ans:** Servlet init parameters are for a single servlet only. No body out side that servlet can access that. It is declared inside the <servlet> tag inside Deployment Descriptor, where as 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”) where as method for accessing servlet init parameter is getServletConfig ().getInitParamter (“name”);

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

**Ans:** 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. **What Is Http Tunneling?**

**Ans:** 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. **What Are The Differences Between A Session And A Cookie?**

**Ans:** 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. **Why Should We Go For Inter Servlet Communication?**

**Ans:** 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. **What Is Client Side Refresh?**

**Ans:** 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. **What Is Server Side Push?**

**Ans:** 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. **What's The Servlet Interface?**

**Ans:** 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. **What is servlet mapping?**

**Ans:** Servlet Mapping is an association mapping between servlet and a URL pattern. This is used to map servlets with the requests.

1. **Which interface should be implemented by all servlets?**

**Ans:** Servlet interface should be implemented by all servlets.

1. **What is the difference between Servlet Request and Servlet Context when calling a Request Dispatcher?**

**Ans:** Relative URL can be called when Servlet Request is used and Relative URL is not used when using Servlet Context

1. **What are the features added in Servlet 2.5?**

**Ans:** Following are the features added in Servlet 2.5:

* Dependency on J2SE 5.0
* Support for annotations
* Loading the class
* Several web.xml
* Removed restrictions
* Edge case clarifications

1. **What are the supporting protocol by HttpServlet ?**

**Ans:** HttpServlet supports only HTTP and HTTPS protocol.

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

**Ans:** 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.

1. **Whether thread can be used in Servlets?**

**Ans:** Yes, Single thread can be used in servlets.

1. **What is the default HTTP method in the servlet?**

**Ans:** Default method is GET method for HTTPservlet.

1. **Which exception is thrown if servlet is not initialized properly?**

**Ans:** Servlet Exception or Unavailable Exception is thrown if servlet is not initialized properly.

1. **How to get the current HttpSession object?**

**Ans:** GetSession method is used to get the current HttpSession object on HttpservletRequest.

1. **What is HttpServlet and how it is different from GenericServlet?**

**Ans:** HttpServlet extends from GenericServlet and inherits the properties of Genericservlet. HttpServlet defines a HTTP protocol servlet while GenericServlet defines a generic, protocol-independent servlet.

1. **What is the difference between Server and Container?**

**Ans:** A server can provide service to the client and it contains one or more containers such as EJBs, Servlet, JSP containers. Containers hold set of objects.

1. **Why session tracking is needed?**

**Ans:** Every HTTP request needs to be captured by HTTP protocol and for that, state is captured. Tracking of state is called session tracking.

1. **When servlet is loaded?**

**Ans:** A servlet can be loaded when:

* First request is made
* Auto loading and Server starts up
* There is a single instance that answers all requests concurrently which  saves memory
* Administrator manually loads.