SERVLETS INTERVIEW QUESTIONS

Dear readers, these **Servlets Interview Questions** have been designed specially to get you acquainted with the nature of questions you may encounter during your interview for the subject of **Java Servlets**. As per my experience good interviewers hardly planned to ask any particular question during your interview, normally questions start with some basic concept of the subject and later they continue based on further discussion and what you answer:

Q: What is servlet?

A: A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request- response programming model. Before the servlet, CGI scripting language was used as server side programming language.

Q: What is the use of servlet?

A: Uses of servlet includes:

- Processing and storing data submitted by an HTML form.
- Providing dynamic content.
- A Servlet can handle multiple request concurrently and be used to develop high performance system
- Managing state information on top of the stateless HTTP.

Q: What is the life cycle of servlet?

A: Life cycle of Servlet:

- Servlet class loading
- Servlet instantiation
- Initialization (call the init method)
- Request handling (call the service method)
- Removal from service (call the destroy method)

Q: Why do we need constructor in servlet if we use the init ()?

A: Even though there is an init method in a servlet which gets called to initialize it, a constructor is still required to instantiate the servlet. Even though you as the developer would never need to explicitly call the servlet's constructor, it is still being used by the container.

Q: How servlet is loaded?

A: The servlet is loaded by:

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

Q: When the servlet is unloaded?

A: Servlet gets unloaded when:

- Server shuts down.
- Administrator manually unloads.

Q: What is servlet interface?

A: 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.

Q: What is the generic servlet class?

A: GenericServlet is an abstract class that implements the Servlet interface and the ServletConfig interface. In addition to the methods declared in these two interfaces, this class also provides simple versions of the lifecycle methods init () and destroy (), and implements the log method declared in the ServletContext interface.

Q: What is the difference between GenericServlet and HttpServlet?

A: The difference is:

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

Q: Why HttpServlet class is declared abstract?

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

Q: Can servlet have a constructor?

A: Yes

Q: What are the type of protocols supported by the HttpServlet?

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

Q: What is the difference between the doGet () and doPost ()?

A: The difference is:

- In doGet() the parameters are appended to the URL and sent along with header information. In doPost (),send the information through a socket back to the webserver and it won't show up in the URL bar.
- The amount of information you can send back using a GET is restricted as URLs can only be 1024 characters. You can send much more information to the server by using post and it's not restricted to textual data either. It is possible to send files and even binary data such as serialized Java objects!
- DoGet() is a request for information.It does not change anything on the server. (doGet () should be idempotent). doPost () provides information (such as placing an order for merchandise) that the server is expected to remember.

Q: When to use doGet() and when doPost()?

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

- If data is sensitive.

- Data is greater than 1024 characters.
- If your application don't need bookmarks.

Q: How do I support both doGet () and doPost () from same servlet?

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

Q: Should I override the service () method?

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

Q: What is the ServletContext?

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

Q: What is the difference between the ServletConfig and ServletContext interface?

A: The ServletConfig interface is implemented by the servlet container in order to pass configuration information to a servlet. The server passes an object that implements the ServletConfig interface to the servlet's init () method. A ServletContext defines a set of methods that a servlet uses to communicate with its servlet container.

Q: What is the difference between forward () and sendRedirect ()?

A: The difference is:

- A forward is performed internally by the servlet. A redirect is a two step process, where the web application instructs the browser to fetch a second URL, which differs from the original.
- The browser is completely unaware that it has taken place, so its original URL remains intact. But in sendRedirect, the browser, in this case, is doing the work and knows that it's making a new request.

Q: What is the difference between forward() and include()?

A: The RequestDispatcher include() method inserts the contents of the specified resource directly in the flow of the servlet response, as if it were part of the calling servlet. The RequestDispatcher forward() method is used to show a different resource in place of the servlet that was originally called.

Q: What is the use of servlet wrapper classes?

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

Q: What is a deployment descriptor?

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

Q: What is the preinitialization of servlet?

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

Q: What is the <load-on-startup> element?

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

Q: What is session?

A: A session refers to all the requests that a single client might make to a server in the course of viewing any pages associated with a given application. Sessions are specific to both the individual user and the application.

Q: What is the session tracking?

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

Q: What is the need of session tracking in web application?

A: HTTP is a stateless protocol. Every request is treated as new request. For web applications to be more realistic they have to retain information across multiple requests. Such information which is part of the application is referred as "state". To keep track of this state we need session tracking.

Q: What are the different types of session tracking?

A: Different types are:

- URL rewriting
- Hidden Form Fields
- Cookies
- Secure Socket Layer (SSL) Sessions

Q: How do I use cookies to store session state on client?

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

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

Q: What are the advantages of storing session state in cookies?

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

Q: What is URL rewriting?

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

Q: How can destroyed session in servlet?

A: Using session.invalidate() method.

Q: What is servlet lazy loading?

A: A container does not initialize the servlets as soon as it starts up; it initializes a servlet when it receives a request for that servlet first time. This is called lazy loading.

Q: What is servlet chaining?

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

Q: What is filter?

A: Filters are Java components that are used to intercept an incoming request to a Web resource and a response sent back from the resource. It is used to abstract any useful information contained in the request or response.

Q: What are the advantages of jsp over servlet?

A: The advantage of JSP is that they are document-centric. Servlets, on the other hand, look and act like programs. A Java Server Page can contain Java program fragments that instantiate and execute Java classes, but these occur inside an HTML template file and are primarily used to generate dynamic content.
Some of the JSP functionality can be achieved on the client, using JavaScript. The power of JSP is that it is server-based and provides a framework for Web application development.

Q: What is the life cycle of jsp?

A: Life cyle of jsp:

- Translation
- Compilation
- Loading the class
- Instantiating the class
- jspInit()
- _jspService()
- jspDestroy()

Q: What is the jspInit() method?

A: The jspInit() method of the javax.servlet.jsp.JspPage interface is similar to the init() method of servlets. This method is invoked by the container only once when a JSP page is initialized. It can be overridden by a page author to initialize resources such as database and network connections, and to allow a JSP page to read persistent configuration data.

Q: What is the _jspService ()?

A: The _jspService() method of the javax.servlet.jsp.HttpJspPage interface is invoked every time a new request comes to a JSP page. This method takes the HttpServletRequest and HttpServletResponse objects as its arguments. A page author cannot override this method, as its implementation is provided by the container.

Q: What is the jspDestroy ()?

A: The jspDestroy() method of the javax.servlet.jsp.JspPage interface is invoked by the container when a JSP page is about to be destroyed. This method is similar to destroy() method of servlets. It can be overridden by a page author to perform any cleanup operation such as closing a database connection.

Q: What jsp life cycle method can I override?

A: You cannot override the _jspService() method within a JSP page. You can however, override the jspInit() and jspDestroy() methods within a JSP page. JspInit() can be useful for allocating resources like database connections, network connections, and so forth for the JSP page. It is good programming practice to free any allocated resources within jspDestroy().

Q: What are implicit objects in jsp?

A: Implicit objects in JSP are the Java objects that the JSP Container makes available to developers in each page. These objects need not be declared or instantiated by the JSP author. They are automatically instantiated by the container and are accessed using standard variables; hence, they are called implicit objects.

Q: How many implicit objects are available in jsp?

A: These implicit objects are available in jsp:

- Request
- Response
- PageContext
- session
- application
- Out
- config
- page
- exception

Q: What are jsp directives?

A: JSP directives are messages for the JSP engine. i.e., JSP directives serve as a message from a JSP page to the JSP container and control the processing of the entire page.
They are used to set global values such as a class declaration, method implementation, output content type, etc. They do not produce any output to the client.

Q: What is page directive?

A: Page Directive is:

- A page directive is to inform the JSP engine about the headers or facilities that page should get from the environment.
- The page directive is found at the top of almost all of our JSP pages.
- There can be any number of page directives within a JSP page (although the attribute – value pair must be unique).
- The syntax of the include directive is: <%@ page attribute="value">

Q: What are the attributes of page directive?

A: There are thirteen attributes defined for a page directive of which the important attributes are as follows:

- **Import:** It specifies the packages that are to be imported.
- **Session:** It specifies whether a session data is available to the JSP page.
- **ContentType:** It allows a user to set the content-type for a page.
- **IsELIgnored:** It specifies whether the EL expressions are ignored when a JSP is translated to a servlet.

Q: What is the include directive?

A: Include directive is used to statically insert the contents of a resource into the current JSP. This enables a user to reuse the code without duplicating it, and includes the contents of the specified file at the translation time.

Q: What are the jsp standard actions?

A: The JSP standard actions affect the overall runtime behaviour of a JSP page and also the response sent back to the client. They can be used to include a file at the request time, to find or instantiate a Java Bean, to forward a request to a new page, to generate a browser-specific code, etc.

Q: What are the standards actions available in jsp?

A: The standards actions include:

- <jsp:include>
- <jsp:forward>
- <jsp:useBean>
- <jsp:setProperty>
- <jsp:getProperty>
- <jsp:param>
- <jsp:plugin>

Q: What is the <jsp: useBean> standard action?

A: The <jsp: useBean> standard action is used to locate an existing Java Bean or to create a Java Bean if it does not exist. It has attributes to identify the object instance, to specify the lifetime of the bean, and to specify the fully qualified class path and type.

Q: What is the scope available in <jsp: useBean>?

A: Scope includes:

- Page scope
- Request scope
- application scope
- session scope

Q: What is the <jsp:forward> standard action?

A: The <jsp:forward> standard action forwards a response from a servlet or a JSP page to another page. The execution of the current page is stopped and control is transferred to the forwarded page.

Q: What is the <jsp: include> standard action?

A: The <jsp: include> standard action enables the current JSP page to include a static or a dynamic resource at runtime. In contrast to the include directive, include action is used for resources that change frequently. The resource to be included must be in the same context.

Q: What is the difference between include directive and include action?

A: The difference is:

- Include directive, includes the content of the specified file during the translation phase–when the page is converted to a servlet. Include action, includes the response generated by executing the specified page (a JSP page or a servlet) during the request processing phase–when the page is requested by a user.
- Include directive is used to statically insert the contents of a resource into the current JSP. Include standard action enables the current JSP page to include a static or a dynamic resource at runtime.

Q: What is the difference between pageContext.include () and <jsp: include>?

A: The <jsp: include> standard action and the pageContext.include() method are both used to include resources at runtime. However, the pageContext.include () method always flushes the output of the current page before including the other components, whereas <jsp: include> flushes the output of the current page only if the value of

flush is explicitly set to true.

Q: What is the <jsp: setProperty> action?

A: You use jsp: setProperty to give values to properties of beans that have been referenced earlier.

Q: What is the <jsp: getProperty> action?

A: The <jsp: getProperty> action is used to access the properties of a bean that was set using the action. The container converts the property to a String as follows:

If it is an object, it uses the toString() method to convert it to a String. If it is a primitive, it converts it directly to a String using the valueOf() method of the corresponding Wrapper class.

The syntax of the <jsp: getProperty> method is: <jsp: getProperty name="Name" property="Property" />

Q: What is the <jsp: param> standard action?

A: The <jsp: param> standard action is used with <jsp: include> or <jsp: forward> to pass parameter names and values to the target resource.

Q: What is the <jsp: plugin> action?

A: This action lets you insert the browser-specific OBJECT or EMBED element needed to specify that the browser run an applet using the Java plug in.

Q: What is the scripting element?

A: JSP scripting elements let you insert Java code into the servlet that will be generated from the current JSP page.

- Expressions
- Scriptlet
- Declarations
- comment

Q: What is the scriptlet?

A: A scriptlet contains Java code that is executed every time a JSP is invoked. When a JSP is translated to a servlet, the scriptlet code goes into the service() method.
Hence, methods and variables written in scriptlet are local to the service() method. A scriptlet is written between the <% and %>tags and is executed by the container at request processing time.

Q: What is the jsp declaration?

A: JSP declarations are used to declare class variables and methods in a JSP page. They are initialized when the class is initialized. Anything defined in a declaration is available for the whole JSP page. A declaration block is enclosed between the <%! and %>tags. A declaration is not included in the service() method when a JSP is translated to a servlet.

Q: What is the jsp expression?

A: A JSP expression is used to write an output without using the out.print statement. It can be said as a shorthand representation for scriptlet. An expression is written between the <%= and %> tags. It is not required to end the expression with a semicolon, as it implicitly adds a semicolon to all the expressions within the expression tags.

Q: How is scripting disabled?

A: Scripting is disabled by setting the scripting-invalid element of the deployment descriptor to true. It is a subelement of jsp-property-group. Its valid values are true and false.

Q: Why is _jspService () start with '_'?

A: _jspService() method will be written by the container hence any methods which are not to be overridden by

the end user are typically written starting with a '_'. This is the reason why we don't override _jspService() method in any JSP page.

Q: How to pre-compile jsp?

A: Add jsp_precompile as a request parameter and send a request to the JSP file. This will make the jsp pre-compile. *http://localhost:8080/jsp1/test.jsp?jsp_precompile=true*
It causes execution of JSP life cycle until jspInit() method without executing _jspService() method.

Q: What is the benefit of pre-compile jsp page?

A: It removes the start-up lag that occurs when a container must translate a JSP page upon receipt of the first request.

Q: What is the difference between variable declared inside the declaration tag and variable declared in scriptlet?

A: Variable declared inside declaration part is treated as a instance variable and will be placed directly at class level in the generated servlet. Variable declared in a scriptlet will be placed inside _jspService () method of generated servlet. It acts as local variable.

Q: What are the three kind of comment in jsp?

A: These are the three types of commenst in jsp:

- JSP Comment: <%-- this is jsp comment -- %>
- HTML Comment: <!-- this is HTMl comment -- >
- Java Comments: <% // single line java comment /* this is multiline comment */ %>

Q: What is the output comment?

A: The comment which is visible in the source of the response is called output comment. <!-- this is HTMl comment -- >

Q: What is a hidden comment?

A: This is also known as JSP comment and it is visible only in the JSP and in rest of phases of JSP life cycle it is not visible. <%-- this is jsp comment -- %>

Q: How does jsp handle the run time exception?

A: You can use the errorPage attribute of the page directive to have uncaught run-time exceptions automatically forwarded to an error processing page.

Q: How can I implement the thread safe jsp page?

A: You can make your JSPs thread-safe by having them implement the SingleThreadModel interface. This is done by adding the directive in the JSP. <%@ page isThreadSafe="false" %>

Q: Is there a way to reference the “this” variable within the jsp?

A: Yes, there is. The page implicit object is equivalent to "this", and returns a reference to the generated servlet.

Q: Can you make the use of servletOutputStream object within jsp?

A: Yes. By using getOutputStream() method on response implicit object we can get it.

Q: What is autoflush?

A: This command is used to autoflush the contents. If a value of true is used it indicates to flush the buffer whenever it is full. In case of false it indicates that an exception should be thrown whenever the buffer is full. If you are trying to access the page at the time of conversion of a JSP into servlet will result in error.

Q: What is the different scope available in jsp?

A:The different scopes are:

- **Page:** Within the same page.
- **Request:** After forward or include also you will get the request scope data.
- **Session:** After sendRedirect also you will get the session scope data. All data stored in session is available to end user till session closed or browser closed.
- **Application:** Data will be available throughout the application. One user can store data in application scope and other can get the data from application scope.

Q: When to use application scope?

A: If we want to make our data available to the entire application then we have to use application scope.

Q: Can a jsp page instantiate a serialized bean?

A: No problem! The use Bean action specifies the bean Name attribute, which can be used for indicating a serialized bean.

Q: In which situation we can use the static include and dynamic include?

A: If the target resource won't change frequently, then it is recommended to use include directives. If the target resource will change frequently, then it is recommended to use include action.

Q: What is the JDBC?

A: Java Database Connectivity (JDBC) is a standard Java API to interact with relational databases form Java. JDBC has set of classes and interfaces which can use from Java application and talk to database without learning RDBMS details and using Database Specific JDBC Drivers

Q: What are the basic steps of using jdbc in java?

A: The basic steps are:

- Load the RDBMS specific JDBC driver because this driver actually communicates with the database.
- Open the connection to database which is then used to send SQL statements and get results back.
- Create JDBC Statement object. This object contains SQL query.
- Execute statement which returns resultset(s). Resultset contains the tuples of database table as a result of SQL query.
- Process the result set.
- Close the connection.

Q: What are the main component of jdbc?

A: The main components are:

- DriverManager
- Driver
- Connection
- Statement
- Resultset

Q: What is DriverManager?

A: DriverManager is a static class. It manages a list of database drivers. Matches connection requests from the

java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain sub protocol under JDBC will be used to establish a database Connection.

Q: What is Driver?

A: The JDBC API defines the Java interfaces and classes that programmers use to connect to databases and send queries. A JDBC driver implements these interfaces and classes for a particular DBMS vendor.database communications link, handling all communication with the database.
Normally, once the driver is loaded, the developer need not call it explicitly.

Q: What is the connection?

A: Interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only

Q: What is the statement?

A: Encapsulates an SQL statement which is passed to the database.

Q: What is the resultset?

A: The Resultset represents set of rows retrieved due to query execution.

Q: How we load a database driver with JDBC?

A: Provided the JAR file containing the driver is properly configured, just place the JAR file in the classpath. Java developers NO longer need to explicitly load JDBC drivers using code like Class.forName() to register a JDBC driver.
The DriverManager class takes care of this by automatically locating a suitable driver when the DriverManager.getConnection() method is called. This feature is backward-compatible, so no changes are needed to the existing JDBC code.

Q: What is the JDBC Driver interface?

A: The JDBC Driver interface provides vendor-specific implementations of the abstract classes provided by the JDBC API. Each vendor driver must provide implementations of the java.sql.Connection,Statement,PreparedStatement, CallableStatement, ResultSet and Driver

Q: What does the connection objects represents?

A: The connection object represents communication context, i.e., all communication with database is through connection object only.

Q: What is the statement?

A: Statement acts like a vehicle through which SQL commands can be sent. Through the connection object we create statement kind of objects.

Q: What is the prepared statement?

A: A prepared statement is an SQL statement that is precompiled by the database. Through precompilation, prepared statements improve the performance of SQL commands that are executed multiple times. Once compiled, prepared statements can be customized prior to each execution by altering predefined SQL parameters.

Q: What is the difference between statement and PreparedStatement?

A: The difference is:

- A standard Statement is used to create a Java representation of a literal SQL statement and execute it on the database. A PreparedStatement is a precompiled statement. This means that when the PreparedStatement is executed, the RDBMS can just run the PreparedStatement SQL statement without having to compile it first.

- Statement has to verify its metadata against the database every time. While a prepared statement has to verify its metadata against the database only once.

- If you want to execute the SQL statement once go for STATEMENT. If you want to execute a single SQL statement multiple number of times, then go for PREPAREDSTATEMENT. PreparedStatement objects can be reused with passing different values to the queries

Q: What is the callable statement?

A: Callable statements are used from JDBC application to invoke stored procedures and functions.

Q: How to call a stored procedure from jdbc?

A: PL/SQL stored procedures are called from within JDBC programs by means of the prepareCall() method of the Connection object created. A call to this method takes variable bind parameters as input parameters as well as output variables and creates an object instance of the CallableStatement class.

Q: What are the types of JDBC Driver?

A: The types are:

- Type 1: JDBC/ODBC
- Type2: Native API (partly-Java driver)
- Type 3: Open Protocol-Net
- Type 4: Proprietary Protocol-Net(pure Java driver)

Q: Which type of jdbc driver is the faster one?

A: JDBC Net pure Java driver(Type IV) is the fastest driver because it converts the JDBC calls into vendor specific protocol calls and it directly interacts with the database.

Q: Does the JDBC-ODBC Bridge support multiple concurrent open statements per connection?

A: No, You can open only one Statement object per connection when you are using the JDBC-ODBC Bridge.

Q: What are the standard isolation levels defined by the jdbc?

A: The standard isolation levels are:

- TRANSACTION_NONE
- TRANSACTION_READ_COMMITTED
- TRANSACTION_READ_UNCOMMITTED
- TRANSACTION_REPEATABLE_READ
- TRANSACTION_SERIALIZABLE

Q: What is the resultset?

A: The ResultSet represents set of rows retrieved due to query execution. Example: ResultSetrs = stmt.executeQuery(sqlQuery);

Q: What are the types of resultset?

A: The types are:

- TYPE_FORWARD_ONLY specifies that a resultset is not scrollable, that is, rows within it can be advanced only in the forward direction.
- TYPE_SCROLL_INSENSITIVE specifies that a resultset is scrollable in either direction but is insensitive to changes committed by other transactions or other statements in the same transaction.
- TYPE_SCROLL_SENSITIVE specifies that a resultset is scrollable in either direction and is affected by changes committed by other transactions or statements within the same transaction.

Q: What is the difference between TYPE_SCROLL_INSENSITIVE and TYPE_SCOLL_SENSITIVE?

A: An insensitive resultset is like the snapshot of the data in the database when query was executed. A sensitive resultset does NOT represent a snapshot of data; rather it contains points to those rows which satisfy the query condition.
After we get the resultset the changes made to data are not visible through the resultset, and hence they are known as insensitive. After we obtain the resultset if the data is modified then such modifications are visible through resultset.

Q: What is the RowSet?

A: A RowSet is an object that encapsulates a set of rows from either Java Database Connectivity (JDBC) result sets or tabular data sources like a file or spreadsheet. RowSets support component-based development models like JavaBeans, with a standard set of properties and an event notification mechanism.

Q: What are the different types of RowSet?

A:The different types are:

- Connected - A connected RowSet object connects to the database once and remains connected until the application terminates.
- Disconnected - A disconnected RowSet object connects to the database, executes a query to retrieve the data from the database and then closes the connection. A program may change the data in a disconnected RowSet while it is disconnected. Modified data can be updated in the database after a disconnected RowSet re-establishes the connection with the database.

Q: What is the need of BatchUpdates?

A: The BatchUpdates feature allows us to group SQL statements together and send to database server in one single trip.

Q: What is the data source?

A: A DataSource object is the representation of a data source in the Java programming language. In basic terms,

- A DataSource is a facility for storing data.
- DataSource can be referenced by JNDI.
- Data Source may point to RDBMS; file System, any DBMS etc.

Q: What are the advantages of data source?

A: The advantages are:

- An application does not need to hardcode driver information, as it does with the DriverManager.
- The DataSource implementations can easily change the properties of data sources.
- The DataSource facility allows developers to implement a DataSource class to take advantage of features like connection pooling and distributed transactions.

Q: What is the main advantage of connection pooling?

A: A connection pool is a mechanism to reuse connections created. Connection pooling can increase performance dramatically by reusing connections rather than creating a new physical connection each time a connection is requested.

Q: What is the multi programming?

A: Multiprogramming is a rapid switching of the CPU back and forth between processes.

Q: What is the difference between TCP and UDP?

A: TCP is designed to provide reliable communication across a variety of reliable and unreliable networks and internets.UDP provides a connectionless so it isbasically an unreliable service. Delivery and duplicate protection are not guaranteed.

Q: What is socket?

A: The combination of an IP address and a port number is called a socket.

Q: What is the advantage of java socket?

A: The advantages are:

- Sockets are flexible and sufficient.
- Efficient socket based programming can be easily implemented for general communications.
- Sockets cause low network traffic.

Q: What is the disadvantage of java socket?

A: The disadvantages are:

- Security restrictions are sometimes overbearing because a Java applet running in a Web browser is only able to establish connections to the machine where it came from, and to nowhere else on the network.
- Despite all of the useful and helpful Java features, Socket based communications allows only to send packets of raw data between applications. Both the client-side and server-side have to provide mechanisms to make the data useful in any way.
- Since the data formats and protocols remain application specific, the re-use of socket based implementations is limited.

Q: What is RMI?

A: It stands for Remote Method Invocation. RMI is a set of APIs that allows to build distributed applications. RMI uses interfaces to define remote objects to turn local method invocations into remote method invocations.

Q: What is socket()?

A: The socket () is very similar to socketPair() except that only one socket is created instead of two. This is most commonly used when if the process you wish to communicate with is not the child process.

Q: What is ServerSocket?

A: The ServerSocket class is used to create serverSocket. This object is used to communicate with client.

Q: What is bind()?

A: It binds the socket to the specified server and port in the SocketAddress object. Use this method if you instantiated the ServerSocket using the no-argument constructor.

Q: What is the Datagram?

A: A datagram is an independent, self-contained message sent over the network whose arrival, arrival time, and content are not guaranteed.

Q: What is getLocalPort()?

A: It returns the port that the server socket is listening on. This method is useful if you passed in 0 as the port number in a constructor and let the server find a port for you.

Q: What is accept()?

A: It waits for an incoming client. This method blocks until either a client connects to the server on the specified port or the socket times out, assuming that the time-out value has been set using the setSoTimeout() method. Otherwise, this method blocks indefinitely.

Q: What is the network interface?

A: A network interface is the point of interconnection between a computer and a private or public network. A network interface is generally a network interface card (NIC), but does not have to have a physical form.

Q: What is the encapsulation technique?

A: Hiding data within the class and making it available only through the methods. This technique is used to protect your class against accidental changes to fields, which might leave the class in an inconsistent state.

Q: How does the race condition occur?

A: It occurs when two or more processes are reading or writing some shared data and the final result depends on who runs precisely when.

Q: What information is needed to create a TCP Socket?

A: Socket is created from this information:

- **Local System's:** IP Address and Port Number
- **Remote System's:** IPAddress and Port Number

What is Next ?

Further you can go through your past assignments you have done with the subject and make sure you are able to speak confidently on them. If you are fresher then interviewer does not expect you will answer very complex questions, rather you have to make your basics concepts very strong.

Second it really doesn't matter much if you could not answer few questions but it matters that whatever you answered, you must have answered with confidence. So just feel confident during your interview. We at tutorialspoint wish you best luck to have a good interviewer and all the very best for your future endeavor. Cheers :-)