**1.What is DTD?**

**Answer: DTD means document type declaration. A document type declaration specifying and external document Type definition(DTD) that identifies markup declarations for the elements used in the body of the document or explicit markup declaration or both.**

1. **What are the benefits of DTD?**

Answer:

With a DTD, each of your XML files can carry a description of its own format.

With a DTD, independent groups of people can agree to use a standard DTD for interchanging data.

Your application can use a standard DTD to verify that the data you receive from the outside world is valid.

You can also use a DTD to verify your own data.

1. **Write the benefit rule of Schema?**

Answer:

* It is easier to describe allowable document content
* It is easier to validate the correctness of data
* It is easier to work with data from a database
* It is easier to define data facets (restrictions on data)
* It is easier to define data patterns (data formats)
* It is easier to convert data between different data types

1. **Write the 4 class for XML parser?**

Answer: 4 class for xml parser

* 1. SAXParserFactory
  2. SAXParser
  3. DocumentBuilderFactory
  4. DocumentBuilder
* **What are welform document and its rule?**

Answer: When an XML document is said to be well-formed, it just means that it conforms to the rules for writing XML as defined by the XML specification.

The rules for a document to be well-formed are as follows:

1. If the XML declaration appears in the prolog, it must include the XML version. Other specifica-tions in the XML document must be in the prescribed sequence — character encoding followed by standalone specification.

2. If the document type declaration appears in the prolog, the DOCTYPE name must match that of the root element, and the markup declarations in the DTD must be according to the rules for writing markup declarations.

3. The body of the document must contain at least one element, the root element, which contains all the other elements, and an instance of the root element must not appear in the content of another element. All elements must be properly nested.

4. Elements in the body of the document must be consistent with the markup declarations identi-fied by the DOCTYPE declaration.

The rules for writing an XML document are absolutely strict.

* **Write the 4 class for XML parser?**

Answer: The javax.xml.parsers package defines four classes supporting the processing of XML documents:

|  |  |
| --- | --- |
| SAXParserFactory | Enables us to create a configurable factory object that we can use to create a SAXParser object encapsulating a SAX-based parser |
| SAXParser | Defines an object that wraps a SAX-based parser |
| DocumentBuilderFactory | Enables us to create a configurable factory object that we can use to create a DocumentBuilder object encapsulating a DOM-based parser |
| DocumentBuilder | Defines an object that wraps a DOM-based parser |

All four classes are abstract.

* **What is default handler class?**

Answer: the DefaultHandler class defines do-nothing methods declared in the other three interfaces. For example, when a parsing error occurs, the parser calls a method to report the error. Three methods for error reporting are declared in the ErrorHandler inter-face and are implemented by the DefaultHandler class.

1. **Difference between PreparedStatement & Statement?**

Answer:PreparedStatement objects are great when:

❑ we need to execute the same statement several times and need to change only specific values.

❑ we are working with large chunks of data that make concatenation unwieldy.

❑ we are working with a large number of parameters in the SQL statement that make string con-catenation unwieldy.

Conversely, Statement objects work well when we have simple statements; and of course, we have no

option if JDBC driver doesn’t support the PreparedStatement interface.

1. **Define SQL state, SQL exception, SQL warning.**

SQL State

The SQL state is a string that contains a state as defined by the X/Open SQL standard. We obtain the SQL state value from the SQLException object by calling its getSQLState() method.

SQLWarnings

JDBC provides a means of obtaining warning information from JDBC objects. Sometimes conditions may arise that may not be serious enough to throw an exception, but do merit the program being signaled that all is not completely well. Warnings are represented by objects of type java.sql.SQLWarning, and an SQLWarning object is silently appended to a JDBC object when an operation using the object causes something odd to occur.

1. **What is DefaultMutableTreeNode & DefaultTreeModel?**

We can create our own class to define nodes, the DefaultMutableTreeNode class in the javax.swing.tree package is adequate for most purposes. This class implements the MutableTreeNode interface and adds a few more methods of its own.

The javax.swing.tree package includes a DefaultTreeModel class that implements TreeModel and that we can use “as is” in many situations. We can create a DefaultTreeModel object using a single node that is the root node for our tree.

For example:

DefaultMutableTreeNode dbNode = new DefaultMutableTreeNode(“No Database”);

DefaultTreeModel dbTreeModel = new DefaultTreeModel(dbNode);

1. **How you get Database Metadata?**

We need to obtain the database metadata that will be displayed by the JTree object in the left scroll

pane. We can do this when a connection has been established, for which we need the database URL

plus the user ID and password. Before you can establish a connection, you want to be sure the driver is

loaded, so add the following data member to the DatabaseBrowse class to store the driver names:

private String[] drivers = {“sun.jdbc.odbc.JdbcOdbcDriver”, // ODBC bridge

“com.imaginary.sql.msql.MsqlDriver” // mSQL driver};

1. **How you get Table Name & Column Name?**

We can get information about the tables within the database with the getTables() method, which is of the following form:

ResultSet getTables(String catalog,String schemaPattern,String tableNamePattern,String[] types)

The third parameter is a pattern for selecting the table names, with the pattern defined as described in the preceding paragraph. Only data on tables with names corresponding to the pattern will be returned.

To get the column names for particular tables, we can call the getColumns() method for a Database MetaData object. The first three arguments are the catalog, schemaPattern, and tableNamePattern, as described for the getTables() method. The fourth argument is a String object specifying a pattern for selecting column names.

**1.What is JDBC?**

**Answer:** JDBC is, by definition, an interface to relational data source. JDBC was designed as an object-oriented Java-based application program interface (API) for database access.

**2.What are the common tasks or steps of JDBC?**

**Answer:** The steps of loading JDBC is given below:

1. Import the necessary classes.
2. Load the JDBC driver.
3. Identify the data source.
4. Allocate a Connection object.
5. Allocate a Statement object.
6. Execute a query using the Statement object.
7. Retrieve data from the returned ResultSet object.
8. Close the ResultSet.
9. Close the Statement object.
10. Close the Connection object.

**3.What Class.forName will do while loading drivers of JDBC?**

**Answer:** By the Class.forName() we load a JDBC Driver Class under DriverManager. forName() is a public static Method that is throws ClassNotFoundException. When we used System.setProperties() for loading JDBC Driver we need to Security manager clearance, otherwise throws SecurityException but Class.forName() we don’t need that.

**5.What is SQLException?**

**Answer:** : SQLException is a subclass of an exception. At runtime Java programming Technology return SQLException When a sql statement error . Every method of every JDBC class and interface can throw an exception of type **SQLException.**

**6.What is ResultSet object?**

**Answer:**  java.sql.ResultSet is a java object that is used for database connectivity to hold the data returned by a select query. When we run a select query it returns us the data in a table format with each row representing one logical group of data with a number of columns.   
The result set would contain this table of data and each row can be accessed one by one. we can use the resultset.get() methods to get the data from it.

**7.What packages are used by JDBC?**

**Answer:** JDBC used java. Sql package for Basic JDBC Classes and Interfaces and DataTypes it also map with java.util for date, java.math for BigDecimal.

**8.What is the difference between executequery () and executeupdate ()?**

**Answer:** The difference between executeQuery and executeUpdate is that executeUpdate is for executing statements that change data in the database. For example, use executeUpdate to execute a CREATE an INSERT or an UPDATE statement. executeUpdate returns an int, and the value of that int corresponds to the number of records that were modified. While executeQuery() method is used for executing SQL statements and it returns ResultSet object.

**9.What is TableModel?**

**Answer:** The TableModel interface declares methods that are used by a JTable object to access

the data item to be displayed at each position in the table. This interface is defined in

the javax.swing.table package, along with the JTable class.

**19.What are the layers of RMI Architecture?**

**Answer:** The three layers are the Stub and Skeleton Layer, the Remote Reference Layer, and the Transport Layer.

**The stub and skeleton layer** is responsible for marshaling and unmarshaling the data and transmitting and receiving them to/from the Remote Reference Layer.

**The Remote reference layer** is responsible for carrying out the invocation.

**The Transport layer** is responsible for setting up connections, managing requests, monitoring them and listening for incoming calls.

**20.What is the role java.rmi.Naming Class?**

**Answer:** The Naming class provides methods for storing and obtaining references to remote objects in the remote object registry.

**21.What is the use of UnicastRemoteObject in RMI?**

**Answer:** The UnicastRemoteObject class provides support for point-to-point active object references using TCP streams. Objects that require remote behavior should extend UnicastRemoteObject.

**22.What is the difference between using bind() and rebind() methods of Naming Class?**

**Answer:** bind method(String name) binds the specified name to a remote object while rebind(String name) method rebinds the specified name to a new remote object,any existing binding for the name is replaced.

**23.What is Stub & skeleton?**

**Answer:** A stub is a remote object at the client-side. This stub implements all the interfaces which remote object implementation supports. The role of the stubs is to marshal and unmarshal the messages that are sent and received on the client or the server side.

A skeleton is a remote object at the server-side. This stub consists of methods that invokes dispatch calls to the remote implementation of objects.

**24.What is DefaultMutableTreeNode & DefaultTreeModel?**

**Answer:** The DefaultMutableTreeNode class in the javax.swing.tree package is adequate for most purposes. This class implements the MutableTreeNode interface and adds a few more methods of its own. The javax.swing.tree package includes a DefaultTreeModel class that implements TreeModel. s We can create a DefaultTreeModel object using a single node that is the root node for our tree.

For example:

DefaultMutableTreeNode dbNode = new DefaultMutableTreeNode(“No Database”);

DefaultTreeModel dbTreeModel = new DefaultTreeModel(dbNode);